UNITED STATES PATENT AND TRADEMARK OFFICE

Konda Technologies, Inc
6278 GRAND OAK WAY
SAN JOSE CA 95135

MAILED

NOV 2 7 2019

INTERNATIONAL PATENT LEGAL ADM.

In re Application of   :
Konda Technologies Inc.   :
Application No.: 16/202,067   :   DECISION
Filing Date: 27 November 2018   :
Attorney Docket No.: V-0070US :

This Decision is in response to the filing of "Petition To Withdraw the Prior Filed Petition on 11/27/2018 To Accept The Unintentionally Delayed Claim Under 35 U.S.C. 119(e)," in the United States Patent and Trademark Office on 17 September 2019.

This application does not contain a delayed claim of benefit. A petition for the unintentionally delayed payment of the basic national fee under 37 CFR 1.137(a) was accepted in 12/601,275.

Applicant's request to withdraw the petition and refund the petition fee is **GRANTED**.

/Erin P. Thomson/
Erin P. Thomson
Attorney Advisor
International Patent Legal Administration
571-272-3292

Electronic Acknowledgement Receipt

EFS ID:	37235943
Application Number:	16202067
International Application Number:	
Confirmation Number:	8134
Title of Invention:	VLSI Layouts of Fully Connected Generalized Networks
First Named Inventor/Applicant Name:	Venkat Konda
Customer Number:	38139
Filer:	Venkat Konda
Filer Authorized By:	
Attorney Docket Number:	V-0070US
Receipt Date:	23-SEP-2019
Filing Date:	27-NOV-2018
Time Stamp:	01:08:04
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Reissue dec filed in accordance with MPEP 1414	V0070US-aia0005.pdf	218167 eb6d66b02b0eb8f341153179cc12222cd750b95d	no	3

Warnings:

Information:

2	Preliminary Amendment	V0070US-PrelimAmend.pdf	238878 4e11df3fe54386626c1600c646245d906699 7be9	no	26

Warnings:

Information:

Total Files Size (in bytes):	457045

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.
National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.
New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

Doc Code: REIS.DECL

Document Description: Reissue Declaration Filed In Accordance With MPEP 1414

REISSUE APPLICATION DECLARATION BY THE INVENTOR	Docket Number (Optional)
	V-0070US

I hereby declare that:

Each inventor's residence and mailing address are stated below next to their name.

I believe I am the original inventor or an original joint inventor of the subject matter which is described and claimed in patent number 8,269,523 , granted September 18, 2012 and for which a reissue patent is sought on the invention titled VLSI Layouts of Fully Connected Generalized Networks ,

the specification of which

☐ is attached hereto.

☑ was filed on 11/27/2018 as reissue application number 16/202,067 .

The above-identified application was made or authorized to be made by me.

I hereby acknowledge that any willful false statement made in this declaration is punishable under 18 U.S.C. 1001 by fine or imprisonment of not more than five (5) years, or both.

I believe the original patent to be wholly or partly inoperative or invalid, for the reasons described below. (Check all boxes that apply.)

☐ by reason of a defective specification or drawing.

☑ by reason of the patentee claiming more or less than he had the right to claim in the patent.

☐ by reason of other errors.

At least one error upon which reissue is based is described below. If the reissue is a broadening reissue, a claim that the application seeks to broaden must be identified:

Because the patentee claimed more than he had the right to claim in the US Patent No. 8,269,523, the scope of the original claim 1 is narrowed by incorporating limitations of claim 3 into original claim 1. Newly added independent claims 49 and 50 also do not broaden the scope of original claims.

[Page 1 of 2]

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

(REISSUE APPLICATION DECLARATION BY THE INVENTOR, page 2)	Docket Number (Optional) V-0070US

Note: To appoint a power of attorney, use form PTO/AIA/81.

Correspondence Address: Direct all communications about the application to:

☑ The address associated with Customer Number: 38139

OR

☐ Firm or Individual Name	
Address	

City		State		Zip	

Country	

Telephone		Email	

WARNING:

Petitioner/applicant is cautioned to avoid submitting personal information in documents filed in a patent application that may contribute to identity theft. Personal information such as social security numbers, bank account numbers, or credit card numbers (other than a check or credit card authorization form PTO-2038 submitted for payment purposes) is never required by the USPTO to support a petition or an application. If this type of personal information is included in documents submitted to the USPTO, petitioners/applicants should consider redacting such personal information from the documents before submitting them to the USPTO. Petitioner/applicant is advised that the record of a patent application is available to the public after publication of the application (unless a non-publication request in compliance with 37 CFR 1.213(a) is made in the application) or issuance of a patent. Furthermore, the record from an abandoned application may also be available to the public if the application is referenced in a published application or an issued patent (see 37 CFR 1.14). Checks and credit card authorization forms PTO-2038 submitted for payment purposes are not retained in the application file and therefore are not publicly available.

Legal name of sole or first inventor (E.g., Given Name (first and middle (if any) and Family Name or Surname)
Venkat Konda

Inventor's Signature /Venkat Konda/	Date (Optional) September 23, 2019

Residence: City San Jose	State CA	Country USA

Mailing Address
6278 Grand Oak Way

City San Jose	State CA	Zip 95135	Country USA

☐☐ Additional joint inventors are named on the _____ supplemental sheet(s) PTO/AIA/10 attached hereto.

[Page 2 of 2]

Privacy Act Statement

The **Privacy Act of 1974 (P.L. 93-579)** requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (*i.e.*, GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

Reissue Application No. 16/202,067   Attorney Docket No. V-0070US
Reissue Application Filed: 11/27/18

In The United States Patent And Trademark Office

Reissue Application	:	16/202,067	Filed: November 27, 2018
Parent US Patent No	:	8,269,523	Issued: September 18, 2012
Parent Application No:		12/601,275	Filed: May 31, 2010
Examiner	:	Ton, My Trang	Group Art Unit: 3992
Applicant(s): Venkat Konda			Filed: November 27, 2018

5

Title: VLSI Layouts of Fully Connected Generalized Networks

San Jose, 2019 Sept 23, Mon

10   **AMENDMENT ACCOMPANYING REISSUE APPLICATION RESUBMITTED PURSUANT TO 37 C.F.R. §1.173(B)**

Mail Stop Reissue

Commissioner for Patents

15   P.O. Box 1450

Alexandria, VA 22313-1450

Dear Sir/Madam:

20   For the above above-referenced reissue patent application, in view of the petition granted on August 19, 2019 that was filed in the parent Patent No. 8,269,523 for the entire delay in filing the basic national fee from the due date for the fee until the filing of a grantable petition under 37 CFR 1.137 was unintentional, applicant is now submitting a new "the reissue application declaration by the inventor". Furthermore the Amendment

25   submitted for the above-referenced reissue patent application filed on November 27, 2018 which supposed to set forth applicant's requested amendments to U.S. Patent No.

Reissue Application No. 16/202,067   Attorney Docket No. V-0070US
Reissue Application Filed: 11/27/18

8,269,523 is non-compliant with 37 C.F.R. §1.173(B) at least since the submitted claims
are not identified with correct format.

Accordingly applicant respectfully requests to disregard the amendment submitted
on November 27, 2018 and consider this amendment. This Amendment accompanies the
reissue patent application filed on November 27, 2018, and sets forth applicant's
requested amendments to U.S. Patent No. 8,269,523.

Amendment to Specification is set forth on page 4.

Amendments to the claims are set forth in the ***Listing of the Claims,*** which begins on
Page 5. Originally issued claims 1, 2, 4, 7, 11, 16, 22, 24, 28, 33, 36, 39, 43 are amended.
Originally issued claims 3, 5, 6, 8 - 10, 12 - 14, 15, 17 – 21, 23, 25 – 27, 29 – 32, 34 – 35,
37 – 38, 40 – 42, 44 - 48 are now cancelled. Originally issued claims are amended
perfecting the claim language. But no new matter is added. Furthermore:

Incorporated limitations of Claim 3 into Claim 1 and cancelled claim 3.

Incorporated limitations of Claims 5 – 6 into Claim 4 and cancelled claims 5 – 6.

Incorporated limitations of Claims 8 – 10 into Claim 7 and cancelled claims 8 – 10.

Incorporated limitations of Claims 12 – 14 into Claim 11 and cancelled claims 12 – 14.

Incorporated limitations of Claims 18, 19, 21, 22, and 47 into Claim 16 and cancelled
claims 18, 19, 21, 22, and 47.

Incorporated limitations of Claims 25 – 27 into Claim 24 and cancelled claims 25 – 27.

Incorporated limitations of Claims 29 – 32 into Claim 28 and cancelled claims 29 – 32.

Incorporated limitations of Claims 34 – 35 into Claim 33 and cancelled claims 34 – 35.

Incorporated limitations of Claims 37 – 38 into Claim 36 and cancelled claims 37 – 38.

Incorporated limitations of Claims 40 – 42 into Claim 39 and cancelled claims 40 – 42.

Incorporated limitations of Claims 44 – 46 into Claim 43 and cancelled claims 44 – 46.

Reissue Application No. 16/202,067   Attorney Docket No. V-0070US
Reissue Application Filed: 11/27/18

New claims 49 and 50 are added both narrowing the original Claim 1 in different ways.

Therefore, the claims 1, 2, 4, 7, 11, 16, 22, 24, 28, 33, 36, 39, 43, 49, 50 are pending in this reissue application following entry of this amendment.

Remarks are set forth on page 25.

5

Reissue Application No. 16/202,067   Attorney Docket No. V-0070US
Reissue Application Filed: 11/27/18

Amendments to the Specification

Amendments to the specification are indicated below wherein matter to added is indicated by underlining and matter to be deleted is indicated in [brackets].

5

Please replace the paragraph at column 1, lines 7-15, following the heading "CROSS REFERENCE TO RELATED APPLICATIONS" with the following:

This application [is related to and] is an application for reissue of U.S. Patent No.

10   8,269,523 which claims priority of the PCT Application Serial No. PCT/US08/64605 entitled "VLSI LAYOUTS OF FULLY CONNECTED GENERALIZED NETWORKS" by Venkat Konda assigned to the same assignee as the current application, filed May 22, 2008, [and] which in turn claims priority of the U.S. Provisional Patent Application Serial No. 60/940, 394 entitled "VLSI LAYOUTS OF FULLY CONNECTED

15   GENERALIZED NETWORKS" by Venkat Konda assigned to the same assignee as the current application, filed May 25, 2007.

Reissue Application No. 16/202,067   Attorney Docket No. V-0070US
Reissue Application Filed: 11/27/18

LISTING OF THE CLAIMS

This Claim listing sets forth all the claims that will be pending in this reissue application following entry of the present amendment.

5   1.   (Amended): An integrated circuit [device] comprising a plurality of sub-integrated circuit blocks and a [routing network] multi-stage network, and

[Said] each sub-integrated circuit block of said plurality of sub-integrated circuit blocks comprising a plurality of inlet links and a plurality of outlet links; and

[Said] said [routing network] multi-stage network comprising [of] a plurality of

10   stages y [in] corresponding to each [said] sub-integrated circuit block of said plurality of sub-integrated circuit blocks, starting from [the lowest] stage [of] 1 to [the highest] stage [of] y, where $y \geq 1$; and

[Said routing network comprising] each stage of said plurality of stages y comprising a plurality of switches of size $d \times d$, where $d \geq 2$ [, in each said stage] and each [said] switch of said plurality of switches of size $d \times d$ having d inlet

15   links and d outlet links; and

[Said] said plurality of outlet links of [said] each sub-integrated circuit block of said plurality of sub-integrated circuit blocks are directly connected to said plurality of inlet links of said plurality of switches of size $d \times d$ of its corresponding said [lowest]

20   stage [of] 1 of said plurality of stages y, and said plurality of inlet links of [said] each sub-integrated circuit block of said plurality of sub-integrated circuit blocks are directly connected from said plurality of outlet links of said plurality of switches of size $d \times d$ of [its corresponding] said [lowest] stage [of] 1 of said plurality of stages y; and

[said] each sub-integrated circuit block of said plurality of sub-integrated circuit

25   blocks comprising a plurality of forward connecting links connecting from said plurality of switches in [a lower] each stage of said plurality of stages y to said plurality of

switches in [its] an immediate succeeding [higher] stage of said plurality of stages y, and

also comprising a plurality of backward connecting links connecting from said plurality

of switches in [a higher] each stage of said plurality of stages y to said plurality of

switches in [its] an immediate [succeeding lower] preceding stage of said plurality of

5   stages y; and

        [Said each sub-integrated circuit block comprising a plurality straight links in said

forward connecting links from switches in said each lower stage to switches in its

immediate succeeding higher stage and a plurality cross links in said forward connecting

links from switches in said each lower stage to switches in its immediate succeeding

10   higher stage, and further comprising a plurality of straight links in said backward

connecting links from switches in said each higher stage to switches in its immediate

preceding lower stage and a plurality of cross links in said backward connecting links

from switches in said each higher stage to switches in its immediate preceding lower

stage,]

15   said plurality of sub-integrated circuit blocks arranged in a two-dimensional grid

of a plurality of rows and a plurality of columns, and

        said plurality of forward connecting links and said plurality of backward

connecting links comprise[s] a plurality of [said all] straight middle links [are] connecting

from said plurality of switches of a first stage of said plurality of stages y in [each said] a

20   first sub-integrated circuit block of said plurality of sub-integrated circuit blocks [are

connecting] to said plurality of switches of an immediate succeeding or an immediate

preceding stage of said first stage of said plurality of stages y in [the same] said first sub-

integrated circuit block of said plurality of sub-integrated circuit blocks; and

        said plurality of forward connecting links and said plurality of backward

25   connecting links comprise[s] a plurality of [said all] cross middle links [are] connecting

[as either vertical or horizontal links between switches in two different said sub-

integrated circuit blocks which are either placed vertically above or below, or placed

horizontally to the left or to the right,] from switches of said plurality of switches in a

stage of said plurality of stages *y* in a first sub-integrated circuit block of said plurality of

sub-integrated circuit blocks to switches of said plurality of switches in a succeeding

5   stage or a preceding stage a first sub-integrated circuit block of said plurality of sub-

integrated circuit blocks wherein said plurality of cross middle links are either vertical

tracks or horizontal tracks, and

        a plurality of said plurality of cross middle links in succeeding stages of said

plurality of stages *y* are connected alternately as vertical tracks and horizontal tracks

10   (hereinafter "hypercube topology"), and

        [each said plurality of sub-integrated circuit blocks comprising same number of

said stages and] each sub-integrated circuit block of said plurality of sub-integrated

circuit blocks connected with said plurality of switches of size $d \times d$ of said plurality of

stages *y* [said switches in each said stage,] regardless of the size of said plurality of rows

15   and size of said plurality of columns of said two-dimensional grid [so that] wherein each

sub-integrated circuit block of said plurality of sub-integrated circuit [block with its

corresponding said stages and said switches in each stage] blocks connected with said

plurality of switches of size $d \times d$ of said plurality of stages *y* is replicable in both

vertical direction or horizontal direction of said two-dimensional grid.

20   2.   (Amended): The integrated circuit [device] of claim 1,

said two-dimensional grid of said plurality of sub-integrated circuit blocks connected with

[their corresponding] said plurality of stages *y* [and said switches in each stage] is

scalable by any power of 2, and,

        for each multiplication of 2 of [the] size of total said plurality of sub-integrated

25   circuit blocks, by adding one more stage [of switches] to said plurality of stages and said

hypercube [layout] topology is extended and placed in said hypercube topology [format]

and [also] the cross middle links between said one more stage to said plurality of stages *y*

Reissue Application No. 16/202,067   Attorney Docket No. V-0070US
Reissue Application Filed: 11/27/18

of switches of said plurality of switches are connected in said hypercube topology [format].

3.   (Cancel claim 3)

4.   (Amended): The integrated circuit [device] of claim [3] 1, wherein [said cross links from switches in a stage said stages in one of said sub-integrated circuit blocks are connecting to switches in the succeeding stage in another of said sub-integrated circuit blocks so that said cross links are either vertical links or horizontal and vice versa, and hereinafter such cross links are "shuffle exchange links").] said plurality of cross middle links between switches in any two same succeeding stages of said plurality of stages y are substantially of equal length in said integrated circuit, and

        the shortest cross middle links of said plurality of cross middle links are connected at stage 1 of said plurality of stages y and between switches of said plurality of switches in two nearest neighboring said plurality of sub-integrated circuit blocks, and length of the cross middle links is doubled in each succeeding stage of said plurality of stages y in either said vertical tracks or said horizontal tracks.

5.   (Cancel claim 5)

6.   (Cancel claim 6)

7.   (Amended): The integrated circuit [device] of claim [6] 4, wherein $y \geq (\log_2 N)$, where $N > 1$, [so that] wherein [the] length or width of the [horizontal] plurality of cross middle links [shuffle exchange links] in [the highest] stage y is equal to half the [size] length or width [of the horizontal size] of said two dimensional grid of said plurality of sub-integrated circuit blocks [and the length of the vertical shuffle exchange links in the highest stage is equal to half the size of the vertical size of said two dimensional grid of sub-integrated circuit blocks,], and wherein $d = 2$ and either

each stage of said plurality of stages in each sub-integrated circuit block of said plurality of sub-integrated circuit blocks comprises only one switch of said plurality of switches connecting said plurality of forward connecting links and each stage of said plurality of stages in each sub-integrated circuit block of said plurality of sub-integrated

5   circuit blocks comprises only one switch of said plurality of switches connecting said plurality of backward connecting links and said multi-stage network is rearrangeably nonblocking for unicast Benes network, or

each stage of said plurality of stages in each sub-integrated circuit block of said plurality of sub-integrated circuit blocks comprises at least two switches of said plurality

10   of switches connecting said plurality of forward connecting links and each stage of said plurality of stages in each sub-integrated circuit block of said plurality of sub-integrated circuit blocks comprises at least two switches of said plurality of switches connecting said plurality of backward connecting links and said multi-stage network is strictly nonblocking for unicast Benes network and is rearrangeably nonblocking for arbitrary

15   fan-out multicast Benes network, or

each stage of said plurality of stages in each sub-integrated circuit block of said plurality of sub-integrated circuit blocks comprises at least three switches of said plurality of switches connecting said plurality of forward connecting links and each stage of said plurality of stages in each sub-integrated circuit block of said plurality of sub-integrated

20   circuit blocks comprises at least three switches of said plurality of switches connecting said plurality of backward connecting links and said multi-stage network is strictly nonblocking for arbitrary fan-out multicast Benes network.

8.   (Cancel claim 8)

25   9.   (Cancel claim 9)

10.   (Cancel claim 10)

11.   (Amended): The integrated circuit [device] of claim [6] 4, wherein $y \geq \left(\log_2 N\right)$,

where $N > 1$, [so that] wherein [the] length or width of the [horizontal] plurality of cross

middle links [shuffle exchange links] in [the highest] stage y is equal to half the [size]

length or width [of the horizontal size] of said two dimensional grid of said plurality of

5   sub-integrated circuit blocks [and the length of the vertical shuffle exchange links in the

highest stage is equal to half the size of the vertical size of said two dimensional grid of

sub-integrated circuit blocks,], and

        [said] each sub-integrated circuit block of said plurality of sub-integrated circuit

blocks further comprising a plurality of U-turn links within switches of said plurality of

10   switches in each stage of said plurality of stages in each of said plurality of sub-integrated

circuit blocks[.], and wherein $d = 2$ and either

        each stage of said plurality of stages in each sub-integrated circuit block of said

plurality of sub-integrated circuit blocks comprises only one switch of said plurality of

switches connecting said plurality of forward connecting links and each stage of said

15   plurality of stages in each sub-integrated circuit block of said plurality of sub-integrated

circuit blocks comprises only one switch of said plurality of switches connecting said

plurality of backward connecting links and said multi-stage network is rearrangeably

nonblocking for unicast butterfly fat tree network, or

        each stage of said plurality of stages in each sub-integrated circuit block of said

20   plurality of sub-integrated circuit blocks comprises at least two switches of said plurality

of switches connecting said plurality of forward connecting links and each stage of said

plurality of stages in each sub-integrated circuit block of said plurality of sub-integrated

circuit blocks comprises at least two switches of said plurality of switches connecting said

plurality of backward connecting links and said multi-stage network is strictly

25   nonblocking for unicast butterfly fat tree network and rearrangeably nonblocking for

arbitrary fan-out multicast butterfly fat tree network, or

        each stage of said plurality of stages in each sub-integrated circuit block of said

plurality of sub-integrated circuit blocks comprises at least three switches of said plurality

of switches connecting said plurality of forward connecting links and each stage of said plurality of stages in each sub-integrated circuit block of said plurality of sub-integrated circuit blocks comprises at least three switches of said plurality of switches connecting said plurality of backward connecting links and said multi-stage network is strictly

5   nonblocking for arbitrary fan-out multicast butterfly fat tree network.

12.   (Cancel claim 12)

13.   (Cancel claim 13)

14.   (Cancel claim 14)

15.   (Cancel claim 15)

10   16.   (Amended): The integrated circuit [device] of claim 1, wherein said plurality of switches comprising [active and] a plurality of reprogrammable cross points and said plurality of [each] reprogrammable cross point is programmable by SRAM cells or Flash Cells[.], or

said integrated circuit is a a field programmable gate array (FPGA) or a

15   programmable logic device (PLD) or an anti-fuse based programmable integrated circuit device or a mask programmable structured ASIC device, or an Application Specific Integrated Circuit (ASIC) embedded with programmable logic circuit or 3D-FPGA, or

said plurality of forward connecting links use a plurality of buffers to amplify signals driven through them and said plurality of backward connecting links use a

20   plurality of buffers to amplify signals driven through them; and said plurality of buffers are either inverting or non-inverting buffers.

17.   (Cancel claim 17)

Reissue Application No. 16/202,067   Attorney Docket No. V-0070US
Reissue Application Filed: 11/27/18

18.   (Cancel claim 18)

19.   (Cancel claim 19)

20.   (Cancel claim 20)

21.   (Cancel claim 21)

5   22.   (Amended) The integrated circuit [device] of claim 1, wherein said <u>plurality of</u> switches comprising passive cross points or just connection of two links or not and said integrated circuit [device] is [a] <u>an</u> Application Specific Integrated Circuit (ASIC) device.

23.   (Cancel claim 23)

24.   (Amended): The integrated circuit [device] of claim [4] <u>1</u>, wherein said [all
10   horizontal] <u>plurality of cross middle links</u> [shuffle exchange links] between switches <u>of</u> <u>said plurality of switches</u> in any two corresponding [said] succeeding stages <u>of said</u> <u>plurality of stages</u> are of different length and said [vertical] <u>plurality of cross middle links</u> [shuffle exchange links] between switches <u>of said plurality of switches</u> in any two corresponding [said] succeeding stages <u>of said plurality of stages</u> are of different [length]
15   <u>width</u> and $y \geq (\log_2 N)$, where $N > 1$ [.]<u>, and wherein</u> $d = 2$ <u>and either</u>
        <u>each stage of said plurality of stages in each sub-integrated circuit block of said</u>
<u>plurality of sub-integrated circuit blocks comprises only one switch of said plurality of</u>
<u>switches connecting said plurality of forward connecting links and each stage of said</u>
<u>plurality of stages in each sub-integrated circuit block of said plurality of sub-integrated</u>
20   <u>circuit blocks comprises only one switch of said plurality of switches connecting said</u>
<u>plurality of backward connecting links and said multi-stage network is rearrangeably</u>
<u>nonblocking for unicast generalized multi-stage network, or</u>

        <u>each stage of said plurality of stages in each sub-integrated circuit block of said</u>
<u>plurality of sub-integrated circuit blocks comprises at least two switches of said plurality</u>
25   <u>of switches connecting said plurality of forward connecting links and each stage of said</u>

plurality of stages in each sub-integrated circuit block of said plurality of sub-integrated circuit blocks comprises at least two switches of said plurality of switches connecting said plurality of backward connecting links and said multi-stage network is strictly nonblocking for unicast generalized multi-stage network and rearrangeably nonblocking

5   for arbitrary fan-out multicast generalized multi-stage network, or

each stage of said plurality of stages in each sub-integrated circuit block of said plurality of sub-integrated circuit blocks comprises at least three switches of said plurality of switches connecting said plurality of forward connecting links and each stage of said plurality of stages in each sub-integrated circuit block of said plurality of sub-integrated

10   circuit blocks comprises at least three switches of said plurality of switches connecting said plurality of backward connecting links and said multi-stage network is strictly nonblocking for arbitrary fan-out multicast generalized multi-stage network.

25.   (Cancel claim 25)

26.   (Cancel claim 26)

15   27.   (Cancel claim 27)

28.   (Amended): The integrated circuit [device] of claim [4] 1, wherein said [all horizontal] plurality of cross middle links [shuffle exchange links] between switches of said plurality of switches in any two corresponding [said] succeeding stages of said plurality of stages are of different length and said [vertical] plurality of cross middle links

20   [shuffle exchange links] between switches of said plurality of switches in any two corresponding [said] succeeding stages of said plurality of stages are of different [length] width and $y \geq \left(\log_2 N\right)$, where $N > 1$ [.], and

[said] each sub-integrated circuit block of said plurality of sub-integrated circuit blocks further comprising a plurality of U-turn links within switches of said plurality of

25   switches in each of said stages of said plurality of stages in each of said plurality of sub-integrated circuit blocks[.], and

Reissue Application No. 16/202,067   Attorney Docket No. V-0070US
Reissue Application Filed: 11/27/18

wherein $d = 2$ and either

each stage of said plurality of stages in each sub-integrated circuit block of said plurality of sub-integrated circuit blocks comprises only one switch of said plurality of switches connecting said plurality of forward connecting links and each stage of said

5   plurality of stages in each sub-integrated circuit block of said plurality of sub-integrated circuit blocks comprises only one switch of said plurality of switches connecting said plurality of backward connecting links and said multi-stage network is rearrangeably nonblocking for unicast generalized butterfly fat tree network, or

each stage of said plurality of stages in each sub-integrated circuit block of said

10   plurality of sub-integrated circuit blocks comprises at least two switches of said plurality of switches connecting said plurality of forward connecting links and each stage of said plurality of stages in each sub-integrated circuit block of said plurality of sub-integrated circuit blocks comprises at least two switches of said plurality of switches connecting said plurality of backward connecting links and said multi-stage network is strictly

15   nonblocking for unicast generalized butterfly fat tree Network and rearrangeably nonblocking for arbitrary fan-out multicast generalized butterfly fat tree network, or

each stage of said plurality of stages in each sub-integrated circuit block of said plurality of sub-integrated circuit blocks comprises at least three switches of said plurality of switches connecting said plurality of forward connecting links and each stage of said

20   plurality of stages in each sub-integrated circuit block of said plurality of sub-integrated circuit blocks comprises at least three switches of said plurality of switches connecting said plurality of backward connecting links and said multi-stage network is strictly nonblocking for arbitrary fan-out multicast generalized butterfly fat tree network.

        29.   (Cancel claim 29)

25   30.   (Cancel claim 30)

        31.   (Cancel claim 31)

32.   (Cancel claim 32)

33.   (Amended): The integrated circuit [device] of claim [7] 4, wherein $y \geq \left(\log_2 N\right)$, where $N > 1$, wherein [the] length or width of said plurality of cross middle links in stage y is equal to half the size of said two dimensional grid of said plurality of sub-integrated

5   circuit blocks, and wherein $d = 4$ and either

       [there is only one switch in each said stage in each said sub-integrated circuit
block connecting said forward connecting links and there is only one switch in each said
stage in each said sub-integrated circuit block connecting said backward connecting links]
each stage of said plurality of stages in each sub-integrated circuit block of said plurality

10   of sub-integrated circuit blocks comprises only one switch of said plurality of switches
connecting said plurality of forward connecting links and each stage of said plurality of
stages in each sub-integrated circuit block of said plurality of sub-integrated circuit
blocks comprises only one switch of said plurality of switches connecting said plurality
of backward connecting links and said [routing network] multi-stage network is

15   rearrangeably nonblocking for unicast multi-link Benes network [with full bandwidth.], or

       each stage of said plurality of stages in each sub-integrated circuit block of said
plurality of sub-integrated circuit blocks comprises at least two switches of said plurality
of switches connecting said plurality of forward connecting links and each stage of said

20   plurality of stages in each sub-integrated circuit block of said plurality of sub-integrated
circuit blocks comprises at least two switches of said plurality of switches connecting said
plurality of backward connecting links and said multi-stage network is strictly
nonblocking for unicast multi-link Benes network and rearrangeably nonblocking for
arbitrary fan-out multicast multi-link Benes network, or

25   each stage of said plurality of stages in each sub-integrated circuit block of said
plurality of sub-integrated circuit blocks comprises at least three switches of said plurality

of switches connecting said plurality of forward connecting links and each stage of said

plurality of stages in each sub-integrated circuit block of said plurality of sub-integrated

circuit blocks comprises at least three switches of said plurality of switches connecting

said plurality of backward connecting links and said multi-stage network is strictly

5   nonblocking for arbitrary fan-out multicast multi-link Benes network.

34.   (Cancel claim 34)

35.   (Cancel claim 35)

36.   (Amended): The integrated circuit [device] of claim [11] 4, wherein

$y \geq \left(\log_2 N\right)$, where $N > 1$, wherein [the] length or width of said plurality of cross middle

10   links in stage y is equal to half the size of said two dimensional grid of said plurality of

sub-integrated circuit blocks, and

        each sub-integrated circuit block of said plurality of sub-integrated circuit blocks

further comprising a plurality of U-turn links within switches of said plurality of switches

in each stage of said plurality of stages in each of said plurality of sub-integrated circuit

15   blocks[.], and wherein $d = 4$ and either

[there is only one switch in each said stage in each said sub-integrated circuit

block connecting said forward connecting links and there is only one switch in each said

stage in each said sub-integrated circuit block connecting said backward connecting links]

each stage of said plurality of stages in each sub-integrated circuit block of said plurality

20   of sub-integrated circuit blocks comprises only one switch of said plurality of switches

connecting said plurality of forward connecting links and each stage of said plurality of

stages in each sub-integrated circuit block of said plurality of sub-integrated circuit

blocks comprises only one switch of said plurality of switches connecting said plurality

of backward connecting links and said [routing network] multi-stage network is

25   rearrangeably nonblocking for unicast multi-link butterfly fat tree network [with full

bandwidth.], or

each stage of said plurality of stages in each sub-integrated circuit block of said
plurality of sub-integrated circuit blocks comprises at least two switches of said plurality
of switches connecting said plurality of forward connecting links and each stage of said
plurality of stages in each sub-integrated circuit block of said plurality of sub-integrated
5   circuit blocks comprises at least two switches of said plurality of switches connecting said
plurality of backward connecting links and said multi-stage network is strictly
nonblocking for unicast multi-link butterfly fat tree network and rearrangeably
nonblocking for arbitrary fan-out multicast multi-link butterfly fat tree network, or

each stage of said plurality of stages in each sub-integrated circuit block of said
10   plurality of sub-integrated circuit blocks comprises at least three switches of said plurality
of switches connecting said plurality of forward connecting links and each stage of said
plurality of stages in each sub-integrated circuit block of said plurality of sub-integrated
circuit blocks comprises at least three switches of said plurality of switches connecting
said plurality of backward connecting links and said multi-stage network is strictly
15   nonblocking for arbitrary fan-out multicast multi-link butterfly fat tree network.

37.   (Cancel claim 37)

38.   (Cancel claim 38)

39.   (Amended): The integrated circuit [device] of claim 4, wherein said [all
horizontal] plurality of cross middle links [shuffle exchange links] between switches of
20   said plurality of switches in any two corresponding said succeeding stages of said
plurality of stages are of different length and said [vertical] plurality of cross middle links
[shuffle exchange links] between switches of said plurality of switches in any two
corresponding said succeeding stages of said plurality of stages are of different [length]
width and $y \geq (\log_2 N)$, where $N > 1$ [.], and wherein $d = 4$ and either

25   each stage of said plurality of stages in each sub-integrated circuit block of said
plurality of sub-integrated circuit blocks comprises only one switch of said plurality of

switches connecting said plurality of forward connecting links and each stage of said

plurality of stages in each sub-integrated circuit block of said plurality of sub-integrated

circuit blocks comprises only one switch of said plurality of switches connecting said

plurality of backward connecting links and said multi-stage network is rearrangeably

5   nonblocking for unicast generalized multi-link multi-stage network, or

        each stage of said plurality of stages in each sub-integrated circuit block of said

plurality of sub-integrated circuit blocks comprises at least two switches of said plurality

of switches connecting said plurality of forward connecting links and each stage of said

plurality of stages in each sub-integrated circuit block of said plurality of sub-integrated

10   circuit blocks comprises at least two switches of said plurality of switches connecting said

plurality of backward connecting links and said multi-stage network is strictly

nonblocking for unicast generalized multi-link multi-stage network and rearrangeably

nonblocking for arbitrary fan-out multicast generalized multi-link multi-stage network, or

        each stage of said plurality of stages in each sub-integrated circuit block of said

15   plurality of sub-integrated circuit blocks comprises at least three switches of said plurality

of switches connecting said plurality of forward connecting links and each stage of said

plurality of stages in each sub-integrated circuit block of said plurality of sub-integrated

circuit blocks comprises at least three switches of said plurality of switches connecting

said plurality of backward connecting links and said multi-stage network is strictly

20   nonblocking for arbitrary fan-out multicast generalized multi-link multi-stage network.

        40.   (Cancel claim 40)

        41.   (Cancel claim 41)

        42.   (Cancel claim 42)

        43.   (Amended): The integrated circuit [device] of claim 4, wherein said [all

25   horizontal] plurality of cross middle links [shuffle exchange links] between switches of

said plurality of switches in any two corresponding said succeeding stages of said

plurality of stages are of different length and said [vertical] plurality of cross middle links [shuffle exchange links] between switches of said plurality of switches in any two corresponding said succeeding stages of said plurality of stages are of different [length] width and $y \geq (\log_2 N)$, where $N > 1$ [.], and

5   [said] each sub-integrated circuit block of said plurality of sub-integrated circuit blocks further comprising a plurality of U-turn links within switches of said plurality of switches in each of said stages of said plurality of stages in each of said plurality of sub-integrated circuit blocks[.], and wherein $d = 4$ and either

          each stage of said plurality of stages in each sub-integrated circuit block of said

10   plurality of sub-integrated circuit blocks comprises only one switch of said plurality of switches connecting said plurality of forward connecting links and each stage of said plurality of stages in each sub-integrated circuit block of said plurality of sub-integrated circuit blocks comprises only one switch of said plurality of switches connecting said plurality of backward connecting links and said multi-stage network is rearrangeably

15   nonblocking for unicast generalized multi-link butterfly fat tree network, or

          each stage of said plurality of stages in each sub-integrated circuit block of said plurality of sub-integrated circuit blocks comprises at least two switches of said plurality of switches connecting said plurality of forward connecting links and each stage of said plurality of stages in each sub-integrated circuit block of said plurality of sub-integrated

20   circuit blocks comprises at least two switches of said plurality of switches connecting said plurality of backward connecting links and said multi-stage network is strictly nonblocking for unicast generalized multi-link butterfly fat tree Network and rearrangeably nonblocking for arbitrary fan-out multicast generalized multi-link butterfly fat tree network, or

25   each stage of said plurality of stages in each sub-integrated circuit block of said plurality of sub-integrated circuit blocks comprises at least three switches of said plurality of switches connecting said plurality of forward connecting links and each stage of said plurality of stages in each sub-integrated circuit block of said plurality of sub-integrated circuit blocks comprises at least three switches of said plurality of switches connecting

said plurality of backward connecting links and said multi-stage network is strictly nonblocking for arbitrary fan-out multicast generalized multi-link butterfly fat tree network.

44.   (Cancel claim 44)

5   45.   (Cancel claim 45)

46.   (Cancel claim 46)

47.   (Cancel claim 47)

48.   (Cancel claim 48)

10   49. (New):   An integrated circuit comprising a plurality of sub-integrated circuit blocks and a multi-stage network, and

each sub-integrated circuit block of said plurality of sub-integrated circuit blocks comprising a plurality of inlet links and a plurality of outlet links; and

said multi-stage network comprising a plurality of stages y corresponding to each

15   sub-integrated circuit block of said plurality of sub-integrated circuit blocks, starting from stage 1 to stage y, where $y \geq 1$; and

each stage of said plurality of stages y comprising a plurality of switches of size $d \times d$, where $d \geq 2$ and said plurality of switches comprising at least one switch of size $d \geq 3$ and each switch of said plurality of switches of size $d \times d$ having d inlet links and

20   d outlet links; and

said plurality of outlet links of each sub-integrated circuit block of said plurality of sub-integrated circuit blocks are directly connected to said plurality of inlet links of said

plurality of switches of size $d \times d$ of its corresponding said stage 1 of said plurality of

stages y, and said plurality of inlet links of each sub-integrated circuit block of said

plurality of sub-integrated circuit blocks are directly connected from said plurality of

outlet links of said plurality of switches of size $d \times d$ of said stage 1 of said plurality of

5   stages y ; and

each sub-integrated circuit block of said plurality of sub-integrated circuit blocks

comprising a plurality of forward connecting links connecting from said plurality of

switches in each stage of said plurality of stages y to said plurality of switches in an

immediate succeeding stage of said plurality of stages y, and also comprising a plurality

10   of backward connecting links connecting from said plurality of switches in each stage of

said plurality of stages y to said plurality of switches in an immediate preceding stage of

said plurality of stages y ; and

said plurality of sub-integrated circuit blocks arranged in a two-dimensional grid

of a plurality of rows and a plurality of columns, and

15   said plurality of forward connecting links and said plurality of backward

connecting links comprise a plurality of straight middle links connecting from said

plurality of switches of a first stage of said plurality of stages y in a first sub-integrated

circuit block of said plurality of sub-integrated circuit blocks to said plurality of switches

of an immediate succeeding or an immediate preceding stage of said first stage of said

20   plurality of stages y in said first sub-integrated circuit block of said plurality of sub-

integrated circuit blocks; and

said plurality of forward connecting links and said plurality of backward

connecting links comprise a plurality of cross middle links connecting from switches of

said plurality of switches in a stage of said plurality of stages y in a first sub-integrated

25   circuit block of said plurality of sub-integrated circuit blocks to switches of said plurality

of switches in a succeeding stage or a preceding stage a first sub-integrated circuit block of said plurality of sub-integrated circuit blocks wherein said plurality of cross middle links are either vertical tracks or horizontal tracks, and

a plurality of said plurality of cross middle links in succeeding stages of said

5   plurality of stages y are connected alternately as vertical tracks and horizontal tracks (hereinafter "hypercube topology"), and

each sub-integrated circuit block of said plurality of sub-integrated circuit blocks connected with said plurality of switches of size $d \times d$ of said plurality of stages y regardless of the size of said plurality of rows and size of said plurality of columns of said

10   two-dimensional grid wherein each sub-integrated circuit block of said plurality of sub-integrated circuit blocks connected with said plurality of switches of size $d \times d$ of said plurality of stages y is replicable in both vertical direction or horizontal direction of said two-dimensional grid.

50.   (New): An integrated circuit comprising a plurality of sub-integrated circuit

15   blocks and a multi-stage network, and

each sub-integrated circuit block of said plurality of sub-integrated circuit blocks comprising a plurality of inlet links and a plurality of outlet links; and

said multi-stage network comprising a plurality of stages y corresponding to each sub-integrated circuit block of said plurality of sub-integrated circuit blocks, starting from

20   stage 1 to stage y, where $y \geq 1$; and

each stage of said plurality of stages y comprising a plurality of switches of size $d \times d$, where $d \geq 2$ and each switch of said plurality of switches of size $d \times d$ having d inlet links and d outlet links; and

said plurality of outlet links of each sub-integrated circuit block of said plurality of

25   sub-integrated circuit blocks are directly connected to said plurality of inlet links of said plurality of switches of size $d \times d$ of its corresponding said stage 1 of said plurality of

stages y, and said plurality of inlet links of each sub-integrated circuit block of said

plurality of sub-integrated circuit blocks are directly connected from said plurality of

outlet links of said plurality of switches of size $d \times d$ of said stage 1 of said plurality of

stages y; and

5   each sub-integrated circuit block of said plurality of sub-integrated circuit blocks

comprising a plurality of forward connecting links connecting from said plurality of

switches in each stage of said plurality of stages y to said plurality of switches in an

immediate succeeding stage of said plurality of stages y, and also comprising a plurality

of backward connecting links connecting from said plurality of switches in each stage of

10   said plurality of stages y to said plurality of switches in an immediate preceding stage of

said plurality of stages y; and

said plurality of sub-integrated circuit blocks arranged in a two-dimensional grid

of a plurality of rows and a plurality of columns, and

said plurality of forward connecting links and said plurality of backward

15   connecting links comprise a plurality of straight middle links connecting from said

plurality of switches of a first stage of said plurality of stages y in a first sub-integrated

circuit block of said plurality of sub-integrated circuit blocks to said plurality of switches

of an immediate succeeding or an immediate preceding stage of said first stage of said

plurality of stages y in said first sub-integrated circuit block of said plurality of sub-

20   integrated circuit blocks; and

said plurality of forward connecting links and said plurality of backward

connecting links comprise a plurality of cross middle links connecting from switches of

said plurality of switches in a stage of said plurality of stages y in a first sub-integrated

circuit block of said plurality of sub-integrated circuit blocks to switches of said plurality

25   of switches in a succeeding stage or a preceding stage a first sub-integrated circuit block

Reissue Application No. 16/202,067   Attorney Docket No. V-0070US
Reissue Application Filed: 11/27/18

of said plurality of sub-integrated circuit blocks wherein said plurality of cross middle links are either vertical tracks or horizontal tracks, and

a plurality of said plurality of cross middle links in succeeding stages of said plurality of stages y are connected alternately as one or more vertical tracks and one or

5   more horizontal tracks (hereinafter "hypercube topology"), and

each sub-integrated circuit block of said plurality of sub-integrated circuit blocks connected with said plurality of switches of size $d \times d$ of said plurality of stages y regardless of the size of said plurality of rows and size of said plurality of columns of said two-dimensional grid wherein each sub-integrated circuit block of said plurality of sub-

10   integrated circuit blocks connected with said plurality of switches of size $d \times d$ of said plurality of stages y is replicable in both vertical direction or horizontal direction of said two-dimensional grid.

Reissue Application No. 16/202,067   Attorney Docket No. V-0070US
Reissue Application Filed: 11/27/18

Support for Amendments

5   The amendments presented herein are fully supported by the Specification as **originally**
filed in US Patent No. 8,269,523, entitled "VLSI LAYOUTS OF FULLY CONNECTED
GENERALIZED NETWORKS," issued on September 18, 2012 , and thus do not add
new matter. More particularly, support for the amendment to claim 49, 50 is found at
least at FIGs 1A – 1J, 1K, 1K1, 1L, 1L1 and col, 8:42 - 25:2.

REMARKS

10   The present amendment is made to cancel some claims issued U.S. Patent No. 8,269,523
which claims priority of the PCT Application Serial No. PCT/US08/64605 filed May
22, 2008, which in turn claims priority of the U.S. Provisional Patent Application Serial
No. 60/940, 394 filed May 25, 2007.

Originally issued claims 1, 2, 4, 7, 11, 16, 22, 24, 28, 33, 36, 39, 43 are amended.

15   Originally issued claims 3, 5, 6, 8 - 10, 12 - 14, 15, 17 – 21, 23, 25 – 27, 29 – 32, 34 – 35,
37 – 38, 40 – 42, 44 - 48 are now cancelled. New claims 49 and 50 are added. Therefore,
the claims 1, 2, 4, 7, 11, 16, 22, 24, 28, 33, 36, 39, 43, 49, 50 are pending in this reissue
application following entry of this amendment.

Should the Office have any questions concerning this communication, please contact the

20   applicant Venkat Konda at (408) 472-3273.

Very respectfully,

/Venkat Konda/

Venkat Konda

25   Konda Technologies, Inc. (USPTO Customer Number: 38139)

6278 Grand Oak Way

Reissue Application No. 16/202,067   Attorney Docket No. V-0070US
Reissue Application Filed: 11/27/18

San Jose, CA 95135

Phone: 408-472-3273

Fax: 408-238-2478

PATENT APPLICATION FEE DETERMINATION RECORD Substitute for Form PTO-875	Application or Docket Number 16/202,067	Filing Date 11/27/2018	☐ To be Mailed

ENTITY: ☐ LARGE ☑ SMALL ☐ MICRO

APPLICATION AS FILED - PART I

FOR	(Column 1) NUMBER FILED	(Column 2) NUMBER EXTRA		RATE ($)	FEE ($)
☐ BASIC FEE (37 CFR 1.16(a), (b), or (c))	N/A	N/A		N/A	
☐ SEARCH FEE (37 CFR 1.16(k), (i), or (m))	N/A	N/A		N/A	
☐ EXAMINATION FEE (37 CFR 1.16(o), (p), or (q))	N/A	N/A		N/A	
TOTAL CLAIMS (37 CFR 1.16(i))	minus 20 =	*		x $50 =	
INDEPENDENT CLAIMS (37 CFR 1.16(h))	minus 3 =	*		x $230 =	
☐ APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is $310 ($155 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).				
☐ MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))					
* If the difference in column 1 is less than zero, enter "0" in column 2.				TOTAL	

APPLICATION AS AMENDED - PART II

AMENDMENT		(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA		RATE ($)	ADDITIONAL FEE ($)
	Total (37 CFR 1.16(i))	*	Minus	**	=		x $0 =	
	Independent (37 CFR 1.16(h))	*	Minus	***	=		x $0 =	
	☐ Application Size Fee (37 CFR 1.16(s))							
	☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							
							TOTAL ADD'L FEE	

AMENDMENT	09/23/2019	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA		RATE ($)	ADDITIONAL FEE ($)
	Total (37 CFR 1.16(i))	* 15	Minus	** 20	= 0		x $50 =	0
	Independent (37 CFR 1.16(h))	* 3	Minus	*** 3	= 0		x $230 =	0
	☐ Application Size Fee (37 CFR 1.16(s))							
	☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							
							TOTAL ADD'L FEE	0

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.	LDRC
** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".	/EVA V GILLIS/
*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".	
The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.	

Reissue Application # 16/202,067   Attorney Docket No. V-0070US
Reissue Application Filed: 11/27/18

In The United States Patent And Trademark Office

Reissue Application   :   16/202,067   Filed: November 27, 2018

Parent US Patent No   :   8,269,523   Issued: September 18, 2012

Parent Application No: 12/601,275   Filed : May 31, 2010

5   Examiner   :   Ton, My Trang   Group Art Unit: 3992

Applicant(s): Venkat Konda

Title: VLSI Layouts of Fully Connected Generalized Networks

San Jose, 2019 September 17, Tue

10   **PETITION TO WITHDRAW THE PRIOR FILED PETITON ON 11/27/2018 TO ACCEPT THE UNINTENTIONALLY DELAYED CLAIM UNDER 35 U.S.C. 119(e)**

Attn: Richard Cole

15   Office of PCT Legal

Mail Stop PCT

Commissioner for Patents

P.O. Box 1450

Alexandria, Virginia, 22313-1450

20

Dear Sir/Madam:

Dear Sir/Madam:

For the above referenced reissue application No. 16/202,067, a petition was filed with the

25   office of PCT legal on 11/27/2018 to accept an unintentionally delayed claim under 35

U.S.C. 119(e) for the benefit of a prior-filed provisional application 37 C.F.R. 1.78(c) and

for the benefit of a prior filed international application 37 C.F.R. 1.78(e). This petition

now became moot in view of the petition granted that was filed in the parent application

Reissue Application # 16/202,067   Attorney Docket No. V-0070US
Reissue Application Filed: 11/27/18

No. 12,601,275 issued as US Patent No. 8,269,523 for the entire delay in filing the basic

national fee from the due date for the fee until the filing of a grantable petition under 37

CFR 1.137 was unintentional. Accordingly please withdraw the petition filed on

11/27/2018 and refund the $1000 fee back into the credit card.

5

Very Respectfully,

/Venkat Konda/

Venkat Konda

6278 Grand Oak Way

10   San Jose, CA 95135

Phone: 408-472-3273

Electronic Acknowledgement Receipt

EFS ID:	37192736
Application Number:	16202067
International Application Number:	
Confirmation Number:	8134
Title of Invention:	VLSI Layouts of Fully Connected Generalized Networks
First Named Inventor/Applicant Name:	Venkat Konda
Customer Number:	38139
Filer:	Venkat Konda
Filer Authorized By:	
Attorney Docket Number:	V-0070US
Receipt Date:	17-SEP-2019
Filing Date:	27-NOV-2018
Time Stamp:	17:52:01
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Petition for review by the PCT legal office	Pet2WDRissue.pdf	79183 c3c8a6e1f6302f269746ab20cff526a76eee1 fac	no	2

Warnings:

Information:	
Total Files Size (in bytes):	79183

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

<u>New Applications Under 35 U.S.C. 111</u>
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.
<u>National Stage of an International Application under 35 U.S.C. 371</u>
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.
<u>New International Application Filed with the USPTO as a Receiving Office</u>
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

To: venkat@kondatech.com,vkonda@gmail.com,
From: PAIR_eOfficeAction@uspto.gov
Cc: PAIR_eOfficeAction@uspto.gov
Subject: Private PAIR Correspondence Notification for Customer Number 38139

Sep 10, 2019 03:40:23 AM

Dear PAIR Customer:

Konda Technologies, Inc
6278 GRAND OAK WAY
SAN JOSE, CA 95135
UNITED STATES

The following USPTO patent application(s) associated with your Customer Number, 38139 , have
new outgoing correspondence. This correspondence is now available for viewing in Private PAIR.

The official date of notification of the outgoing correspondence will be indicated on the form PTOL-90
accompanying the correspondence.

Disclaimer:
The list of documents shown below is provided as a courtesy and is not part of the official file
wrapper. The content of the images shown in PAIR is the official record.

Application	Document	Mailroom Date	Attorney Docket No.
16202067	M327	09/09/2019	V-0070US

To view your correspondence online or update your email addresses, please visit us anytime at
https://sportal.uspto.gov/secure/myportal/privatepair.

If you have any questions, please email the Electronic Business Center (EBC) at EBC@uspto.gov
with 'e-Office Action' on the subject line or call 1-866-217-9197 during the following hours:

    Monday - Friday 6:00 a.m. to 12:00 a.m.

Thank you for prompt attention to this notice,

UNITED STATES PATENT AND TRADEMARK OFFICE

PATENT APPLICATION INFORMATION RETRIEVAL SYSTEM

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

Konda Technologies, Inc
6278 GRAND OAK WAY
SAN JOSE CA 95135

MAILED

SEP 0 9 2019

INTERNATIONAL PATENT LEGAL ADM.

In re Application of   :
Konda Technologies Inc.   :
Application No.: 16/202,067   :   NOTIFICATION
Filing Date: 27 November 2018   :
Attorney Docket No.: V-0070US   :

This Notification is in response to the filing of "In Response to the Notification Dated June 11, 2019 From International Patent Legal Administration," in the United States Patent and Trademark Office on 04 July 2019.

The 27 November 2018 petition was not properly signed under 37 CFR 1.33(b)(3) at the time it was presented. Later changes to applicant do not cure the petition signature. See 37 CFR 3.73(c)(1). If applicant wants the Office to address the petition, it must be refiled.

Any further correspondence with respect to this matter may be filed electronically via EFS-Web selecting the document description "Petition for review and processing by the PCT Legal Office" or by mail addressed to Mail Stop PCT, Commissioner for Patents, Office of PCT Legal Administration, P.O. Box 1450, Alexandria, Virginia 22313-1450, with the contents of the letter marked to the attention of the International Patent Legal Administration.

/Erin P. Thomson/
Erin P. Thomson
Attorney Advisor
International Patent Legal Administration
571-272-3292

Reissue Application # 16/202,067   Attorney Docket No. V-0070US
Reissue Application Filed: 11/27/18

In The United States Patent And Trademark Office

Reissue Application   :   16, 202,067   Filed: November 27, 2018

Parent US Patent No   :   8,269,523   Issued: September 18, 2012

Examiner   :   Ton, My Trang   Group Art Unit: 3992

5   Applicant(s): Venkat Konda

Title: VLSI Layouts of Fully Connected Generalized Networks

San Jose, 2018 July 4, Thu

IN RESPONSE TO THE NOTIFICATION DATED JUNE 11, 2019 FROM
10   **INTERNATIONAL PATENT LEGAL ADMINISTRATION**

Attn: Erin P. Thomson

International Patent Legal Administration

Commissioner for Patents

15   P.O. Box 1450

Alexandria, VA 22313-1450

Dear Sir/Madam:

20   This is in response to the notification dated June 11, 2019, for the reissue application #
16/202,067, from the International Patent Legal Administration regarding the Petition
under 37 C.F.R. 1.78 filed in the United States Patent and Trademark Office on 27
November 2018 and submission of 17 April 2019. The applicant is submitting all the
required forms as follows:

25

1) On April 15, 2019, Konda Technologies, Inc. assigned this reissue application
#16/202,067, including its US Patent 8,269,523 to Venkat Konda. The recorded

Reissue Application # 16/202,067   Attorney Docket No. V-0070US
Reissue Application Filed: 11/27/18

assignment by USPTO was already filed and currently in the IFW of this application.

2) On December 20, 2012, the inventor, Venkat Konda assigned, the parent US Patent 8,269,523 of this reissue application #16/202,067, to Konda Technologies Inc. The recorded assignment by USPTO is currently filed with this submission.

3) The entire chain of title from the inventor to the current applicant, who is the inventor as well, is refiled with the form PTO/AIA/96 in this submission

4) Now since the applicant is not juristic entity and the applicant is the inventor, the Petition filed under 37 C.F.R. 1.78 in the United States Patent and Trademark Office on 27 November 2018 is now correctly signed. Accordingly applicant respectfully requests to consider the Petition.

5) A markedup ADS showing all the changes is also refiled with this submission.

Applicant now submits that the Petition filed under 37 C.F.R. 1.78 in the United States Patent and Trademark Office on 27 November 2018 is in correct form and requests International Patent Legal Administration to approve the petition.

Should the Office have any questions concerning this communication, please contact the inventor Venkat Konda at (408) 472-3273.

Very respectfully,

/Venkat Konda/

Venkat Konda

USPTO Customer Number: 38139

6278 Grand Oak Way

San Jose, CA 95135

Phone: 408-472-3273

Fax: 408-238-2478

PATENT ASSIGNMENT

Electronic Version v1.1
Stylesheet Version v1.1

SUBMISSION TYPE:	NEW ASSIGNMENT
NATURE OF CONVEYANCE:	ASSIGNMENT

CONVEYING PARTY DATA

Name	Execution Date
Venkat Konda	12/20/2012

RECEIVING PARTY DATA

Name:	Konda Technologies Inc.
Street Address:	6278 Grand Oak Way
City:	San Jose
State/Country:	CALIFORNIA
Postal Code:	95135

PROPERTY NUMBERS Total: 6

Property Type	Number
Patent Number:	8270400
Patent Number:	8170040
Patent Number:	8269523
Application Number:	12601274
Application Number:	13502207
PCT Number:	US1253814

CORRESPONDENCE DATA

Fax Number: 4082382478
Correspondence will be sent via US Mail when the fax attempt is unsuccessful.
Phone: 408-472-3273
Email: venkat@kondatech.com
Correspondent Name: Venkat Konda
Address Line 1:   6278 Grand Oak Way
Address Line 4:   San Jose, CALIFORNIA 95135

NAME OF SUBMITTER:	Venkat Konda

Total Attachments: 2
source=AssignKondaFinal12202012#page1.tif
source=AssignKondaFinal12202012#page2.tif

OP $240.00 8270400

UNITED STATES PATENT AND TRADEMARK OFFICE

UNDER SECRETARY OF COMMERCE FOR INTELLECTUAL PROPERTY AND
DIRECTOR OF THE UNITED STATES PATENT AND TRADEMARK OFFICE

DECEMBER 21, 2012

                              PTAS

VENKAT KONDA   **502172070**
6278 GRAND OAK WAY
SAN JOSE, CA 95135

                   UNITED STATES PATENT AND TRADEMARK OFFICE
                  NOTICE OF RECORDATION OF ASSIGNMENT DOCUMENT

THE ENCLOSED DOCUMENT HAS BEEN RECORDED BY THE ASSIGNMENT RECORDATION BRANCH
OF THE U.S. PATENT AND TRADEMARK OFFICE. A COMPLETE COPY IS AVAILABLE AT THE
ASSIGNMENT SEARCH ROOM ON THE REEL AND FRAME NUMBER REFERENCED BELOW.

PLEASE REVIEW ALL INFORMATION CONTAINED ON THIS NOTICE. THE INFORMATION
CONTAINED ON THIS RECORDATION NOTICE REFLECTS THE DATA PRESENT IN THE PATENT
AND TRADEMARK ASSIGNMENT SYSTEM. IF YOU SHOULD FIND ANY ERRORS OR HAVE
QUESTIONS CONCERNING THIS NOTICE, YOU MAY CONTACT THE ASSIGNMENT RECORDATION
BRANCH AT 571-272-3350. PLEASE SEND REQUEST FOR CORRECTION TO: U.S. PATENT
AND TRADEMARK OFFICE, MAIL STOP: ASSIGNMENT RECORDATION BRANCH, P.O. BOX
1450, ALEXANDRIA, VA 22313.

RECORDATION DATE: 12/20/2012   REEL/FRAME: 029513/0134
                                      NUMBER OF PAGES: 3

BRIEF: ASSIGNMENT OF ASSIGNORS INTEREST (SEE DOCUMENT FOR DETAILS).

ASSIGNOR:
  KONDA, VENKAT   DOC DATE: 12/20/2012

ASSIGNEE:
  KONDA TECHNOLOGIES INC.
  6278 GRAND OAK WAY
  SAN JOSE, CALIFORNIA 95135

APPLICATION NUMBER: 12530207   FILING DATE: 09/06/2009
PATENT NUMBER: 8270400   ISSUE DATE: 09/18/2012
TITLE: FULLY CONNECTED GENERALIZED MULTI-STAGE NETWORKS

APPLICATION NUMBER: 12601273   FILING DATE: 11/22/2009
PATENT NUMBER: 8170040   ISSUE DATE: 05/01/2012
TITLE: FULLY CONNECTED GENERALIZED BUTTERFLY FAT TREE NETWORKS

APPLICATION NUMBER: 12601274   FILING DATE: 05/31/2010
PATENT NUMBER: ISSUE DATE:
TITLE: FULLY CONNECTED GENERALIZED MULTI-LINK MULTI-STAGE NETWORKS

APPLICATION NUMBER: 12601275   FILING DATE: 05/31/2010
PATENT NUMBER: 8269523   ISSUE DATE: 09/18/2012
TITLE: VLSI LAYOUTS OF FULLY CONNECTED GENERALIZED NETWORKS

029513/0134 PAGE 2


```
APPLICATION NUMBER: 13502207          FILING DATE: 04/16/2012
PATENT NUMBER:                        ISSUE DATE:
TITLE:  VLSI LAYOUTS OF FULLY CONNECTED GENERALIZED AND PYRAMID NETWORKS
        WITH LOCALITY EXPLOITATION

APPLICATION NUMBER:                   FILING DATE:
PATENT NUMBER:                        ISSUE DATE:
PCT NUMBER: US1253814
TITLE:
```


ASSIGNMENT RECORDATION BRANCH
PUBLIC RECORDS DIVISION

Page 1

ASSIGNMENT OF PATENTS and PATENT APPLICATIONS

WHEREAS, I, Konda, Venkat of San Jose, California having a residence at 6278 Grand Oak Way, San Jose, CA, hereinafter referred to as "Assignor," am the record owner of the patents and patent applications described as

1)   Title: Fully Connected Generalized Multi-stage Networks
       US Patent No.:   8,270,400
       Patent Issue Date: September 18, 2012;

2)   Title: Fully Connected Generalized Butterfly Fat Tree Networks
       US Patent No.:   8,170,040
       Patent Issue Date: May 1, 2012;

3)   Title: Fully Connected Generalized Multi-link Multi-stage Networks
       US Application No.:   12/601,274
       Filing Date: November 22, 2009;
       Priority Date: May 25, 2007;

4)   Title: VLSI Layouts of Fully Connected Generalized Networks
       US Application No.:   8,269,523
       Patent Issue Date: September 18, 2012;

5)   Title: VLSI Layouts of Fully Connected Generalized and Pyramid
                                  Networks with Locality Exploitation
       US Application No.:   13/502,207
       Filing Date: April 16, 2012;
       Priority Date: October 16, 2009; and

6)   Title: Optimization of Multi-stage Hierarchical Networks for Practical
                                  Routing Applications
       PCT Application No.:   PCT/US12/53814
       Filing Date: September 6, 2012;
       Priority Date: September 7, 2011;

WHEREAS, Konda Technologies, Inc., a California corporation, located at 6278 Grand Oak Way, San Jose, CA 95135, hereinafter referred to as "ASSIGNEE," wishes to acquire all right, title and interest to and under the patent applications described above owned by Assignor and in and to any and all improvements to said applications and in any Letters Patent and Registrations which may be granted on the same in the United States or any country throughout the world;
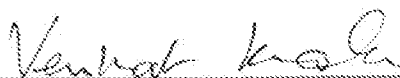
Page 2

For good and valuable consideration, receipt of which is hereby acknowledged by Assignor, Assignor, effective 20th day of December, 2012, has assigned, and by these presents does assign to Assignee all right, title and interest for the United States and all foreign countries, in and to any and all improvements for the applications described above and in and to said applications and to all utility divisional continuing, substitute, renewal, reissue, and all other patent applications which have been or shall be filed in the United States and all foreign counterparts (including patent, utility model and industrial designs), and in and to any Letters Patent and Registrations which may hereafter be granted on the same in the United States and all countries throughout the world, and to claim the priority from the application as provided by the Paris Convention. The right, title and interest is to be held and enjoyed by Assignee and Assignee's successors and assigns as fully and exclusively as it would have been held and enjoyed by Assignor had this Assignment not been made, for the full term of any Letters Patent and Registrations which may be granted thereon, or of any division, renewal, continuation in whole or in part, substitution, conversion, reissue, prolongation or extension thereof.

Assignor further agrees that they will, without charge to Assignee, but at Assignee's expense, (a) cooperate with Assignee in the prosecution of U.S. Patent applications and foreign counterparts on the applications and any improvements, (b) execute, verify, acknowledge and deliver all such further papers, including patent applications and instruments of transfer, and (c) perform such other acts as Assignee lawfully may request to obtain or maintain Letters Patent and Registrations for the applications and improvements in any and all countries, and to vest title thereto in Assignee, or Assignee's successors and assigns.

IN TESTIMONY WHEREOF, Assignor has signed on the date indicated.

Date: _December 20, 2012_ By: _____
                                              Venkat Konda

Application Data Sheet 37 CFR 1.76	**Attorney Docket Number**	V-0070US
	Application Number	

Title of Invention	VLSI Layouts of Fully Connected Generalized Networks

The application data sheet is part of the provisional or nonprovisional application for which it is being submitted. The following form contains the bibliographic data arranged in a format specified by the United States Patent and Trademark Office as outlined in 37 CFR 1.76.
This document may be completed electronically and submitted to the Office in electronic format using the Electronic Filing System (EFS) or the document may be printed and included in a paper filed application.

Secrecy Order 37 CFR 5.2:

☐ Portions or all of the application associated with this Application Data Sheet may fall under a Secrecy Order pursuant to 37 CFR 5.2 (Paper filers only. Applications that fall under Secrecy Order may not be filed electronically.)

Inventor Information:

Inventor	1		Remove

Legal Name

Prefix	Given Name	Middle Name	Family Name	Suffix
	Venkat		Konda	

Residence Information (Select One) ⦿ US Residency ◯ Non US Residency ◯ Active US Military Service

City	San Jose	State/Province	CA	Country of Residence	US

Mailing Address of Inventor:

Address 1	6278 Grand Oak Way
Address 2	

City	San Jose	State/Province	CA
Postal Code	95135	Country	US

All Inventors Must Be Listed - Additional Inventor Information blocks may be generated within this form by selecting the **Add** button.   [Add]

Correspondence Information:

Enter either Customer Number or complete the Correspondence Information section below.
For further information see 37 CFR 1.33(a).

☐ An Address is being provided for the correspondence Information of this application.

Customer Number	38139
Email Address	venkat@kondatech.com [Add Email] [Remove Email]

Application Information:

Title of the Invention	VLSI Layouts of Fully Connected Generalized Networks		
Attorney Docket Number	V-0070US	Small Entity Status Claimed	☒
Application Type	Nonprovisional		
Subject Matter	Utility		
Total Number of Drawing Sheets (if any)	39	Suggested Figure for Publication (if any)	1H

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	V-0070US
	Application Number	

Title of Invention	VLSI Layouts of Fully Connected Generalized Networks

Filing By Reference:

Only complete this section when filing an application by reference under 35 U.S.C. 111(c) and 37 CFR 1.57(a). Do not complete this section if application papers including a specification and any drawings are being filed. Any domestic benefit or foreign priority information must be provided in the appropriate section(s) below (i.e., "Domestic Benefit/National Stage Information" and "Foreign Priority Information").

For the purposes of a filing date under 37 CFR 1.53(b), the description and any drawings of the present application are replaced by this reference to the previously filed application, subject to conditions and requirements of 37 CFR 1.57(a).

Application number of the previously filed application	Filing date (YYYY-MM-DD)	Intellectual Property Authority or Country

Publication Information:

☐ Request Early Publication (Fee required at time of Request 37 CFR 1.219)

☐ **Request Not to Publish.** I hereby request that the attached application not be published under 35 U.S.C. 122(b) and certify that the invention disclosed in the attached application **has not and will not** be the subject of an application filed in another country, or under a multilateral international agreement, that requires publication at eighteen months after filing.

Representative Information:

Representative information should be provided for all practitioners having a power of attorney in the application. Providing this information in the Application Data Sheet does not constitute a power of attorney in the application (see 37 CFR 1.32). Either enter Customer Number or complete the Representative Name section below. If both sections are completed the customer Number will be used for the Representative Information during processing.

Please Select One:	● Customer Number	○ US Patent Practitioner	○ Limited Recognition (37 CFR 11.9)
Customer Number	38139		

Domestic Benefit/National Stage Information:

This section allows for the applicant to either claim benefit under 35 U.S.C. 119(e), 120, 121, 365(c), or 386(c) or indicate National Stage entry from a PCT application. Providing benefit claim information in the Application Data Sheet constitutes the specific reference required by 35 U.S.C. 119(e) or 120, and 37 CFR 1.78.
When referring to the current application, please leave the "Application Number" field blank.

Prior Application Status	Patented				Remove
Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)	Patent Number	Issue Date (YYYY-MM-DD)
	reissued of	12/601275	2010-05-31	8269523	2012-09-18

| **Application Data Sheet 37 CFR 1.76** | Attorney Docket Number | V-0070US |
| | Application Number | |

Title of Invention	VLSI Layouts of Fully Connected Generalized Networks

Prior Application Status	Expired		Remove
Application Number	Continuity Type	Prior Application Number	Filing or 371(c) Date (YYYY-MM-DD)
12/601275	a 371 of international	PCT/US08/64605	2008-05-22
Prior Application Status	Expired		Remove
Application Number	Continuity Type	Prior Application Number	Filing or 371(c) Date (YYYY-MM-DD)
PCT/US08/64605	Claims benefit of provisional	60/940394	2007-05-25
Prior Application Status			Remove
Application Number	Continuity Type	Prior Application Number	Filing or 371(c) Date (YYYY-MM-DD)
	Continuation of	PCT/US08/64605	2008-05-22

Additional Domestic Benefit/National Stage Data may be generated within this form by selecting the **Add** button.

Foreign Priority Information:

This section allows for the applicant to claim priority to a foreign application. Providing this information in the application data sheet constitutes the claim for priority as required by 35 U.S.C. 119(b) and 37 CFR 1.55. When priority is claimed to a foreign application that is eligible for retrieval under the priority document exchange program (PDX) the information will be used by the Office to automatically attempt retrieval pursuant to 37 CFR 1.55(i)(1) and (2). Under the PDX program, applicant bears the ultimate responsibility for ensuring that a copy of the foreign application is received by the Office from the participating foreign intellectual property office, or a certified copy of the foreign priority application is filed, within the time period specified in 37 CFR 1.55(g)(1).

			Remove
Application Number	Country	Filing Date (YYYY-MM-DD)	Access Code (if applicable)

Additional Foreign Priority Data may be generated within this form by selecting the **Add** button.

Statement under 37 CFR 1.55 or 1.78 for AIA (First Inventor to File) Transition Applications

☐ This application (1) claims priority to or the benefit of an application filed before March 16, 2013 and (2) also contains, or contained at any time, a claim to a claimed invention that has an effective filing date on or after March 16, 2013.
NOTE: By providing this statement under 37 CFR 1.55 or 1.78, this application, with a filing date on or after March 16, 2013, will be examined under the first inventor to file provisions of the AIA.

| **Application Data Sheet 37 CFR 1.76** | Attorney Docket Number | V-0070US |
| | Application Number | |

| Title of Invention | VLSI Layouts of Fully Connected Generalized Networks |

Authorization or Opt-Out of Authorization to Permit Access:

When this Application Data Sheet is properly signed and filed with the application, applicant has provided written authority to permit a participating foreign intellectual property (IP) office access to the instant application-as-filed (see paragraph A in subsection 1 below) and the European Patent Office (EPO) access to any search results from the instant application (see paragraph B in subsection 1 below).

Should applicant choose not to provide an authorization identified in subsection 1 below, applicant **must opt-out** of the authorization by checking the corresponding box A or B or both in subsection 2 below.

NOTE: This section of the Application Data Sheet is **ONLY** reviewed and processed with the **INITIAL** filing of an application. After the initial filing of an application, an Application Data Sheet cannot be used to provide or rescind authorization for access by a foreign IP office(s). Instead, Form PTO/SB/39 or PTO/SB/69 must be used as appropriate.

1. Authorization to Permit Access by a Foreign Intellectual Property Office(s)

A. Priority Document Exchange (PDX) - Unless box A in subsection 2 (opt-out of authorization) is checked, the undersigned hereby **grants the USPTO authority** to provide the European Patent Office (EPO), the Japan Patent Office (JPO), the Korean Intellectual Property Office (KIPO), the State Intellectual Property Office of the People's Republic of China (SIPO), the World Intellectual Property Organization (WIPO), and any other foreign intellectual property office participating with the USPTO in a bilateral or multilateral priority document exchange agreement in which a foreign application claiming priority to the instant patent application is filed, access to: (1) the instant patent application-as-filed and its related bibliographic data, (2) any foreign or domestic application to which priority or benefit is claimed by the instant application and its related bibliographic data, and (3) the date of filing of this Authorization. See 37 CFR 1.14(h)(1).

B. Search Results from U.S. Application to EPO - Unless box B in subsection 2 (opt-out of authorization) is checked, the undersigned hereby **grants the USPTO authority** to provide the EPO access to the bibliographic data and search results from the instant patent application when a European patent application claiming priority to the instant patent application is filed. See 37 CFR 1.14(h)(2).

The applicant is reminded that the EPO's Rule 141(1) EPC (European Patent Convention) requires applicants to submit a copy of search results from the instant application without delay in a European patent application that claims priority to the instant application.

2. Opt-Out of Authorizations to Permit Access by a Foreign Intellectual Property Office(s)

☐   A. Applicant **DOES NOT** authorize the USPTO to permit a participating foreign IP office access to the instant application-as-filed. If this box is checked, the USPTO will not be providing a participating foreign IP office with any documents and information identified in subsection 1A above.

☐   B. Applicant **DOES NOT** authorize the USPTO to transmit to the EPO any search results from the instant patent application. If this box is checked, the USPTO will not be providing the EPO with search results from the instant application.

NOTE: Once the application has published or is otherwise publicly available, the USPTO may provide access to the application in accordance with 37 CFR 1.14.

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	V-0070US
	Application Number	

Title of Invention	VLSI Layouts of Fully Connected Generalized Networks

Applicant Information:

Providing assignment information in this section does not substitute for compliance with any requirement of part 3 of Title 37 of CFR to have an assignment recorded by the Office.

Applicant   1

If the applicant is the inventor (or the remaining joint inventor or inventors under 37 CFR 1.45), this section should not be completed. The information to be provided in this section is the name and address of the legal representative who is the applicant under 37 CFR 1.43; or the name and address of the assignee, person to whom the inventor is under an obligation to assign the invention, or person who otherwise shows sufficient proprietary interest in the matter who is the applicant under 37 CFR 1.46. If the applicant is an applicant under 37 CFR 1.46 (assignee, person to whom the inventor is obligated to assign, or person who otherwise shows sufficient proprietary interest) together with one or more joint inventors, then the joint inventor or inventors who are also the applicant should be identified in this section.

[Clear]

● Assignee	○ Legal Representative under 35 U.S.C. 117	○ Joint Inventor
○ Person to whom the inventor is obligated to assign.	○ Person who shows sufficient proprietary interest	

If applicant is the legal representative, indicate the authority to file the patent application, the inventor is:

Name of the Deceased or Legally Incapacitated Inventor:

If the Applicant is an Organization check here.   ☒

~~Organization Name~~	~~Konda Technologies Inc~~ Venkat Konda

Mailing Address Information For Applicant:

Address 1	6278 Grand Oak Way		
Address 2			
City	San Jose	**State/Province**	CA
Country	US	Postal Code	95135
Phone Number	408-472-3273	Fax Number	408-238-2478
Email Address	venkat@kondatech.com		

Additional Applicant Data may be generated within this form by selecting the Add button.

Assignee Information including Non-Applicant Assignee Information:

Providing assignment information in this section does not substitute for compliance with any requirement of part 3 of Title 37 of CFR to have an assignment recorded by the Office.

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	V-0070US
	Application Number	

Title of Invention	VLSI Layouts of Fully Connected Generalized Networks

Assignee   1

Complete this section if assignee information, including non-applicant assignee information, is desired to be included on the patent application publication. An assignee-applicant identified in the "Applicant Information" section will appear on the patent application publication as an applicant. For an assignee-applicant, complete this section only if identification as an assignee is also desired on the patent application publication.

If the Assignee or Non-Applicant Assignee is an Organization check here.	☐

Prefix	**Given Name**	Middle Name	**Family Name**	Suffix

Mailing Address Information For Assignee including Non-Applicant Assignee:

Address 1	
Address 2	

City		**State/Province**	
Country		Postal Code	
Phone Number		Fax Number	
Email Address	

Additional Assignee or Non-Applicant Assignee Data may be generated within this form by selecting the Add button.

Signature:

NOTE: This Application Data Sheet must be signed in accordance with 37 CFR 1.33(b). **However, if this Application Data Sheet is submitted with the INITIAL filing of the application and either box A or B is not checked in subsection 2 of the "Authorization or Opt-Out of Authorization to Permit Access" section, then this form must also be signed in accordance with 37 CFR 1.14(c).**

This Application Data Sheet **must** be signed by a patent practitioner if one or more of the applicants is a **juristic entity** (e.g., corporation or association). If the applicant is two or more joint inventors, this form must be signed by a patent practitioner, **all** joint inventors who are the applicant, or one or more joint inventor-applicants who have been given power of attorney (e.g., see USPTO Form PTO/AIA/81) on behalf of **all** joint inventor-applicants.

See 37 CFR 1.4(d) for the manner of making signatures and certifications.

Signature	/Venkat Konda/			Date (YYYY-MM-DD)	2019-07-04
First Name	Venkat	**Last Name**	Konda	Registration Number	

Additional Signature may be generated within this form by selecting the Add button.

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	V-0070US
	Application Number	

Title of Invention	VLSI Layouts of Fully Connected Generalized Networks

This collection of information is required by 37 CFR 1.76. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 23 minutes to complete, including gathering, preparing, and submitting the completed application data sheet form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

STATEMENT UNDER 37 CFR 3.73(c)

Applicant/Patent Owner: Venkat Konda

Application No./Patent No.: 16/202,067   Filed/Issue Date: 11/27/2018

Titled: VLSI Layouts of Fully Connected Generalized Networks

Venkat Konda   , a   Individual

(Name of Assignee)   (Type of Assignee, e.g., corporation, partnership, university, government agency, etc.)

states that, for the patent application/patent identified above, it is (choose **one** of options 1, 2, 3 or 4 below):

1. ☐ The assignee of the entire right, title, and interest.

2. ☐ An assignee of less than the entire right, title, and interest (check applicable box):

   ☐ The extent (by percentage) of its ownership interest is _____%. Additional Statement(s) by the owners holding the balance of the interest must be submitted to account for 100% of the ownership interest.

   ☐ There are unspecified percentages of ownership. The other parties, including inventors, who together own the entire right, title and interest are:

   ┌───┐
   │   │
   │   │
   │   │
   │   │
   └───┘

   Additional Statement(s) by the owner(s) holding the balance of the interest must be submitted to account for the entire right, title, and interest.

3. ☐ The assignee of an undivided interest in the entirety (a complete assignment from one of the joint inventors was made). The other parties, including inventors, who together own the entire right, title, and interest are:

   ┌───┐
   │   │
   │   │
   │   │
   └───┘

   Additional Statement(s) by the owner(s) holding the balance of the interest must be submitted to account for the entire right, title, and interest.

4. ☐ The recipient, via a court proceeding or the like (*e.g.*, bankruptcy, probate), of an undivided interest in the entirety (a complete transfer of ownership interest was made). The certified document(s) showing the transfer is attached.

The interest identified in option 1, 2 or 3 above (not option 4) is evidenced by either (choose **one** of options A or B below):

A. ☐ An assignment from the inventor(s) of the patent application/patent identified above. The assignment was recorded in the United States Patent and Trademark Office at Reel _____, Frame _____, or for which a copy thereof is attached.

B. ☑ A chain of title from the inventor(s), of the patent application/patent identified above, to the current assignee as follows:

   1. From: Venkat Konda   To: Konda Technologies Inc.

      The document was recorded in the United States Patent and Trademark Office at
      Reel 029513_____, Frame 0134_____, or for which a copy thereof is attached.

   2. From: Konda Technologies Inc.   To: Venkat Konda

      The document was recorded in the United States Patent and Trademark Office at
      Reel 048887_____, Frame 0397_____, or for which a copy thereof is attached.

[Page 1 of 2]

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

STATEMENT UNDER 37 CFR 3.73(c)

3. From: _____ To: _____

    The document was recorded in the United States Patent and Trademark Office at

    Reel _____, Frame _____, or for which a copy thereof is attached.

4. From: _____ To: _____

    The document was recorded in the United States Patent and Trademark Office at

    Reel _____, Frame _____, or for which a copy thereof is attached.

5. From: _____ To: _____

    The document was recorded in the United States Patent and Trademark Office at

    Reel _____, Frame _____, or for which a copy thereof is attached.

6. From: _____ To: _____

    The document was recorded in the United States Patent and Trademark Office at

    Reel _____, Frame _____, or for which a copy thereof is attached.

☐   Additional documents in the chain of title are listed on a supplemental sheet(s).

☑   As required by 37 CFR 3.73(c)(1)(i), the documentary evidence of the chain of title from the original owner to the assignee was, or concurrently is being, submitted for recordation pursuant to 37 CFR 3.11.

    [NOTE: A separate copy (i.e., a true copy of the original assignment document(s)) must be submitted to Assignment Division in accordance with 37 CFR Part 3, to record the assignment in the records of the USPTO. See MPEP 302.08]

The undersigned (whose title is supplied below) is authorized to act on behalf of the assignee.

/Venkat Konda/	07/04/2019
Signature	Date
Venkat Konda	
Printed or Typed Name	Title or Registration Number

[Page 2 of 2]

Privacy Act Statement

The **Privacy Act of 1974 (P.L. 93-579)** requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (*i.e.*, GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

Electronic Acknowledgement Receipt

EFS ID:	36501210
Application Number:	16202067
International Application Number:	
Confirmation Number:	8134
Title of Invention:	VLSI Layouts of Fully Connected Generalized Networks
First Named Inventor/Applicant Name:	Venkat Konda
Customer Number:	38139
Filer:	Venkat Konda
Filer Authorized By:	
Attorney Docket Number:	V-0070US
Receipt Date:	04-JUL-2019
Filing Date:	27-NOV-2018
Time Stamp:	00:30:12
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Petition for review by the PCT legal office	V0070US-CoverLetter.pdf	89333 541783df09856a090737707b4109920e57bf55f6	no	2

Warnings:

Information:					
2	Oath or Declaration filed	Assignment-12-20-12.pdf	722170 2ef2bddfa655f1f0174a064bfa928a25a90c7331	no	5
Warnings:					
Information:					
3	Application Data Sheet	aia0014-Scan.pdf	10356973 43ecc4e0f9197da09e317902d51252399ecf77ec	no	7
Warnings:					
Information:					
This is not an USPTO supplied ADS fillable form					
4	Assignee showing of ownership per 37 CFR 3.73	aia0096.pdf	141428 003071150b09ad62682a58669142c873777ed7e2	no	3
Warnings:					
Information:					
		Total Files Size (in bytes):	11309904		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.
National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.
New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

To: venkat@kondatech.com,vkonda@gmail.com,
From: PAIR_eOfficeAction@uspto.gov
Cc: PAIR_eOfficeAction@uspto.gov
Subject: Private PAIR Correspondence Notification for Customer Number 38139

Jun 13, 2019 04:07:32 AM

Dear PAIR Customer:

Konda Technologies, Inc
6278 GRAND OAK WAY
SAN JOSE, CA 95135
UNITED STATES

The following USPTO patent application(s) associated with your Customer Number, 38139 , have
new outgoing correspondence. This correspondence is now available for viewing in Private PAIR.

The official date of notification of the outgoing correspondence will be indicated on the form PTOL-90
accompanying the correspondence.

Disclaimer:
The list of documents shown below is provided as a courtesy and is not part of the official file
wrapper. The content of the images shown in PAIR is the official record.

Application	Document	Mailroom Date	Attorney Docket No.
16202067	M327	06/11/2019	V-0070US

To view your correspondence online or update your email addresses, please visit us anytime at
https://sportal.uspto.gov/secure/myportal/privatepair.

If you have any questions, please email the Electronic Business Center (EBC) at EBC@uspto.gov
with 'e-Office Action' on the subject line or call 1-866-217-9197 during the following hours:

    Monday - Friday 6:00 a.m. to 12:00 a.m.

Thank you for prompt attention to this notice,

UNITED STATES PATENT AND TRADEMARK OFFICE

PATENT APPLICATION INFORMATION RETRIEVAL SYSTEM

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

MAILED

JUN 1 1 2019

INTERNATIONAL PATENT LEGAL ADM.

Konda Technologies, Inc
6278 GRAND OAK WAY
SAN JOSE CA 95135

In re Application of   :
Konda Technologies Inc.   :
Application No.: 16/202,067   :   NOTIFICATION
Filing Date: 27 November 2018   :
Attorney Docket No.: V-0070US   :

    This Notification is in response to the filing of a purported Petition Under 37 CFR 1.78, in the United States Patent and Trademark Office on 27 November 2018 and submission of 17 April 2019.

    The applicant is a juristic entity. The Petition is not signed in accordance with 37 CFR 1.33(b)(3), stating "Unless otherwise specified, all papers submitted on behalf of a juristic entity must be signed by a registered practitioner." It will not be treated on the merits.

    A request to change the applicant is governed by 37 CFR 1.46(c)(2). Applicant must supply a marked up ADS showing the changes and comply with 37 CFR 3.73. That showing must show the entire chain of title from the inventors to the new applicant.

    Any further correspondence with respect to this matter may be filed electronically via EFS-Web selecting the document description "Petition for review and processing by the PCT Legal Office" or by mail addressed to Mail Stop PCT, Commissioner for Patents, Office of PCT Legal Administration, P.O. Box 1450, Alexandria, Virginia 22313-1450, with the contents of the letter marked to the attention of the International Patent Legal Administration.

/Erin P. Thomson/
Erin P. Thomson
Attorney Advisor
International Patent Legal Administration
571-272-3292

To: venkat@kondatech.com,vkonda@gmail.com,
From: PAIR_eOfficeAction@uspto.gov
Cc: PAIR_eOfficeAction@uspto.gov
Subject: Private PAIR Correspondence Notification for Customer Number 38139

Apr 18, 2019 05:22:12 AM

Dear PAIR Customer:

Konda Technologies, Inc
6278 GRAND OAK WAY
SAN JOSE, CA 95135
UNITED STATES

The following USPTO patent application(s) associated with your Customer Number, 38139 , have new outgoing correspondence. This correspondence is now available for viewing in Private PAIR.

The official date of notification of the outgoing correspondence will be indicated on the form PTOL-90 accompanying the correspondence.

Disclaimer:
The list of documents shown below is provided as a courtesy and is not part of the official file wrapper. The content of the images shown in PAIR is the official record.

Application	Document	Mailroom Date	Attorney Docket No.
16202067	M327	04/17/2019	V-0070US

To view your correspondence online or update your email addresses, please visit us anytime at https://sportal.uspto.gov/secure/myportal/privatepair.

If you have any questions, please email the Electronic Business Center (EBC) at EBC@uspto.gov with 'e-Office Action' on the subject line or call 1-866-217-9197 during the following hours:

     Monday - Friday 6:00 a.m. to 12:00 a.m.

Thank you for prompt attention to this notice,

UNITED STATES PATENT AND TRADEMARK OFFICE

PATENT APPLICATION INFORMATION RETRIEVAL SYSTEM

UNITED STATES PATENT AND TRADEMARK OFFICE

MAILED

APR 1 7 2019

INTERNATIONAL PATENT LEGAL ADM.

Konda Technologies, Inc
6278 GRAND OAK WAY
SAN JOSE CA 95135

In re Application of   :
Konda Technologies Inc.   :
Application No.: 16/202,067   :   NOTIFICATION
Filing Date: 27 November 2018   :
Attorney Docket No.: V-0070US   :

        This Notification is in response to the filing of a purported Petition Under 37 CFR 1.78, in the United States Patent and Trademark Office on 27 November 2018.

        The applicant is a juristic entity. The Petition is not signed in accordance with 37 CFR 1.33(b)(3), stating "Unless otherwise specified, all papers submitted on behalf of a juristic entity must be signed by a registered practitioner." It will not be treated on the merits.

        Any further correspondence with respect to this matter may be filed electronically via EFS-Web selecting the document description "Petition for review and processing by the PCT Legal Office" or by mail addressed to Mail Stop PCT, Commissioner for Patents, Office of PCT Legal Administration, P.O. Box 1450, Alexandria, Virginia 22313-1450, with the contents of the letter marked to the attention of the International Patent Legal Administration.

/Erin P. Thomson/
Erin P. Thomson
Attorney Advisor
International Patent Legal Administration
571-272-3292

Application Number: 16/202,067   (Venkat Konda)   Art Unit: 3992

In The United States Patent And Trademark Office

Application Number: 16/202,067

Application Filed: 11/27/2018

Applicant(s): Venkat Konda

5 Title: VLSI Layouts of Fully Connected Generalized Networks

Examiner/Art Unit: Ton, My Trang / 3992

Priority Date: 5/25/2007

San Jose, 2019 Apr 17, Wed

Mail Stop PCT

10 Commissioner for Patents

Office of PCT Legal Administration

P.O. Box 1450

Alexandria, Virginia, 22313-1450

15 Dear Sir/Madam:

In response to the notification mailed 2019 April 17, which is in response to the filing of a purported Petition under 37 CFR 1.78 in the United States Patent and Trademark Office on 27 November 2018, please consider the following:

1) On April 15, 2019 the current application was assigned to Venkat Konda
20 by Konda Technologies Inc. which was recorded on the same day by the United States Patent and Trademark Office.

2) Accordingly a corrected Application Data Sheet is also filed along with this letter.

For the above reasons, applicant submits that now the applicant Venkat Konda is Pro Se
25 for the current application. Therefore applicant submits that the Petition under 37 CFR

Application Number: 16/202,067   (Venkat Konda)   Art Unit: 3992

1.78 in the United States Patent and Trademark Office submitted on 27 November 2018 is now proper, which action he respectfully solicits.

Very respectfully,

5   /Venkat Konda/

Venkat Konda

6278 Grand Oak Way

San Jose, CA 95135

Phone: 408-472-3273

10   Fax: 408-238-2478

Electronic Acknowledgement Receipt

EFS ID:	35753705
Application Number:	16202067
International Application Number:	
Confirmation Number:	8134
Title of Invention:	VLSI Layouts of Fully Connected Generalized Networks
First Named Inventor/Applicant Name:	
Customer Number:	38139
Filer:	Venkat Konda
Filer Authorized By:	
Attorney Docket Number:	V-0070US
Receipt Date:	17-APR-2019
Filing Date:	27-NOV-2018
Time Stamp:	15:03:40
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Application Data Sheet to update/ correct info	CorrectedADS.pdf	86836 c86a3a7b937abbabeb898d51165dd30dd1 666973	no	6

Warnings:

Information:

2	Petition for review by the PCT legal office	Assignment-V0070US.pdf	67531 b87e75cd0c13f4467fbbcb376c632d8f9870ee45	no	2

Warnings:

Information:

	Total Files Size (in bytes):	154367

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.
National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.
New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

CORRECTED ADS FORM

Application Number	16202067
Title of Invention	VLSI Layouts of Fully Connected Generalized Networks

Inventor Information

If no data is shown, no data has been corrected

	Data of Record	Updated Data
Order Number		
Name		

Residence Information

Residency		
City		
State		
Country of Residence		

Mailing Address of Inventor

Address 1		
Address 2		
City,State/Province, Postal Code		
Country		

Corrected ADS 1.0

Document Description: Application Data Sheet to update/correct info
Doc Code: ADS.CORR

Application Information

	Data of Record	Updated Data
Title of Invention	VLSI Layouts of Fully Connected Generalized Networks	
Attorney Docket Number	V-0070US	
Entity Type	Small	

Domestic Benefit/National Stage Information

If no data is shown, no data has been corrected

This section allows for the applicant to either claim benefit under 35 U.S.C. 119(e), 120, 121,365(c), or 386(c) or indicate National Stage entry from a PCT application. Providing this information in the application data sheet constitutes the specific reference required by 35 U.S.C. 119(e) or 120, and 37 CFR 1.78(a).

		Data of Record	Updated Data
	Prior Application Status	~~patented~~	pending
	Application Number	~~16202067~~	PCT/US08/64605
	Continuity Type	~~REI~~	PRO
	Prior Application Number	~~12601275~~	60940394
	Filing Date (YYYY-MM-DD)	~~2010-05-31~~	2007-05-25
	Patent Number	~~8269523~~	
	Issue Date (YYYY-MM-DD)	~~2012-09-18~~	0001-01-01

Corrected ADS 1.0

Document Description: Application Data Sheet to update/correct info
Doc Code: ADS.CORR

		Data of Record	Updated Data
	Prior Application Status		
	Application Number	~~12601275~~	16202067
	Continuity Type	~~NST~~	CON
	Prior Application Number	PCT/US08/64605	
	Filing Date (YYYY-MM-DD)	2008-05-22	
	Patent Number		
	Issue Date (YYYY-MM-DD)		

		Data of Record	Updated Data
	Prior Application Status	~~pending~~	
	Application Number	~~PCT/US08/64605~~	12601275
	Continuity Type	~~PRO~~	NST
	Prior Application Number	~~60940394~~	PCT/US08/64605
	Filing Date (YYYY-MM-DD)	~~2007-05-25~~	2008-05-22
	Patent Number		
	Issue Date (YYYY-MM-DD)	~~0001-01-01~~	

		Data of Record	Updated Data
	Prior Application Status		patented
	Application Number		16202067
	Continuity Type		REI
	Prior Application Number		12601275
	Filing Date (YYYY-MM-DD)		2010-05-31
	Patent Number		8269523
	Issue Date (YYYY-MM-DD)		2012-09-18

Corrected ADS 1.0

Document Description: Application Data Sheet to update/correct info
Doc Code: ADS.CORR

Foreign Priority Information

If no data is shown, no data has been corrected

This section allows for the applicant to claim priority to a foreign application. Providing this information in the application data sheet constitutes the claim for priority as required by 35 U.S.C. 119(b) and 37 CFR 1.55. When priority is claimed to a foreign application that is eligible for retrieval under the priority document exchange program (PDX) the information will be used by the Office to automatically attempt retrieval pursuant to 37 CFR 1.55(i)(1) and (2). Under the PDX program, applicant bears the ultimate responsibility for ensuring that a copy of the foreign application is received by the Office from the participating foreign intellectual property office, or a certified copy of the foreign priority application is filed, within the time period specified in 37 CFR 1.55(g)(1).

		Data of Record	Updated Data
	Application Number		
	Country		<u>US</u>
	Filing Date		
	Access Code		

Applicant Information

If no data is shown, no data has been corrected
Providing assignment information in this section does not substitute for compliance with any requirement of part 3 of Title 37 of CFR to have an assignment recorded by the Office.

	Data of Record	Updated Data
Applicant Type	~~ASG~~	
If applicant is the legal representative, indicate the authority to file the patent application, the inventor is		
Name of the Deceased or Legally Incapacitated Inventor		
Applicant is an Organization	~~Yes~~	
Name		
Organization Name	~~Konda Technologies Inc.~~	
Address 1	~~6278 Grand Oak Way~~	

Corrected ADS 1.0

Document Description: Application Data Sheet to update/correct info
Doc Code: ADS.CORR

Address 2		
City,State/Province,Postal Code	San Jose CA 95135	
Country	US	
Phone Number		
Fax Number		
Email Address		

Assignee Information including Non-Applicant Assignee Information

If no data is shown, no data has been corrected
Providing this information in the application data sheet does not substitute for compliance with any requirement of part 3 of Title 37 of the CFR to have an assignment recorded in the Office

	Data of Record	Updated Data
Order	1	
Applicant is an Organization	Yes	
Name		
Organization Name	Konda Technologies Inc.	

Mailing Address

Address 1	6278 Grand Oak Way	
Address 2		
City,State/Province,Postal Code	San Jose CA 95135	
Country	US	
Phone Number		
Fax Number		

Corrected ADS 1.0

Document Description: Application Data Sheet to update/correct info
Doc Code: ADS.CORR

Email Address		

Signature

NOTE: This Application Data Sheet must be signed in accordance with 37 CFR 1.33(b).

This Application Data Sheet **must** be signed by a patent practitioner if one or more of the applicants is a **juristic entity** (e.g., corporation or association). If the applicant is two or more joint inventors, this form must be signed by a patent practitioner, **all** joint inventors who are the applicant, or one or more joint inventor-applicants who have been given power of attorney (e.g., see USPTO Form PTO/AIA/81) on behalf of **all** joint inventor-applicants.

See 37 CFR 1.4(d) for the manner of making signatures and certifications.

Signature	/venkat Konda/	Registration Number	
First Name	venkat	Last Name	konda

Corrected ADS 1.0

PATENT ASSIGNMENT COVER SHEET

Electronic Version v1.1
Stylesheet Version v1.2

EPAS ID: PAT5474857

SUBMISSION TYPE:	NEW ASSIGNMENT
NATURE OF CONVEYANCE:	ASSIGNMENT

CONVEYING PARTY DATA

Name	Execution Date
KONDA TECHNOLOGIES INC.	04/15/2019

RECEIVING PARTY DATA

Name:	VENKAT KONDA
Street Address:	6278 GRAND OAK WAY
City:	SAN JOSE
State/Country:	CALIFORNIA
Postal Code:	95135

PROPERTY NUMBERS Total: 2

Property Type	Number
Patent Number:	8269523
Application Number:	16202067

CORRESPONDENCE DATA

Fax Number:	(408)238-2478

Correspondence will be sent to the e-mail address first; if that is unsuccessful, it will be sent using a fax number, if provided; if that is unsuccessful, it will be sent via US Mail.

Phone:	4084723273
Email:	venkat@kondatech.com
Correspondent Name:	KONDA TECHNOLOGIES INC.
Address Line 1:	6278 GRAND OAK WAY
Address Line 4:	SAN JOSE, CALIFORNIA 95135

NAME OF SUBMITTER:	VENKAT KONDA
SIGNATURE:	/venkat Konda/
DATE SIGNED:	04/15/2019
	This document serves as an Oath/Declaration (37 CFR 1.63).

Total Attachments: 2
source=Konda-pat-exe#page1.tif
source=Konda-pat-exe#page2.tif

ASSIGNMENT OF PATENTS

WHEREAS, Konda Technologies, Inc., a California corporation, located at 6278 Grand Oak Way, San Jose, CA 95135, hereinafter referred to as "Assignor," is the record owner of the patents described as

1)   Title: VLSI Layouts of Fully Connected Generalized Networks
       US Patent No.:   8,269,523
       Patent Issue Date: September 18, 2012;

2)   Title: VLSI Layouts of Fully Connected Generalized Networks
       US Patent Application No.:   16/202,067
       Patent Filing Date: November 27, 2018;

WHEREAS, Konda, Venkat of San Jose, California, having a residence at 6278 Grand Oak Way, San Jose, CA, hereinafter referred to as "ASSIGNEE," wishes to acquire all right, title and interest to and under the patents described above owned by Assignor and in and to any and all improvements to said applications and in any Letters Patent and Registrations which may be granted on the same in the United States or any country throughout the world;
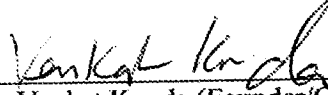
For good and valuable consideration, receipt of which is hereby acknowledged by Assignor, effective 15th day of April, 2019, has assigned, and by these presents does assign to Assignee all right, title and interest for the United States and all foreign countries, in and to any and all improvements for the applications described above and in and to said applications and to all utility divisional continuing, substitute, renewal, reissue, and all other patent applications which have been or shall be filed in the United States and all foreign counterparts (including patent, utility model and industrial designs), and in and to any Letters Patent and Registrations which may hereafter be granted on the same in the United States and all countries throughout the world, and to claim the priority from the application as provided by the Paris Convention. The right, title and interest is to be held and enjoyed by Assignee and Assignee's successors and assigns as fully and exclusively as it would have been held and enjoyed by Assignor had this Assignment not been made, for the full term of any Letters Patent and Registrations which may be granted thereon, or of any division, renewal, continuation in whole or in part, substitution, conversion, reissue, prolongation or extension thereof.

Assignor further agrees that they will, without charge to Assignee, but at Assignee's expense, (a) cooperate with Assignee in the prosecution of U.S. Patent applications and foreign counterparts on the applications and any improvements, (b) execute, verify, acknowledge and deliver all such further papers, including patent applications and instruments of transfer, and (c) perform such other acts as Assignee lawfully may request to obtain or maintain Letters Patent and Registrations for the applications and improvements in any and all countries, and to vest title thereto in Assignee, or Assignee's successors and assigns.

Page 2

IN TESTIMONY WHEREOF, Assignor has signed on the date indicated.

Date: ___4/15/2019___   By: _____

Venkat Konda (Founder/CEO)
Konda Technologies Inc.
6278 Grand Oak Way
San Jose, CA 95135

PLUS Search Results for S/N 16202067, Searched Tue Apr 09 11:23:13 EDT 2019
The Patent Linguistics Utility System (PLUS) is a USPTO automated search system
for U.S. Patents from 1971 to the present PLUS is a query-by-example search system which
produces a list of patents that are most closely related linguistically to the application
searched. This search was prepared by the staff of the Scientific and Technical Information
Center, SIRA.

5784374 99   5864552 99
5218676 99
3920914 99
3912873 99
3851105 99
4023141 99
4038638 99
4289932 99
4400627 99
4417245 99
4473900 99
4626066 99
4787692 99
4817083 99
4817084 99
4821034 99
4845736 99
4914430 99
4975909 99
5005167 99
5007070 99
5179558 99
5216668 99
5274642 99
5276425 99
5291477 99
5323386 99
5337378 99
5402415 99
5406556 99
5440549 99
5440553 99
5450074 99
5451936 99
5455956 99
5526352 99
5555543 99
5560038 99
5590129 99
5604867 99
5627925 99
5634004 99
5655140 99
5734764 99
5737103 99
5754120 99
5801641 99
5805320 99
5812792 99

Litigation Search Report CRU 3999

APPLICATION NUMBER 16/202,067

TO: HETUL PATEL	From: MANUEL SALDANA
Location: CRU	Location: CRU 3999
Art Unit: 3992	REM04C71
Date: 02/25/2019	Phone: (571) 272-7740
	MANUEL.SALDANA@uspto.gov

Search Notes

Litigation was found for US Patent Number: **8,269,523**

3:18CV7581 KONDA V. FLEX LOGIX (OPEN)

5:18CV7581 KONDA V. FLEX LOGIX (OPEN)

1) I performed a KeyCite Search in Westlaw, which retrieves all history on the patent including any litigation.

2) I performed a search on the patent in Lexis CourtLink for any open dockets or closed cases.

3) I performed a search in Lexis in the Federal Courts and Administrative Materials databases for any cases found.

4) I performed a search in Lexis in the IP Journal and Periodicals database for any articles on the patent.

5) I performed a search in Lexis in the news databases for any articles about the patent or any articles about litigation on this patent.

US District Court Civil Docket

U.S. District - California Northern
(San Francisco)

3:18cv7581

Konda Technologies, Inc. v. Flex Logix Technologies, Inc.

This case was retrieved from the court on Wednesday, December 19, 2018

Date Filed:	**12/17/2018**	Class Code:	**OPEN**
Assigned To:	**Judge William H. Orrick**	Closed:	
Referred To:		Statute:	**35:271**
Nature of suit:	**Patent (830)**	Jury Demand:	**Plaintiff**
Cause:	**Patent Infringement**	Demand Amount:	**$0**
Lead Docket:	**None**	NOS Description:	**Patent**
Other Docket:	**None**		
Jurisdiction:	**Federal Question**		

Litigants	Attorneys
Konda Technologies, Inc. Plaintiff	Nitoj P. Singh LEAD ATTORNEY;ATTORNEY TO BE NOTICED Dhillon Law Group Inc. 177 Post Street, Suite 700 San Francisco, CA 94108 USA 415-433-1700 Fax: 415-520-6593 Email:Nsingh@dhillonlaw.Com Harmeet K. Dhillon ATTORNEY TO BE NOTICED Dhillon Law Group Inc. 177 Post Street, Suite 700 San Francisco, CA 94108 USA 415-433-1700 Fax: 415-520-6593 Email:Harmeet@dhillonlaw.Com
Flex Logix Technologies, Inc. Defendant	Steven McCall Perry LEAD ATTORNEY;ATTORNEY TO BE NOTICED Munger Tolles & Olson LLP 355 South Grand Avenue 35th Floor Los Angeles, CA 90071-1560 USA 213-683-9100 Fax: 213-687-3702 Email:Steven.Perry@mto.Com Elizabeth Ann Laughton ATTORNEY TO BE NOTICED

Munger, Tolles Olson LLP
350 South Grand Avenue 50th Floor
Los Angeles, CA 90071-3426
USA
213-683-9100
Fax: 213-687-3702
Email:Elizabeth.Laughton@mto.Com

Date	#	Proceeding Text
12/17/2018	1	COMPLAINT against Flex Logix Technologies, Inc. (Filing fee $ 400, receipt number 0971-12935682.). Filed byKonda Technologies, Inc.. (Attachments: # 1 Exhibit, # 2 Exhibit, # 3 Exhibit, # 4 Exhibit, # 5 Exhibit, # 6 Exhibit, # 7 Exhibit, # 8 Exhibit)(Singh, Nitoj) (Filed on 12/17/2018) (Entered: 12/17/2018)
12/18/2018		Electronic filing error. No Civil cover sheet filed. Please refer to Civil Local Rules 3-2(a) re civil cover sheet requirement. This filing will not be processed by the clerks office until the civil cover sheet is filed. Re: 1 Complaint, filed by Konda Technologies, Inc. (jmlS, COURT STAFF) (Filed on 12/18/2018) (Entered: 12/18/2018)
12/18/2018	2	Proposed Summons. (Singh, Nitoj) (Filed on 12/18/2018) (Entered: 12/18/2018)
12/18/2018	3	Civil Cover Sheet by Konda Technologies, Inc. . (Singh, Nitoj) (Filed on 12/18/2018) (Entered: 12/18/2018)
12/18/2018	4	Case assigned to Magistrate Judge Sallie Kim. Counsel for plaintiff or the removing party is responsible for serving the Complaint or Notice of Removal, Summons and the assigned judge's standing orders and all other new case documents upon the opposing parties. For information, visit E-Filing A New Civil Case at http://cand.uscourts.gov/ecf/caseopening.Standing orders can be downloaded from the court's web page at www.cand.uscourts.gov/judges. Upon receipt, the summons will be issued and returned electronically. Counsel is required to send chambers a copy of the initiating documents pursuant to L.R. 5-1(e)(7). A scheduling order will be sent by Notice of Electronic Filing (NEF) within two business days. Consent/Declination due by 1/2/2019. (jmlS, COURT STAFF) (Filed on 12/18/2018) (Entered: 12/18/2018)
12/19/2018	5	Initial Case Management Scheduling Order with ADR Deadlines: Joint Case Management Statement due by 3/11/2019. Initial Case Management Conference set for 3/18/2019 at 1:30 PM in San Francisco, Courtroom C, 15th Floor. (tnS, COURT STAFF) (Filed on 12/19/2018) (Entered: 12/19/2018)
12/19/2018	6	Summons Issued as to Defendant Flex Logix Technologies, Inc.. (tnS, COURT STAFF) (Filed on 12/19/2018) (Entered: 12/19/2018)
12/19/2018	7	REPORT on the filing of an action regarding patent infringement. (cc: form mailed to register). (tnS, COURT STAFF) (Filed on 12/19/2018) (Entered: 12/19/2018)
12/20/2018	8	CONSENT/DECLINATION to Proceed Before a US Magistrate Judge by Konda Technologies, Inc... (Singh, Nitoj) (Filed on 12/20/2018) (Entered: 12/20/2018)
12/27/2018	9	CLERK'S NOTICE OF IMPENDING REASSIGNMENT TO A U.S. DISTRICT COURT JUDGE: The Clerk of this Court will now randomly reassign this case to a District Judge because either (1) a party has not consented to the jurisdiction of a Magistrate Judge, or (2) time is of the essence in deciding a pending judicial action for which the necessary consents to Magistrate Judge jurisdiction have not been secured. You will be informed by separate notice of the district judge to whom this case is reassigned. ALL HEARING DATES PRESENTLY SCHEDULED BEFORE THE CURRENT MAGISTRATE JUDGE ARE VACATED AND SHOULD BE RE-NOTICED FOR HEARING BEFORE THE JUDGE TO WHOM THIS CASE IS REASSIGNED. This is a text only docket entry; there is no document associated with this notice. (mklS, COURT STAFF) (Filed on 12/27/2018) (Entered: 12/27/2018)
12/28/2018	10	ORDER REASSIGNING CASE. Case reassigned to Judge William H. Orrick for all further proceedings. Magistrate Judge Sallie Kim no longer assigned to the case. This case is assigned to a judge who participates in the Cameras in the Courtroom Pilot Project. See General Order 65 and http://cand.uscourts.gov/cameras. Signed by the Executive Committee on 12/28/18. (Attachments: # 1 Notice of Eligibility for Video Recording)(srnS, COURT STAFF) (Filed on 12/28/2018) (Entered: 12/28/2018)
01/03/2019	11	CASE MANAGEMENT CONFERENCE ORDER - Case Management Conference set for 3/26/2019 02:00 PM in San Francisco, Courtroom 02, 17th Floor. Case Management Statement due by 3/19/2019. Signed by Judge William H. Orrick on 1/3/2019. (jmdS, COURT STAFF) (Filed on 1/3/2019) (Entered: 01/03/2019)

01/09/2019	12	NOTICE of Appearance by Steven McCall Perry (Perry, Steven) (Filed on 1/9/2019) (Entered: 01/09/2019)
01/09/2019	13	NOTICE of Appearance by Elizabeth Ann Laughton (Laughton, Elizabeth) (Filed on 1/9/2019) (Entered: 01/09/2019)
01/10/2019	14	ADMINISTRATIVE MOTION whether case should be related to dismissed matter filed by Flex Logix Technologies, Inc.. Responses due by 1/14/2019. (Perry, Steven) (Filed on 1/10/2019) (Entered: 01/10/2019)
01/10/2019	--	Electronic filing error. This filing will not be processed by the clerks office. Re: 14 ADMINISTRATIVE MOTION whether case should be related to dismissed matter fil ed by Flex Logix Technologies, Inc. The Motion Must be E-filed into the Lower Case 18-4222 LHK for Judge Koh's Consideration. (aaaS, COURT STAFF) (Filed on 1/10/2019) (Entered: 01/10/2019)
01/16/2019	15	ORDER RELATING CASE by Judge Lucy H. Koh (granting 24 in 5:18-cv-04222-LHK Administrative Motion to Relate Cases). 18-7581-WHO is related to 18-4222-LHK and shall be reassigned to Judge Lucy H. Koh. The parties are instructed that all future filings in any reassigned case are to bear the initials of the newly assigned judge immediately after the case number. Any case management conference in any reassigned case will be rescheduled by the Court.(This is a text-only entry generated by the court. There is no document associated with this entry.). Signed by Judge Lucy H. Koh on 1/16/2019. (This is a text-only entry generated by the court. There is no document associated with this entry.) (ecgS, COURT STAFF) (Filed on 1/16/2019) (Entered: 01/16/2019)
01/18/2019	16	ORDER REASSIGNING CASE. Case reassigned to Judge Lucy H. Koh for all further proceedings pursuant to Order Granting Administrative Motion to Relate Cases. This case is assigned to a judge who participates in the Cameras in the Courtroom Pilot Project. See General Order 65 and http://cand.uscourts.gov/cameras. Judge William H. Orrick no longer assigned to the case. (Attachments: # 1 Notice of Eligibility for Video Recording)(bwS, COURT STAFF) (Filed on 1/18/2019) (Entered: 01/18/2019)

US District Court Civil Docket

U.S. District - California Northern
(San Jose)

5:18cv7581

Konda Technologies, Inc. v. Flex Logix Technologies, Inc.

This case was retrieved from the court on Tuesday, December 18, 2018

Date Filed:	**12/17/2018**	Class Code:	**OPEN**
Assigned To:		Closed:	
Referred To:		Statute:	**35:271**
Nature of suit:	**Patent (830)**	Jury Demand:	**Plaintiff**
Cause:	**Patent Infringement**	Demand Amount:	**$0**
Lead Docket:	**None**	NOS Description:	**Patent**
Other Docket:	**None**		
Jurisdiction:	**Federal Question**		

Litigants | Attorneys

Konda Technologies, Inc.
Plaintiff

Nitoj P. Singh
ATTORNEY TO BE NOTICED
Dhillon Law Group Inc.
177 Post Street Suite 700
San Francisco, CA 94108
USA
415-433-1700
Fax: 4154331700
Email: Nsingh@dhillonlaw.Com

Flex Logix Technologies, Inc.
Defendant

Date | # | Proceeding Text

Date	#	Proceeding Text
12/17/2018	1	COMPLAINT against Flex Logix Technologies, Inc. (Filing fee $ 400, receipt number 0971-12935682.). Filed by Konda Technologies, Inc.. (Attachments: # 1 Exhibit, # 2 Exhibit, # 3 Exhibit, # 4 Exhibit, # 5 Exhibit, # 6 Exhibit, # 7 Exhibit, # 8 Exhibit)(Singh, Nitoj) (Filed on 12/17/2018) (Entered: 12/17/2018)
12/18/2018	--	Electronic filing error. No Civil cover sheet filed. Please refer to Civil Local Rules 3-2(a) re civil cover sheet requirement. This filing will not be processed by the clerks of fice until the civil cover sheet is filed. Re: 1 Complaint, filed by Konda Technologies, Inc. (jmlS, COURT STAFF) (Filed on 12/18/2018) (Entered: 12/18/2018)
12/18/2018	2	Proposed Summons. (Singh, Nitoj) (Filed on 12/18/2018) (Entered: 12/18/2018)
12/18/2018	3	Civil Cover Sheet by Konda Technologies, Inc. . (Singh, Nitoj) (Filed on 12/18/2018) (Entered: 12/18/2018)
12/18/2018	4	Case assigned to Magistrate Judge Sallie Kim. Counsel for plaintiff or the removing party is responsible for serving the Complaint or Notice of Removal, Summons and the assigned judge's

standing orders and all other new case documents upon the opposing parties. For information, visit E-Filing A New Civil Case at http://cand.uscourts.gov/ecf/caseopening.Standing orders can be downloaded from the court's web page at www.cand.uscourts.gov/judges. Upon receipt, the summons will be issued and returned electronically. Counsel is required to send chambers a copy of the initiating documents pursuant to L.R. 5-1(e)(7). A scheduling order will be sent by Notice of Electronic Filing (NEF) within two business days. Consent/Declination due by 1/2/2019. (jmlS, COURT STAFF) (Filed on 12/18/2018) (Entered: 12/18/2018)

01/18/2019	17	Order re Consent/Declination to Magistrate Judge Jurisdiction. Signed by Judge Lucy H. Koh on 1/18/19. (lhklc2S, COURT STAFF) (Filed on 1/18/2019) (Entered: 01/18/2019)
01/18/2019	18	CLERK'S NOTICE RESETTING CASE MANAGEMENT CONFERENCE FOLLOWING REASSIGNMENT. Following the case reassignment to Hon. Lucy H. Koh, an Initial Case Management Conference set for 2/27/2019 02:00 PM in San Jose, Courtroom 7, 4th Floor before Hon. Lucy H. Koh. A Joint Case Management Conference Statement is due 7 days before the scheduled conference date. See Civil L.R. 16-9 and Civil L.R. 16-10(a). The parties shall familiarize themselves with the Scheduling Notes and Standing Orders for the Hon. Lucy H. Koh: http://cand.uscourts.gov/lhk. (This is a text-only entry generated by the court. There is no document associated with this entry.) (ecgS, COURT STAFF) (Filed on 1/18/2019) Modified on 1/18/2019 (ecgS, COURT STAFF). (Entered: 01/18/2019)
01/18/2019	--	Set/Reset Hearing per ECF No. 18 Clerk's Notice Initial Case Management Conference set for 2/27/2019 02:00 PM in San Jose, Courtroom 7, 4th Floor. (Correcting clerical data entry error re: time) (ecgS, COURT STAFF) (Filed on 1/18/2019) (Entered: 01/18/2019)
01/18/2019	19	CLERK'S NOTICE Continuing Case Management Conference. The 2/27/2019 Initial Case Management Conference is continued to 4/3/2019 02:00 PM in San Jose, Courtroom 8, 4th Floor. A Joint Case Management Conference Statement is due 7 days before the scheduled conference date. See Civil L.R. 16-9 and Civil L.R. 16-10(a). The parties shall familiarize themselves with the Scheduling Notes and Standing Orders for the Hon. Lucy H. Koh: http://cand.uscourts.gov/lhk. (This is a text-only entry generated by the court. There is no document associated with this entry.) (ecgS, COURT STAFF) (Filed on 1/18/2019) (Entered: 01/18/2019)
01/18/2019	20	CONSENT/DECLINATION to Proceed Before a US Magistrate Judge by Flex Logix Technologies, Inc... (Perry, Steven) (Filed on 1/18/2019) (Entered: 01/18/2019)
01/24/2019	21	MOTION to Dismiss COMPLAINT PURSUANT TO FED. R. CIV. P. 12(b)(6) AND TO STRIKE PORTIONS OF COMPLAINT PURSUANT TO FED. R. CIV. P. 12(f) filed by Flex Logix Technologies, Inc.. Motion Hearing set for 5/9/2019 01:30 PM in San Jose, Courtroom 8, 4th Floor before Judge Lucy H. Koh. Responses due by 2/7/2019. Replies due by 2/14/2019. (Attachments: # 1 Proposed Order)(Stone, Gregory) (Filed on 1/24/2019) (Entered: 01/24/2019)
01/24/2019	22	Declaration of Elizabeth A. Laughton in Support of 21 MOTION to Dismiss COMPLAINT PURSUANT TO FED. R. CIV. P. 12(b)(6) AND TO STRIKE PORTIONS OF COMPLAINT PURSUANT TO FED. R. CIV. P. 12(f) filed byFlex Logix Technologies, Inc.. (Related document(s) 21) (Stone, Gregory) (Filed on 1/24/2019) (Entered: 01/24/2019)
01/24/2019	23	Request for Judicial Notice IN SUPPORT OF MOTION TO DISMISS COMPLAINT PURSUANT TO FED. R. CIV. P. 12(b)(6) AND TO STRIKE PORTIONS OF COMPLAINT PURSUANT TO FED. R. CIV. P. 12(f) filed byFlex Logix Technologies, Inc.. (Stone, Gregory) (Filed on 1/24/2019) (Entered: 01/24/2019)
01/25/2019	--	Electronic filing error. Document not properly linked. [err102]Corrected by Clerk's Office. No further action is necessary. Re: 23 Request for Judicial Notice, filed by Flex Logix Technologies, Inc. (dhmS, COURT STAFF) (Filed on 1/25/2019) (Entered: 01/25/2019)
02/05/2019	24	STIPULATION WITH PROPOSED ORDER re 21 MOTION to Dismiss COMPLAINT PURSUANT TO FED. R. CIV. P. 12(b)(6) AND TO STRIKE PORTIONS OF COMPLAINT PURSUANT TO FED. R. CIV. P. 12(f) for Extension of Time to Submit Opposition and Reply Briefs filed by Konda Technologies, Inc.. (Singh, Nitoj) (Filed on 2/5/2019) (Entered: 02/05/2019)
02/06/2019	25	Order by Judge Lucy H. Koh Granting 24 Stipulation.(lhklc2, COURT STAFF) (Filed on 2/6/2019) (Entered: 02/06/2019)
02/21/2019	26	OPPOSITION/RESPONSE (re 21 MOTION to Dismiss COMPLAINT PURSUANT TO FED. R. CIV. P. 12(b)(6) AND TO STRIKE PORTIONS OF COMPLAINT PURSUANT TO FED. R. CIV. P. 12(f)) filed byKonda Technologies, Inc.. (Singh, Nitoj) (Filed on 2/21/2019) (Entered: 02/21/2019)
02/21/2019	27	DECLARATION of Venkat Konda, Ph.D. in Opposition to 21 MOTION to Dismiss COMPLAINT PURSUANT TO FED. R. CIV. P. 12(b)(6) AND TO STRIKE PORTIONS OF COMPLAINT PURSUANT TO FED. R. CIV. P. 12(f) filed byKonda Technologies, Inc.. (Attachments: # 1 Exhibit No. 1, # 2 Exhibit

No. 2, # 3 Exhibit No. 3, # 4 Exhibit No. 4, # 5 Exhibit No. 5, # 6 Exhibit No. 6, # 7 Exhibit No. 7)(Related document(s) 21) (Singh, Nitoj) (Filed on 2/21/2019) (Entered: 02/21/2019)

02/21/2019   28   Proposed Order re 26 Opposition/Response to Motion, by Konda Technologies, Inc.. (Attachments: # 1 Proposed First Amended Complaint, # 2 Exhibit 1 to Proposed FAC, # 3 Exhibit 2 to Proposed FAC, # 4 Exhibit 3 to Proposed FAC, # 5 Exhibit 4 to Proposed FAC, # 6 Exhibit 5 to Proposed FAC, # 7 Exhibit 6 to Proposed FAC, # 8 Exhibit 7 to Proposed FAC, # 9 Exhibit 8 to Proposed FAC, # 10 Exhibit 9 to Proposed FAC, # 11 Exhibit 10 to Proposed FAC, # 12 Exhibit 11 to Proposed FAC, # 13 Exhibit 12 to Proposed FAC, # 14 Exhibit 13 to Proposed FAC, # 15 Exhibit 14 to Proposed FAC, # 16 Exhibit 15 to Proposed FAC)(Singh, Nitoj) (Filed on 2/21/2019) (Entered: 02/21/2019)

Lexis Advance®
Research   More

Home / News / Results

Select Category
News   5

Results for: 8269523 *OR* 8,269,523
☐ | Actions

Filters

News

Clear |

Filter Results

News (5)

Group Duplicates: Off

☐   Sort by: Relevance

☐ 1. Chesswood Announces Results for Q2 2015

Marketwired (formerly Canadian Corporate Newswire) | Jul 30, 2015 | 967 words

... **8,269,523** 2,540,766 ...
... **8,269,523** 2,540,866 ...

☐ 2. Chesswood Announces Results for Q2 2015;Strong Originations Growth
and Acquisition Drive 58% Increase in Net Income Gross Finance
Receivables Exceed $500 million

Marketwired | Jul 30, 2015 | 960 words

... **8,269,523** 2,540,766 ...
... **8,269,523** 2,540,866 ...

☐ 3. Results For Q2 2015

Market News Publishing | Jul 30, 2015 | 730 words

... Leeper 8,269,423 2,540,766 David Obront 10,514,679 295,610 Barry W. Shafran
8,269,523 2,540,766 Frederick W. Steiner 9,430,742 1,379,547 Daniel Wittlin **8,269,523**
2,540,866 Jeffrey Wortsman 10,136,800 673,489 Total number of shares represented ...

☐ 4. Konda Technologies Assigned Patent

Targeted News Service ┊ Sep 21, 2012 ┊ 412 words ┊ Targeted News Service

... The full-text of the patent can be found at http://patft.uspto.gov/netacgi/nph-Parser?
Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=%2Fnetahtml%2FPTO%
2Fsrchnum.htm&r=1&f=G&l=50&s1=8,269,523.PN.&OS=PN/8,269,523&RS=PN/8,269,523
Written by Kusum Sangma; edited by Anand Kumar. For more ...
... San Jose, Calif. , has been assigned a patent (8,269,523) developed by Venkat Konda,
San Jose, Calif. , for ...

☐ 5. ICU Medical Inc. reports record quarterly earnings

Business Wire ┊ Feb 07, 1996 ┊ 872 words

...   Effect of income   tax credit   0.04   -   0.04   -   Net income per
      share $ 0.18 $ 0.10 $ 0.50 $ 0.39   Weighted average
   common and common   equivalent shares
   outstanding   9,021,908 7,466,895   8,269,523   7,453,716 CONTACT: ICU Medical
Inc., San Clemente   William Moore, 714/366-4325 ...

1

List of 20 Citing References for VLSI LAYOUTS OF FULLY CONNECTE...

Citing References (20)

Treatment	Title	Date	Type	Depth	Headnote(s)
Cited by	1. VLSI layouts of fully connected generalized networks LitAlert P2018-51-31	Dec. 18, 2018	Lit Alert		—
Cited by	2. VLSI layouts of fully connected generalized networks LitAlert P2018-51-26	Dec. 17, 2018	Lit Alert		—
—	3. INTEGRATED CIRCUIT DEVICE COMPRISES SUB-INTEGRATED CIRCUIT BLOCKS HAVING INLET LINKS AND OUTLET LINKS, AND ROUTING NETWORK INTERCONNECTING OUTLET LINKS AND INLET LINKS OF SUB-INTEGRATED CIRCUIT BLOCKS Out Of Plan DWPI 2008-O21116	May 25, 2007	DWPI	—	—
—	4. RF 029513/0134 Out Of Plan	Dec. 20, 2012	Assignments	—	—
—	5. PatStat 8269523	2019	Patent Status Files	—	—
—	6. PatStat 8269523	2019	Patent Status Files	—	—
—	7. Konda Technologies, Inc. v. Flex Logix Technologies, Inc.	Dec. 18, 2018	Docket Summaries	—	—
—	8. Konda Technologies, Inc. v. Flex Logix Technologies, Inc.	Dec. 17, 2018	Docket Summaries	—	—
—	9. VLSI LAYOUTS OF FULLY CONNECTED GENERALIZED AND PYRAMID NETWORKS WITH LOCALITY EXPLOITATION Out Of Plan US PAT 10050904+ , U.S. PTO Utility	Aug. 14, 2018	Patents	—	—
—	10. OPTIMIZATION OF MULTI-STAGE HIERARCHICAL NETWORKS FOR PRACTICAL ROUTING APPLICATIONS Out Of Plan US PAT 10003553+ , U.S. PTO Utility	June 19, 2018	Patents	—	—
—	11. FAST SCHEDULING AND OPTIMIZATION OF MULTI-STAGE HIERARCHICAL NETWORKS Out Of Plan US PAT 9929977+ , U.S. PTO Utility	Mar. 27, 2018	Patents	—	—
—	12. INTEGRATED CIRCUIT INCLUDING AN ARRAY OF LOGIC TILES, EACH LOGIC TILE INCLUDING A CONFIGURABLE SWITCH INTERCONNECT NETWORK Out Of Plan US PAT 9906225 , U.S. PTO Utility	Feb. 27, 2018	Patents	—	—

List of 20 Citing References for VLSI LAYOUTS OF FULLY CONNECTE...

Treatment	Title	Date	Type	Depth	Headnote(s)
—	13. SYSTEMS AND METHODS FOR SWITCHING USING HIERARCHICAL NETWORKS Out Of Plan US PAT 9817933 , U.S. PTO Utility	Nov. 14, 2017	Patents	—	—
—	14. MIXED-RADIX AND/OR MIXED-MODE SWITCH MATRIX ARCHITECTURE AND INTEGRATED CIRCUIT, AND METHOD OF OPERATING SAME Out Of Plan US PAT 9793898 , U.S. PTO Utility	Oct. 17, 2017	Patents	—	—
—	15. VLSI LAYOUTS OF FULLY CONNECTED GENERALIZED AND PYRAMID NETWORKS WITH LOCALITY EXPLOITATION Out Of Plan US PAT 9529958+ , U.S. PTO Utility	Dec. 27, 2016	Patents	—	—
—	16. FAST SCHEDULING AND OPTMIZATION OF MULTI- STAGE HIERARCHICAL NETWORKS Out Of Plan US PAT 9509634+ , U.S. PTO Utility	Nov. 29, 2016	Patents	—	—
—	17. MIXED-RADIX AND/OR MIXED-MODE SWITCH MATRIX ARCHITECTURE AND INTEGRATED CIRCUIT, AND METHOD OF OPERATING SAME Out Of Plan US PAT 9503092 , U.S. PTO Utility	Nov. 22, 2016	Patents	—	—
—	18. OPTIMIZATION OF MULTI-STAGE HIERARCHICAL NETWORKS FOR PRACTICAL ROUTING APPLICATIONS Out Of Plan US PAT 9374322+ , U.S. PTO Utility	June 21, 2016	Patents	—	—
—	19. VLSI LAYOUTS OF FULLY CONNECTED GENERALIZED AND PYRAMID NETWORKS WITH LOCALITY EXPLOITATION Out Of Plan US PAT 8898611 , U.S. PTO Utility	Nov. 25, 2014	Patents	—	—
—	20. VLSI LAYOUTS OF FULLY CONNECTED GENERALIZED AND PYRAMID NETWORKS WITH LOCALITY EXPLOITATION Out Of Plan US PAT APP 20190036844 , U.S. PTO Application	Jan. 31, 2019	Patents	—	—

PATENT APPLICATION FEE DETERMINATION RECORD Substitute for Form PTO-875	Application or Docket Number 16/202,067

APPLICATION AS FILED - PART I

	(Column 1) NUMBER FILED	(Column 2) NUMBER EXTRA	SMALL ENTITY		OR	OTHER THAN SMALL ENTITY	
FOR	NUMBER FILED	NUMBER EXTRA	RATE($)	FEE($)		RATE($)	FEE($)
BASIC FEE (37 CFR 1.16(a), (b), or (c))	N/A	N/A	N/A	150		N/A	
SEARCH FEE (37 CFR 1.16(k), (i), or (m))	N/A	N/A	N/A	330		N/A	
EXAMINATION FEE (37 CFR 1.16(o), (p), or (q))	N/A	N/A	N/A	1100		N/A	
TOTAL CLAIMS (37 CFR 1.16(i))	20 minus 20 =	*	x 50 =	0.00	OR		
INDEPENDENT CLAIMS (37 CFR 1.16(h))	1 minus 3 =	*	x 230 =	0.00			
APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is $310 ($155 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).			0.00			
MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))				0.00			
* If the difference in column 1 is less than zero, enter "0" in column 2.			TOTAL	1580		TOTAL	

APPLICATION AS AMENDED - PART II

		(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	SMALL ENTITY		OR	OTHER THAN SMALL ENTITY	
						RATE($)	ADDITIONAL FEE($)		RATE($)	ADDITIONAL FEE($)
AMENDMENT A	Total (37 CFR 1.16(i))	*	Minus	**	=	x =		OR	x =	
	Independent (37 CFR 1.16(h))	*	Minus	***	=	x =		OR	x =	
	Application Size Fee (37 CFR 1.16(s))									
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
						TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	

		(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	RATE($)	ADDITIONAL FEE($)	OR	RATE($)	ADDITIONAL FEE($)
AMENDMENT B	Total (37 CFR 1.16(i))	*	Minus	**	=	x =		OR	x =	
	Independent (37 CFR 1.16(h))	*	Minus	***	=	x =		OR	x =	
	Application Size Fee (37 CFR 1.16(s))									
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
						TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.
** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".
*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".
The "Highest Number Previously Paid For" (Total or Independent) is the highest found in the appropriate box in column 1.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING or 371(c) DATE	GRP ART UNIT	FIL FEE REC'D	ATTY.DOCKET.NO	TOT CLAIMS	IND CLAIMS
16/202,067	11/27/2018	2819	1660	V-0070US	20	1

CONFIRMATION NO. 8134

38139
Konda Technologies, Inc
6278 GRAND OAK WAY
SAN JOSE, CA 95135

UPDATED FILING RECEIPT

Date Mailed: 02/21/2019

Receipt is acknowledged of this reissue patent application. The application will be taken up for examination in due course. Applicant will be notified as to the results of the examination. Any correspondence concerning the application must include the following identification information: the U.S. APPLICATION NUMBER, FILING DATE, NAME OF APPLICANT, and TITLE OF INVENTION. Fees transmitted by check or draft are subject to collection. Please verify the accuracy of the data presented on this receipt. **If an error is noted on this Filing Receipt, please submit a written request for a Filing Receipt Correction. Please provide a copy of this Filing Receipt with the changes noted thereon. If you received a "Notice to File Missing Parts" for this application, please submit any corrections to this Filing Receipt with your reply to the Notice. When the USPTO processes the reply to the Notice, the USPTO will generate another Filing Receipt incorporating the requested corrections**

Inventor(s)
> Venkat Konda, San Jose, CA;

Applicant(s)
> Konda Technologies Inc., San Jose, CA, Assignee (with 37 CFR 1.172 Interest);

Assignment For Published Patent Application
> Konda Technologies Inc., San Jose, CA

Power of Attorney: The patent practitioners associated with Customer Number 38139

Domestic Priority data as claimed by applicant
> This application is a REI of 12/601,275 05/31/2010 PAT 8269523
> which is a 371 of PCT/US08/64605 05/22/2008
> which claims benefit of 60/940,394 05/25/2007

Foreign Applications for which priority is claimed (You may be eligible to benefit from the **Patent Prosecution Highway** program at the USPTO. Please see http://www.uspto.gov for more information.) - None.
Foreign application information must be provided in an Application Data Sheet in order to constitute a claim to foreign priority. See 37 CFR 1.55 and 1.76.

Permission to Access Application via Priority Document Exchange: Yes

Permission to Access Search Results: Yes

Applicant may provide or rescind an authorization for access using Form PTO/SB/39 or Form PTO/SB/69 as appropriate.

If Required, Foreign Filing License Granted: 12/04/2018

The country code and number of your priority application, to be used for filing abroad under the Paris Convention, is **US 16/202,067**

Projected Publication Date: None, application is not eligible for pre-grant publication

Non-Publication Request: No

Early Publication Request: No
** SMALL ENTITY **
Title

> VLSI Layouts of Fully Connected Generalized Networks

Preliminary Class

> 326

Statement under 37 CFR 1.55 or 1.78 for AIA (First Inventor to File) Transition Applications: No

PROTECTING YOUR INVENTION OUTSIDE THE UNITED STATES

Since the rights granted by a U.S. patent extend only throughout the territory of the United States and have no effect in a foreign country, an inventor who wishes patent protection in another country must apply for a patent in a specific country or in regional patent offices. Applicants may wish to consider the filing of an international application under the Patent Cooperation Treaty (PCT). An international (PCT) application generally has the same effect as a regular national patent application in each PCT-member country. The PCT process **simplifies** the filing of patent applications on the same invention in member countries, but **does not result** in a grant of "an international patent" and does not eliminate the need of applicants to file additional documents and fees in countries where patent protection is desired.

Almost every country has its own patent law, and a person desiring a patent in a particular country must make an application for patent in that country in accordance with its particular laws. Since the laws of many countries differ in various respects from the patent law of the United States, applicants are advised to seek guidance from specific foreign countries to ensure that patent rights are not lost prematurely.

Applicants also are advised that in the case of inventions made in the United States, the Director of the USPTO must issue a license before applicants can apply for a patent in a foreign country. The filing of a U.S. patent application serves as a request for a foreign filing license. The application's filing receipt contains further information and guidance as to the status of applicant's license for foreign filing.

Applicants may wish to consult the USPTO booklet, "General Information Concerning Patents" (specifically, the section entitled "Treaties and Foreign Patents") for more information on timeframes and deadlines for filing foreign patent applications. The guide is available either by contacting the USPTO Contact Center at 800-786-9199, or it can be viewed on the USPTO website at http://www.uspto.gov/web/offices/pac/doc/general/index.html.

For information on preventing theft of your intellectual property (patents, trademarks and copyrights), you may wish to consult the U.S. Government website, http://www.stopfakes.gov. Part of a Department of Commerce initiative, this website includes self-help "toolkits" giving innovators guidance on how to protect intellectual property in specific countries such as China, Korea and Mexico. For questions regarding patent enforcement issues, applicants may call the U.S. Government hotline at 1-866-999-HALT (1-866-999-4258).

LICENSE FOR FOREIGN FILING UNDER

Title 35, United States Code, Section 184

Title 37, Code of Federal Regulations, 5.11 & 5.15

GRANTED

The applicant has been granted a license under 35 U.S.C. 184, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" followed by a date appears on this form. Such licenses are issued in all applications where the conditions for issuance of a license have been met, regardless of whether or not a license may be required as set forth in 37 CFR 5.15. The scope and limitations of this license are set forth in 37 CFR 5.15(a) unless an earlier license has been issued under 37 CFR 5.15(b). The license is subject to revocation upon written notification. The date indicated is the effective date of the license, unless an earlier license of similar scope has been granted under 37 CFR 5.13 or 5.14.

This license is to be retained by the licensee and may be used at any time on or after the effective date thereof unless it is revoked. This license is automatically transferred to any related applications(s) filed under 37 CFR 1.53(d). This license is not retroactive.

The grant of a license does not in any way lessen the responsibility of a licensee for the security of the subject matter as imposed by any Government contract or the provisions of existing laws relating to espionage and the national security or the export of technical data. Licensees should apprise themselves of current regulations especially with respect to certain countries, of other agencies, particularly the Office of Defense Trade Controls, Department of State (with respect to Arms, Munitions and Implements of War (22 CFR 121-128)); the Bureau of Industry and Security, Department of Commerce (15 CFR parts 730-774); the Office of Foreign AssetsControl, Department of Treasury (31 CFR Parts 500+) and the Department of Energy.

NOT GRANTED

No license under 35 U.S.C. 184 has been granted at this time, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" DOES NOT appear on this form. Applicant may still petition for a license under 37 CFR 5.12, if a license is desired before the expiration of 6 months from the filing date of the application. If 6 months has lapsed from the filing date of this application and the licensee has not received any indication of a secrecy order under 35 U.S.C. 181, the licensee may foreign file the application pursuant to 37 CFR 5.15(b).

SelectUSA

The United States represents the largest, most dynamic marketplace in the world and is an unparalleled location for business investment, innovation, and commercialization of new technologies. The U.S. offers tremendous resources and advantages for those who invest and manufacture goods here. Through SelectUSA, our nation works to promote and facilitate business investment. SelectUSA provides information assistance to the international investor community; serves as an ombudsman for existing and potential investors; advocates on behalf of U.S. cities, states, and regions competing for global investment; and counsels U.S. economic development organizations on investment attraction best practices. To learn more about why the United States is the best country in the world to develop technology, manufacture products, deliver services, and grow your business, visit http://www.SelectUSA.gov or call +1-202-482-6800.

To: venkat@kondatech.com,vkonda@gmail.com,
From: PAIR_eOfficeAction@uspto.gov
Cc: PAIR_eOfficeAction@uspto.gov
Subject: Private PAIR Correspondence Notification for Customer Number 38139

Feb 21, 2019 03:27:29 AM

Dear PAIR Customer:

Konda Technologies, Inc
6278 GRAND OAK WAY
SAN JOSE, CA 95135
UNITED STATES

The following USPTO patent application(s) associated with your Customer Number, 38139 , have
new outgoing correspondence. This correspondence is now available for viewing in Private PAIR.

The official date of notification of the outgoing correspondence will be indicated on the form PTOL-90
accompanying the correspondence.

Disclaimer:
The list of documents shown below is provided as a courtesy and is not part of the official file
wrapper. The content of the images shown in PAIR is the official record.

Application	Document	Mailroom Date	Attorney Docket No.
16202067	APP.FILE.REC	02/21/2019	V-0070US

To view your correspondence online or update your email addresses, please visit us anytime at
https://sportal.uspto.gov/secure/myportal/privatepair.

If you have any questions, please email the Electronic Business Center (EBC) at EBC@uspto.gov
with 'e-Office Action' on the subject line or call 1-866-217-9197 during the following hours:

    Monday - Friday 6:00 a.m. to 12:00 a.m.

Thank you for prompt attention to this notice,

UNITED STATES PATENT AND TRADEMARK OFFICE

PATENT APPLICATION INFORMATION RETRIEVAL SYSTEM

Electronic Patent Application Fee Transmittal

Application Number:	16202067
Filing Date:	27-Nov-2018
Title of Invention:	VLSI Layouts of Fully Connected Generalized Networks
First Named Inventor/Applicant Name:	Venkat Konda
Filer:	Venkat Konda
Attorney Docket Number:	V-0070US

Filed as Small Entity

Filing Fees for   Utility under 35 USC 111(a)

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
REISSUE OR REISSUE DESIGN CPA SEARCH FEE	2114	1	330	330
REISSUE OR REISSUE DESIGN CPA EXAM. FEE	2314	1	1100	1100
Pages:				
Claims:				
Miscellaneous-Filing:				
LATE FILING FEE FOR OATH OR DECLARATION	2051	1	80	80
Petition:				
Patent-Appeals-and-Interference:				

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				
Miscellaneous:				
			Total in USD ($)	**1510**

Electronic Acknowledgement Receipt

EFS ID:	35042525
Application Number:	16202067
International Application Number:	
Confirmation Number:	8134
Title of Invention:	VLSI Layouts of Fully Connected Generalized Networks
First Named Inventor/Applicant Name:	Venkat Konda
Customer Number:	38139
Filer:	Venkat Konda
Filer Authorized By:	
Attorney Docket Number:	V-0070US
Receipt Date:	03-FEB-2019
Filing Date:	27-NOV-2018
Time Stamp:	17:16:05
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Fee Worksheet (SB06)	fee-info.pdf	35302 22e18476d2ee7c06c7350fc163ddac7f778405c7	no	2

Warnings:

Information:		
	Total Files Size (in bytes):	35302

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.
National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.
New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

Electronic Patent Application Fee Transmittal

Application Number:	16202067
Filing Date:	27-Nov-2018
Title of Invention:	VLSI Layouts of Fully Connected Generalized Networks
First Named Inventor/Applicant Name:	Venkat Konda
Filer:	Venkat Konda
Attorney Docket Number:	V-0070US

Filed as Small Entity

Filing Fees for   Utility under 35 USC 111(a)

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
REISSUE OR REISSUE DESIGN CPA SEARCH FEE	2114	1	330	330
REISSUE OR REISSUE DESIGN CPA EXAM. FEE	2314	1	1100	1100
Pages:				
Claims:				
Miscellaneous-Filing:				
LATE FILING FEE FOR OATH OR DECLARATION	2051	1	80	80
Petition:				
Patent-Appeals-and-Interference:				

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				
Miscellaneous:				
			Total in USD ($)	1510

Electronic Acknowledgement Receipt

EFS ID:	35042595
Application Number:	16202067
International Application Number:	
Confirmation Number:	8134
Title of Invention:	VLSI Layouts of Fully Connected Generalized Networks
First Named Inventor/Applicant Name:	Venkat Konda
Customer Number:	38139
Filer:	Venkat Konda
Filer Authorized By:	
Attorney Docket Number:	V-0070US
Receipt Date:	03-FEB-2019
Filing Date:	27-NOV-2018
Time Stamp:	19:47:11
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	yes
Payment Type	CARD
Payment was successfully received in RAM	$ 1510
RAM confirmation Number	020419INTEFSW19533800
Deposit Account	
Authorized User	
The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows:	

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Fee Worksheet (SB06)	fee-info.pdf	35302 2d3b9f4c59d00197e2b9ec8f78779ff32166d8b0	no	2

Warnings:

Information:

	Total Files Size (in bytes):	35302

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.
National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.
New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

Electronic Patent Application Fee Transmittal

Application Number:	16202067
Filing Date:	27-Nov-2018
Title of Invention:	VLSI Layouts of Fully Connected Generalized Networks
First Named Inventor/Applicant Name:	Venkat Konda
Filer:	Venkat Konda
Attorney Docket Number:	V-0070US

Filed as Small Entity

Filing Fees for   Utility under 35 USC 111(a)

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
REISSUE OR REISSUE DESIGN CPA SEARCH FEE	2114	1	330	330
REISSUE OR REISSUE DESIGN CPA EXAM. FEE	2314	1	1100	1100
Pages:				
Claims:				
Miscellaneous-Filing:				
LATE FILING FEE FOR OATH OR DECLARATION	2051	1	80	80
Petition:				
Patent-Appeals-and-Interference:				

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				
Miscellaneous:				
			Total in USD ($)	1510

Electronic Acknowledgement Receipt

EFS ID:	35042585
Application Number:	16202067
International Application Number:	
Confirmation Number:	8134
Title of Invention:	VLSI Layouts of Fully Connected Generalized Networks
First Named Inventor/Applicant Name:	Venkat Konda
Customer Number:	38139
Filer:	Venkat Konda
Filer Authorized By:	
Attorney Docket Number:	V-0070US
Receipt Date:	03-FEB-2019
Filing Date:	27-NOV-2018
Time Stamp:	19:26:58
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Fee Worksheet (SB06)	fee-info.pdf	35302 aee7a4c70e6a2ce46e279515d699cc48e70c9820	no	2

Warnings:

Information:		
	Total Files Size (in bytes):	35302

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.
National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.
New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371(C) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
16/202,067	11/27/2018	Venkat Konda	V-0070US

CONFIRMATION NO. 8134

38139
Konda Technologies, Inc
6278 GRAND OAK WAY
SAN JOSE, CA 95135

FORMALITIES LETTER

OC000000104258959

Date Mailed: 12/06/2018

NOTICE TO FILE MISSING PARTS OF REISSUE APPLICATION

Filing Date Granted

An application number and filing date have been accorded to this reissue application. The item(s) indicated below, however, are missing. Applicant is given TWO MONTHS from the date of this Notice within which to file all required items and pay any fees required below to avoid abandonment. Extensions of time may be obtained by filing a petition accompanied by the extension fee under the provisions of 37 CFR 1.136(a).

- The application search fee must be submitted.
- The application examination fee must be submitted.
- Surcharge as set forth in 37 CFR 1.16(f) must be submitted.
  The surcharge is due for any one of:
  - late submission of the basic filing fee, search fee, or examination fee,
  - late submission of inventor's oath or declaration,
  - filing an application that does not contain at least one claim on filing, or
  - submission of an application filed by reference to a previously filed application.

SUMMARY OF FEES DUE:

The fee(s) required within **TWO MONTHS** from the date of this Notice to avoid abandonment is/are itemized below. Small entity discount is in effect. If applicant is qualified for micro entity status, an acceptable Certification of Micro Entity Status must be submitted to establish micro entity status. (See 37 CFR 1.29 and forms PTO/SB/15A and 15B.)
- $ **80** surcharge.
- $ **330** search fee.
- $ **1100** examination fee.
- $(**0**) previous unapplied payment amount.
- $ **1510** TOTAL FEE BALANCE DUE.

Replies must be received in the USPTO within the set time period or must include a proper Certificate of Mailing or Transmission under 37 CFR 1.8 with a mailing or transmission date within the set time period. For more information and a suggested format, see Form PTO/SB/92 and MPEP 512.

Replies should be mailed to:

Mail Stop Missing Parts
Commissioner for Patents
P.O. Box 1450
Alexandria VA 22313-1450

Registered users of EFS-Web may alternatively submit their reply to this notice via EFS-Web, including a copy of this Notice and selecting the document description "Applicant response to Pre-Exam Formalities Notice". https://sportal.uspto.gov/authenticate/AuthenticateUserLocalEPF.html

For more information about EFS-Web please call the USPTO Electronic Business Center at 1-866-217-9197 or visit our website at http://www.uspto.gov/ebc.

If you are not using EFS-Web to submit your reply, you must include a copy of this notice.

Questions about the contents of this notice and the
requirements it sets forth should be directed to the Office
of Data Management, Application Assistance Unit, at
(571) 272-4000 or **(571) 272-4200** or **1-888-786-0101**.

/cmhaywood/

PATENT APPLICATION FEE DETERMINATION RECORD	Application or Docket Number
Substitute for Form PTO-875	16/202,067

APPLICATION AS FILED - PART I

	(Column 1)	(Column 2)	SMALL ENTITY		OR	OTHER THAN SMALL ENTITY	
FOR	NUMBER FILED	NUMBER EXTRA	RATE($)	FEE($)		RATE($)	FEE($)
BASIC FEE (37 CFR 1.16(a), (b), or (c))	N/A	N/A	N/A	150		N/A	
SEARCH FEE (37 CFR 1.16(k), (i), or (m))	N/A	N/A	N/A	330		N/A	
EXAMINATION FEE (37 CFR 1.16(o), (p), or (q))	N/A	N/A	N/A	1100		N/A	
TOTAL CLAIMS (37 CFR 1.16(i))	20 minus 20 =	*	x 50 =	0.00	OR		
INDEPENDENT CLAIMS (37 CFR 1.16(h))	1 minus 3 =	*	x 230 =	0.00			
APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is $310 ($155 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).			0.00			
MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))				0.00			
* If the difference in column 1 is less than zero, enter "0" in column 2.			TOTAL	1580		TOTAL	

APPLICATION AS AMENDED - PART II

AMENDMENT A

		(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	SMALL ENTITY RATE($)	ADDITIONAL FEE($)	OR	OTHER THAN SMALL ENTITY RATE($)	ADDITIONAL FEE($)
	Total (37 CFR 1.16(i))	*	Minus	**	=	x =		OR	x =	
	Independent (37 CFR 1.16(h))	*	Minus	***	=	x =		OR	x =	
	Application Size Fee (37 CFR 1.16(s))									
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
						TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	

AMENDMENT B

		(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	SMALL ENTITY RATE($)	ADDITIONAL FEE($)	OR	OTHER THAN SMALL ENTITY RATE($)	ADDITIONAL FEE($)
	Total (37 CFR 1.16(i))	*	Minus	**	=	x =		OR	x =	
	Independent (37 CFR 1.16(h))	*	Minus	***	=	x =		OR	x =	
	Application Size Fee (37 CFR 1.16(s))									
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
						TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	

 * If the entry in column 1 is less than the entry in column 2, write "0" in column 3.
 ** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".
 *** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".
    The "Highest Number Previously Paid For" (Total or Independent) is the highest found in the appropriate box in column 1.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING or 371(c) DATE	GRP ART UNIT	FIL FEE REC'D	ATTY.DOCKET.NO	TOT CLAIMS	IND CLAIMS
16/202,067	11/27/2018	2819	150	V-0070US	20	1

CONFIRMATION NO. 8134

38139
Konda Technologies, Inc
6278 GRAND OAK WAY
SAN JOSE, CA 95135

FILING RECEIPT

OC000000104258958

Date Mailed: 12/06/2018

Receipt is acknowledged of this reissue patent application. The application will be taken up for examination in due course. Applicant will be notified as to the results of the examination. Any correspondence concerning the application must include the following identification information: the U.S. APPLICATION NUMBER, FILING DATE, NAME OF APPLICANT, and TITLE OF INVENTION. Fees transmitted by check or draft are subject to collection. Please verify the accuracy of the data presented on this receipt. **If an error is noted on this Filing Receipt, please submit a written request for a Filing Receipt Correction. Please provide a copy of this Filing Receipt with the changes noted thereon. If you received a "Notice to File Missing Parts" for this application, please submit any corrections to this Filing Receipt with your reply to the Notice. When the USPTO processes the reply to the Notice, the USPTO will generate another Filing Receipt incorporating the requested corrections**

Inventor(s)
Venkat Konda, San Jose, CA;

Applicant(s)
Konda Technologies Inc., San Jose, CA, Assignee (with 37 CFR 1.172 Interest);

Assignment For Published Patent Application
Konda Technologies Inc., San Jose, CA

Power of Attorney: The patent practitioners associated with Customer Number 38139

Domestic Priority data as claimed by applicant
This application is a REI of 12/601,275 05/31/2010 PAT 8269523
which is a 371 of PCT/US08/64605 05/22/2008
which claims benefit of 60/940,394 05/25/2007

Foreign Applications for which priority is claimed (You may be eligible to benefit from the **Patent Prosecution Highway** program at the USPTO. Please see http://www.uspto.gov for more information.) - None.
Foreign application information must be provided in an Application Data Sheet in order to constitute a claim to foreign priority. See 37 CFR 1.55 and 1.76.

Permission to Access Application via Priority Document Exchange: Yes

Permission to Access Search Results: Yes

Applicant may provide or rescind an authorization for access using Form PTO/SB/39 or Form PTO/SB/69 as appropriate.

If Required, Foreign Filing License Granted: 12/04/2018

The country code and number of your priority application, to be used for filing abroad under the Paris Convention, is **US 16/202,067**

Projected Publication Date: None, application is not eligible for pre-grant publication

Non-Publication Request: No

Early Publication Request: No
** SMALL ENTITY **
Title

> VLSI Layouts of Fully Connected Generalized Networks

Preliminary Class

> 326

Statement under 37 CFR 1.55 or 1.78 for AIA (First Inventor to File) Transition Applications: No

PROTECTING YOUR INVENTION OUTSIDE THE UNITED STATES

Since the rights granted by a U.S. patent extend only throughout the territory of the United States and have no effect in a foreign country, an inventor who wishes patent protection in another country must apply for a patent in a specific country or in regional patent offices. Applicants may wish to consider the filing of an international application under the Patent Cooperation Treaty (PCT). An international (PCT) application generally has the same effect as a regular national patent application in each PCT-member country. The PCT process **simplifies** the filing of patent applications on the same invention in member countries, but **does not result** in a grant of "an international patent" and does not eliminate the need of applicants to file additional documents and fees in countries where patent protection is desired.

Almost every country has its own patent law, and a person desiring a patent in a particular country must make an application for patent in that country in accordance with its particular laws. Since the laws of many countries differ in various respects from the patent law of the United States, applicants are advised to seek guidance from specific foreign countries to ensure that patent rights are not lost prematurely.

Applicants also are advised that in the case of inventions made in the United States, the Director of the USPTO must issue a license before applicants can apply for a patent in a foreign country. The filing of a U.S. patent application serves as a request for a foreign filing license. The application's filing receipt contains further information and guidance as to the status of applicant's license for foreign filing.

Applicants may wish to consult the USPTO booklet, "General Information Concerning Patents" (specifically, the section entitled "Treaties and Foreign Patents") for more information on timeframes and deadlines for filing foreign patent applications. The guide is available either by contacting the USPTO Contact Center at 800-786-9199, or it can be viewed on the USPTO website at http://www.uspto.gov/web/offices/pac/doc/general/index.html.

For information on preventing theft of your intellectual property (patents, trademarks and copyrights), you may wish to consult the U.S. Government website, http://www.stopfakes.gov. Part of a Department of Commerce initiative, this website includes self-help "toolkits" giving innovators guidance on how to protect intellectual property in specific countries such as China, Korea and Mexico. For questions regarding patent enforcement issues, applicants may call the U.S. Government hotline at 1-866-999-HALT (1-866-999-4258).

LICENSE FOR FOREIGN FILING UNDER

Title 35, United States Code, Section 184

Title 37, Code of Federal Regulations, 5.11 & 5.15

GRANTED

The applicant has been granted a license under 35 U.S.C. 184, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" followed by a date appears on this form. Such licenses are issued in all applications where the conditions for issuance of a license have been met, regardless of whether or not a license may be required as set forth in 37 CFR 5.15. The scope and limitations of this license are set forth in 37 CFR 5.15(a) unless an earlier license has been issued under 37 CFR 5.15(b). The license is subject to revocation upon written notification. The date indicated is the effective date of the license, unless an earlier license of similar scope has been granted under 37 CFR 5.13 or 5.14.

This license is to be retained by the licensee and may be used at any time on or after the effective date thereof unless it is revoked. This license is automatically transferred to any related applications(s) filed under 37 CFR 1.53(d). This license is not retroactive.

The grant of a license does not in any way lessen the responsibility of a licensee for the security of the subject matter as imposed by any Government contract or the provisions of existing laws relating to espionage and the national security or the export of technical data. Licensees should apprise themselves of current regulations especially with respect to certain countries, of other agencies, particularly the Office of Defense Trade Controls, Department of State (with respect to Arms, Munitions and Implements of War (22 CFR 121-128)); the Bureau of Industry and Security, Department of Commerce (15 CFR parts 730-774); the Office of Foreign AssetsControl, Department of Treasury (31 CFR Parts 500+) and the Department of Energy.

NOT GRANTED

No license under 35 U.S.C. 184 has been granted at this time, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" DOES NOT appear on this form. Applicant may still petition for a license under 37 CFR 5.12, if a license is desired before the expiration of 6 months from the filing date of the application. If 6 months has lapsed from the filing date of this application and the licensee has not received any indication of a secrecy order under 35 U.S.C. 181, the licensee may foreign file the application pursuant to 37 CFR 5.15(b).

SelectUSA

The United States represents the largest, most dynamic marketplace in the world and is an unparalleled location for business investment, innovation, and commercialization of new technologies. The U.S. offers tremendous resources and advantages for those who invest and manufacture goods here. Through SelectUSA, our nation works to promote and facilitate business investment. SelectUSA provides information assistance to the international investor community; serves as an ombudsman for existing and potential investors; advocates on behalf of U.S. cities, states, and regions competing for global investment; and counsels U.S. economic development organizations on investment attraction best practices. To learn more about why the United States is the best country in the world to develop technology, manufacture products, deliver services, and grow your business, visit http://www.SelectUSA.gov or call +1-202-482-6800.

MULTIPLE DEPENDENT CLAIM
FEE CALCULATION SHEET

Substitute for Form PTO-1360
(For use with Form PTO/SB/06)

Application Number 16202067

Filing Date

Applicant(s) Venkat Konda

* May be used for additional claims or amendments

CLAIMS	AS FILED		AFTER FIRST AMENDMENT		AFTER SECOND AMENDMENT				*		*		*	
	Indep	Depend	Indep	Depend	Indep	Depend			Indep	Depend	Indep	Depend	Indep	Depend
1	1						51							
2		1					52							
3		1					53							
4		1					54							
5		1					55							
6		1					56							
7		1					57							
8		1					58							
9		1					59							
10		1					60							
11		1					61							
12		1					62							
13		1					63							
14		1					64							
15		1					65							
16		1					66							
17		1					67							
18		1					68							
19		1					69							
20		1					70							
21							71							
22							72							
23							73							
24							74							
25							75							
26							76							
27							77							
28							78							
29							79							
30							80							
31							81							
32							82							
33							83							
34							84							
35							85							
36							86							
37							87							
38							88							
39							89							
40							90							
41							91							
42							92							
43							93							
44							94							
45							95							
46							96							
47							97							
48							98							
49							99							
50							100							
Total Indep	1		0		0									
Total Depend	19	↵	0	↵	0	↵								
Total Claims	20		0		0									

CORRECTED ADS FORM

Application Number	16202067
Title of Invention	VLSI Layouts of Fully Connected Generalized Networks

Inventor Information

If no data is shown, no data has been corrected

	Data of Record	Updated Data
Order Number		
Name		

Residence Information

Residency		
City		
State		
Country of Residence		

Mailing Address of Inventor

Address 1		
Address 2		
City,State/Province, Postal Code		
Country		

Corrected ADS 1.0

Document Description: Application Data Sheet to update/correct info
Doc Code: ADS.CORR

Application Information

	Data of Record	Updated Data
Title of Invention	VLSI Layouts of Fully Connected Generalized Networks	
Attorney Docket Number	V-0070US	
Entity Type	Small	

Domestic Benefit/National Stage Information

If no data is shown, no data has been corrected

This section allows for the applicant to either claim benefit under 35 U.S.C. 119(e), 120, 121,365(c), or 386(c) or indicate National Stage entry from a PCT application. Providing this information in the application data sheet constitutes the specific reference required by 35 U.S.C. 119(e) or 120, and 37 CFR 1.78(a).

		Data of Record	Updated Data
	Prior Application Status	~~patented~~	pending
	Application Number	~~16202067~~	PCT/US08/64605
	Continuity Type	~~REI~~	PRO
	Prior Application Number	~~12601275~~	60940394
	Filing Date (YYYY-MM-DD)	~~2010-05-31~~	2007-05-25
	Patent Number	~~8269523~~	
	Issue Date (YYYY-MM-DD)	~~2012-09-18~~	0001-01-01

Corrected ADS 1.0

Document Description: Application Data Sheet to update/correct info
Doc Code: ADS.CORR

		Data of Record	Updated Data
	Prior Application Status		
	Application Number	~~12601275~~	16202067
	Continuity Type	~~NST~~	CON
	Prior Application Number	PCT/US08/64605	
	Filing Date (YYYY-MM-DD)	2008-05-22	
	Patent Number		
	Issue Date (YYYY-MM-DD)		

		Data of Record	Updated Data
	Prior Application Status	~~pending~~	
	Application Number	~~PCT/US08/64605~~	12601275
	Continuity Type	~~PRO~~	NST
	Prior Application Number	~~60940394~~	PCT/US08/64605
	Filing Date (YYYY-MM-DD)	~~2007-05-25~~	2008-05-22
	Patent Number		
	Issue Date (YYYY-MM-DD)	~~0001-01-01~~	

		Data of Record	Updated Data
	Prior Application Status		patented
	Application Number		16202067
	Continuity Type		REI
	Prior Application Number		12601275
	Filing Date (YYYY-MM-DD)		2010-05-31
	Patent Number		8269523
	Issue Date (YYYY-MM-DD)		2012-09-18

Corrected ADS 1.0

Foreign Priority Information

If no data is shown, no data has been corrected

This section allows for the applicant to claim priority to a foreign application. Providing this information in the application data sheet constitutes the claim for priority as required by 35 U.S.C. 119(b) and 37 CFR 1.55. When priority is claimed to a foreign application that is eligible for retrieval under the priority document exchange program (PDX) the information will be used by the Office to automatically attempt retrieval pursuant to 37 CFR 1.55(i)(1) and (2). Under the PDX program, applicant bears the ultimate responsibility for ensuring that a copy of the foreign application is received by the Office from the participating foreign intellectual property office, or a certified copy of the foreign priority application is filed, within the time period specified in 37 CFR 1.55(g)(1).

		Data of Record	Updated Data
	Application Number		
	Country		
	Filing Date		
	Access Code		

Applicant Information

If no data is shown, no data has been corrected

Providing assignment information in this section does not substitute for compliance with any requirement of part 3 of Title 37 of CFR to have an assignment recorded by the Office.

	Data of Record	Updated Data
Applicant Type		
If applicant is the legal representative, indicate the authority to file the patent application, the inventor is		
Name of the Deceased or Legally Incapacitated Inventor		
Applicant is an Organization		
Name		
Organization Name		
Address 1		

Corrected ADS 1.0

Document Description: Application Data Sheet to update/correct info
Doc Code: ADS.CORR

Address 2		
City,State/Province,Postal Code		
Country		
Phone Number		
Fax Number		
Email Address		

Assignee Information including Non-Applicant Assignee Information

If no data is shown, no data has been corrected

Providing this information in the application data sheet does not substitute for compliance with any requirement of part 3 of Title 37 of the CFR to have an assignment recorded in the Office

	Data of Record	Updated Data
Order		
Applicant is an Organization		
Name		
Organization Name		

Mailing Address

Address 1		
Address 2		
City,State/Province,Postal Code		
Country		
Phone Number		
Fax Number		

Corrected ADS 1.0

Document Description: Application Data Sheet to update/correct info
Doc Code: ADS.CORR

Email Address		

Signature

NOTE: This Application Data Sheet must be signed in accordance with 37 CFR 1.33(b).

This Application Data Sheet **must** be signed by a patent practitioner if one or more of the applicants is a **juristic entity** (e.g., corporation or association). If the applicant is two or more joint inventors, this form must be signed by a patent practitioner, **all** joint inventors who are the applicant, or one or more joint inventor-applicants who have been given power of attorney (e.g., see USPTO Form PTO/AIA/81) on behalf of **all** joint inventor-applicants.

See 37 CFR 1.4(d) for the manner of making signatures and certifications.

Signature	/venkat Konda/	Registration Number	
First Name	venkat	Last Name	konda

Corrected ADS 1.0

Electronic Acknowledgement Receipt

EFS ID:	34502404
Application Number:	16202067
International Application Number:	
Confirmation Number:	8134
Title of Invention:	VLSI Layouts of Fully Connected Generalized Networks
First Named Inventor/Applicant Name:	
Customer Number:	38139
Filer:	Venkat Konda
Filer Authorized By:	
Attorney Docket Number:	V-0070US
Receipt Date:	06-DEC-2018
Filing Date:	27-NOV-2018
Time Stamp:	11:43:59
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Application Data Sheet to update/ correct info	CorrectedADS.pdf	83253 e15ad24bb86cc648475b582a3e98fc5626e27a9b	no	6

Warnings:

Information:		
	Total Files Size (in bytes):	83253

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.
National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.
New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

To: venkat@kondatech.com,vkonda@gmail.com,
From: PAIR_eOfficeAction@uspto.gov
Cc: PAIR_eOfficeAction@uspto.gov
Subject: Private PAIR Correspondence Notification for Customer Number 38139

Dec 06, 2018 03:30:10 AM

Dear PAIR Customer:

Konda Technologies, Inc
6278 GRAND OAK WAY
SAN JOSE, CA 95135
UNITED STATES

The following USPTO patent application(s) associated with your Customer Number, 38139 , have
new outgoing correspondence. This correspondence is now available for viewing in Private PAIR.

The official date of notification of the outgoing correspondence will be indicated on the form PTOL-90
accompanying the correspondence.

Disclaimer:
The list of documents shown below is provided as a courtesy and is not part of the official file
wrapper. The content of the images shown in PAIR is the official record.

Application	Document	Mailroom Date	Attorney Docket No.
16202067	NTC.MISS.PRT	12/06/2018	V-0070US
	APP.FILE.REC	12/06/2018	V-0070US

To view your correspondence online or update your email addresses, please visit us anytime at
https://sportal.uspto.gov/secure/myportal/privatepair.

If you have any questions, please email the Electronic Business Center (EBC) at EBC@uspto.gov
with 'e-Office Action' on the subject line or call 1-866-217-9197 during the following hours:

    Monday - Friday 6:00 a.m. to 12:00 a.m.

Thank you for prompt attention to this notice,

UNITED STATES PATENT AND TRADEMARK OFFICE

PATENT APPLICATION INFORMATION RETRIEVAL SYSTEM

Substitute for form 1449/PTO **INFORMATION DISCLOSURE STATEMENT BY APPLICANT** *(Use as many sheets as necessary)*	Complete if Known	
	Application Number	16/202067
	Filing Date	12-4-2018
	First Named Inventor	Venkat Konda
	Art Unit	
	Examiner Name	

Sheet	1	of	1	Attorney Docket Number	V-0070US

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
	1	C. Clos, "A Study of Non-Blocking Switching Networks," Bell System Technical Journal, 32:406-424, 1953.	
	2	A. DeHon, "Balancing Interconnect and Computation in a Reconfigurable Computing Array," ACM Int. Symp. on FPGA, pp. 69-78, Feb. 1999	
	3	Chihming Chang, Rami Melhem, "Arbitrary Size Benes Networks", Journal: Parallel Processing Letters - PPL , vol. 7, no. 3, pp. 279-284, 1997.	
	4	HODA EL-SAYED and ABDOU YOUSSEF; "The r-truncated Benes Networks and their Randomized Routing Algorithms"1997 Intl Conf on Parallel and Dist Sys, Seoul, Korea, December 1997.	
	5	Guy Lemieux and David Lewis, "Using Sparse Crossbars within LUT Clusters", Procds of the ACM/SIGDA Intl Symp on Field Prog Gate Arrays 2001, Feb. 11–13, 2001, Monterey, CA.	
	6	P. Manuel, W. K. Qureshi, A. William, A. Muthumalai, "VLSI layout of Benes networks,", J. of Discrete Math. Sci. & Cryptography, vol. 10, no, 4, pp. 461-472, 2007	
	7	Quinn, Michael J, "Parallel Computing: Theory and Practice", 2nd. ed., 1994, McGraw Hill Series in computer Science, Networks, and parallel computing, ISBN 0-07-051294-9	
	8	Ronald I. Greenberg, "The Fat-Pyramid and Universal Parallel Computation Independent of wire delay" IEEE Trans. Computers, 43(12):1358-1364, December 1994.	
	9	Hypertree: A Multiprocessor Interconnection Topology , by James R. Goodman and Carlo H Sequin, Computer Science Technical Report #427, Dept , of EECS, University of California	
	10	Data Movement Techniques for the pyramid computer, Russ Miller and Quentin F. Stout, SIAM Journal on Computing, Vol. 16, no. 1, pp. 38 - 60, Feb. 1987.	

Examiner Signature		Date Considered	

Privacy Act Statement

The **Privacy Act of 1974 (P.L. 93-579)** requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (*i.e.*, GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

Arbitrary Size Benes Networks

Chihming Chang and Rami Melhem
University of Pittsburgh
Department of Computer Science

Abstract

The Benes network is a rearrangeable nonblocking network which can realize any arbitrary permutation. Overall, the r-dimensional Benes network connects 2^r inputs to 2^r outputs through $2r - 1$ levels of 2×2 switches. Each level of switches consists of 2^{r-1} switches, and hence the size of the network has to be a power of two. In this paper, we extend Benes networks to arbitrary sizes. We also show that the looping routing algorithm used in Benes networks can be slightly modified and applied to arbitrary size Benes networks.

1   Introduction

A multistage network consists of more than one stage of switching elements and is usually capable of connecting an arbitrary input terminal to an arbitrary output terminal. Multistage networks are classified into blocking, rearrangeable, or nonblocking networks. In blocking networks, simultaneous connections of more than one terminal pair may result in conflicts in the use of network communication links. A network is a rearrangeable nonblocking network if it can realize all possible permutations between inputs and outputs. However, if the connections in a permutation are established in the network sequentially, the establishment of a connection may require rearranging the existing connections. A network which can handle all possible permutations without rearranging connections is a nonblocking network [3].

The Benes network [1], which is a special instance of CLOS networks [2], is an excellent example of a rearrangeable network. Overall, the r-dimensional Benes network has $2r - 1$ levels of switches, with 2^{r-1} switches in each level. Figure 1 shows the Benes network with $r = 3$. Given any one-to-one mapping, Π, of 2^r inputs to 2^r outputs, there is a set of edge-disjoint paths from the inputs of an r-dimensional Benes network to its outputs connecting input i to output $\Pi(i)$ for $0 \le i \le 2^r - 1$ [1, 4].

The Benes topology is specified such that the number of input or output terminals has to be a power of 2. In practical terms, this is a severe restriction on the sizes of systems that will use the network. If the size of the needed network is not a power of 2, a larger than needed network has to be used, and many of the resources in the used network will remain idle. In this paper, we present a constructive way of building an arbitrary size Benes network (AS-Benes) for any number of terminals. The routing algorithm presented for the AS-Benes is nearly as simple as the looping algorithm for the regular Benes network [5].

1
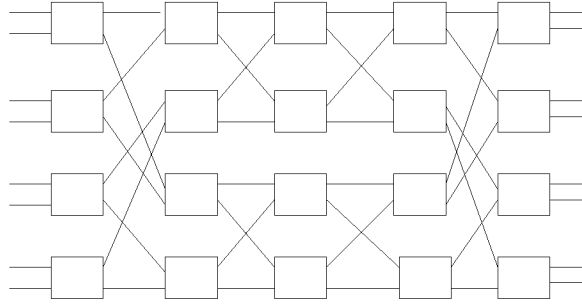
Figure 1: A 8 x 8 Benes network with $r = 3$
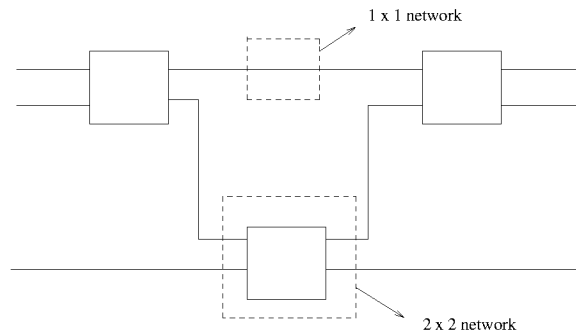
Figure 2: A 3 x 3 AS-Benes network

2   Construction Strategy

Multistage interconnection networks are usually constructed from a single type of modular switching elements. Each element is a 2×2 switch which can be set by a control line into a direct-connection state or a crossed-connection state, thus realizing all permutations from two inputs to two outputs. Three 2×2 switches can be used to construct a network which can relize any 3×3 permutation as shown in Figure 2. If we consider a simple wire to be a network that can realize any 1×1 permutation, we can view the 3×3 network in Figure 2 as being built from a 2×2 network and a 1×1 network.

The procedure used to construct a network of size 3 can be generalized to recursively construct a network of any size. Specifically, an AS-Benes of size n is constructed recursively from an AS-Benes of size $\lfloor \frac{n}{2} \rfloor$ and an AS-Benes of size $\lceil \frac{n}{2} \rceil$. When n is even, the construction is similar to that of the Benes network where the n inputs are connected to $\frac{n}{2}$ switches and each switch is connected to two AS-Benes networks of size $\frac{n}{2}$. Similarly, the n outputs are connected to $\frac{n}{2}$ switches and each switch is connected to the two $\frac{n}{2}$ AS-Benes networks (see Figure 3(a)).

To construct an AS-Benes of an odd size, the first $n-1$ inputs are connected to $\lfloor \frac{n}{2} \rfloor$ switches and each switch is connected to the AS-Benes of size $\lfloor \frac{n}{2} \rfloor$ and the AS-Benes of size $\lceil \frac{n}{2} \rceil$. Similarly, the first $n-1$ outputs are connected to $\lfloor \frac{n}{2} \rfloor$ switches and each switch is connected

2

(a) n even   (b) n odd
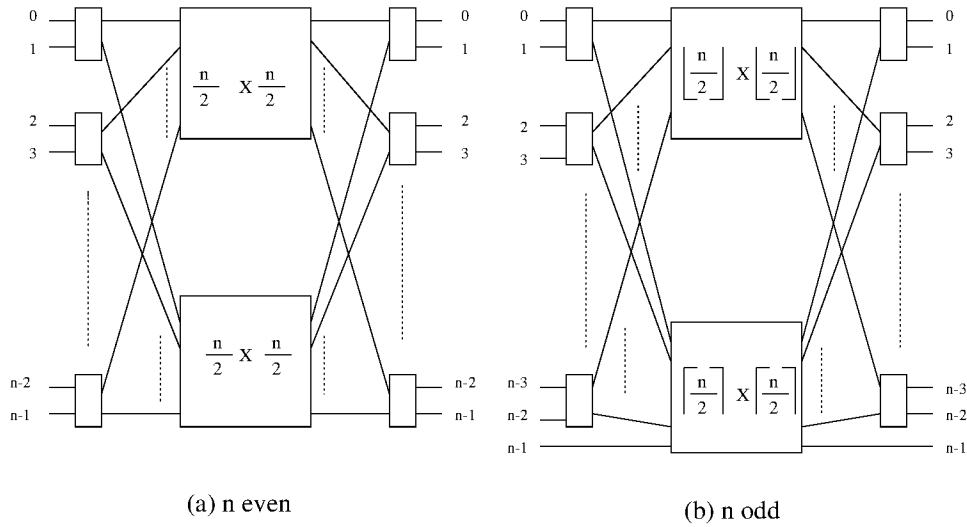
Figure 3: Constructions of AS-Benes networks

to the two AS-Benes networks. The last input and the last output are connected directly to the $\lceil \frac{n}{2} \rceil$ AS-Benes as shown in Figure 3(b). This process is illustrated in Figure 4 where an AS-Benes of size 5 is built from an AS-Benes of size 2 and an AS-Benes of size 3. To build an AS-Benes of size 6, two size 3 AS-Benes can be used, and in general, an AS-Benes of size n, for any n, may be constructed.

3   Routing Algorithm

A quick inspection of Figure 3 reveals that, except for paths involving the last input and/or the last output of odd size networks, a path between an input and an output may be established through either the upper AS-Benes sub-network (of size $\lfloor \frac{n}{2} \rfloor$) or the lower AS-Benes sub-network (of size $\lceil \frac{n}{2} \rceil$). Given that each switch at the first and last levels in an AS-Benes has precisely one connection to each of the upper and lower sub-networks, the realization of any given permutation, Π, in an AS-Benes should satisfy the property that paths sharing any switch at the first or last levels must go to different sub-networks. By enforcing this property, it can be shown that, given any one-to-one mapping, Π, of n inputs to n outputs, there is a set of edge-disjoint paths from the inputs of a size n AS-Benes to its outputs connecting input i to output $\Pi(i)$ for $0 \le i \le n-1$.

As an example, we illustrate the paths in Figure 5 for the mapping

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 7 & 4 & 8 & 6 & 2 & 1 & 0 & 3 & 5 \end{pmatrix}$$

in a 9×9 AS-Benes network. The bold paths represent the first loop which starts at input $n-1$ and terminates at output $n-1$. After this loop, there are only two pairs of input/output

3
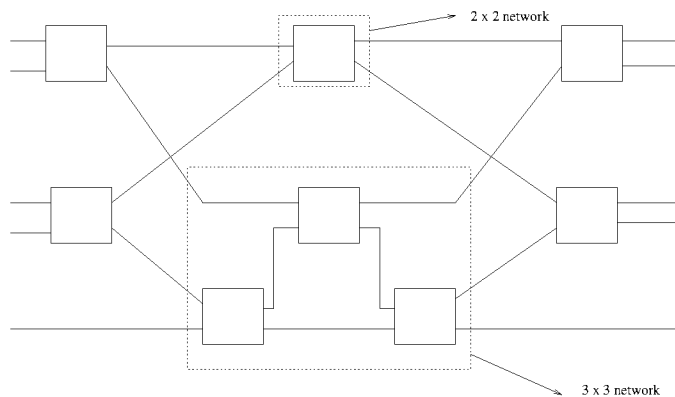
Figure 4: A 5 x 5 Benes network

left which form a second loop. In this way, all paths can be assigned to the upper or lower sub-networks without conflict.

4   Comparison with Benes Networks

It is important to be able to construct rearrangeable networks of arbitrary sizes with minimum cost. In this section, we compare the number of switches used for an AS-Benes of size n with a Benes of size $2^{\lceil \log n \rceil}$, which is the smallest Benes that can realize any $n \times n$ permutation. For instance, to realize any 5×5 permutations, an 8×8 Benes is needed, which requires twenty 2×2 switches. A 5×5 AS-Benes requires only eight 2×2 switches (see Figure 4). That is more than 100% saving. In general, if $S(k)$ is the number of switches used for a size k AS-Benes, then $S(1) = 0$, $S(2) = 1$ and

$$S(k) = 2\lfloor\frac{k}{2}\rfloor + S(\lceil\frac{k}{2}\rceil) + S(\lfloor\frac{k}{2}\rfloor).$$

It is easy to use induction to prove that the solution to the above equation satisfies $S(k) \leq \frac{k}{2}(2\,log k - 1)$. That is the number of switches in AS-Benes is of order $O(k\,log k)$. The recursive equation for $S(k)$ may be also used to compare the number of switches needed in an $n \times n$ AS-Benes, and an $2^{\lceil \log n \rceil} \times 2^{\lceil \log n \rceil}$ Benes. This comparison is shown in Figure 6 for n up to 32. Clearly, when n is a power of 2, AS-Benes is identical to a Benes. The curves follow similar trends for larger values of n.

Finally, we want to point out that different paths in an AS-Benes may pass through different number of switches. However, the maximum length of any path will never exceed the length of a path in a Benes of size $2^{\lceil \log n \rceil}$. In other word, the delay in an AS-Benes is at most equal to the delay in the corresponding Benes network.

4
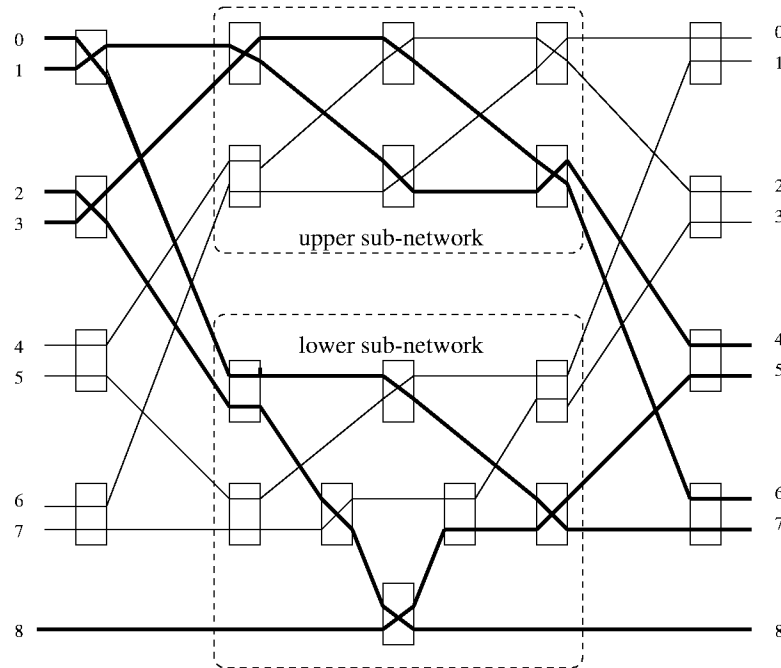
Figure 5: Two loops in the realization of a permutation in a 9 × 9 AS-Benes

5   Conclusions

We have shown that there is a simple and efficient way for building arbitrary size re-arrangeable networks of size n using $O(n\ logn)$ two by two switches, and for routing permutations in such networks.

References

[1] V. Benes. Permutation groups, complexes, and rearrangeable multistage connecting networks. *Bell System Technical Journal*, 43:1619–1640, 1964.

[2] C. Clos. A study of non-blocking switching networks. *Bell System Technical Journal*, 32:406–424, 1953.

[3] T.Y. Feng. A survey of interconnection networks. *IEEE Computer*, 14(12):12–27, 1981.

[4] T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*. Morgan Kaufmann Publishers, MIT, 1992.

[5] D. Opferman and N. Tsao-Wu. On a class of rearrangeable switching networks, part i: Control algorithm. *Bell System Technical Journal*, 50(5):1579–1600, 1971.
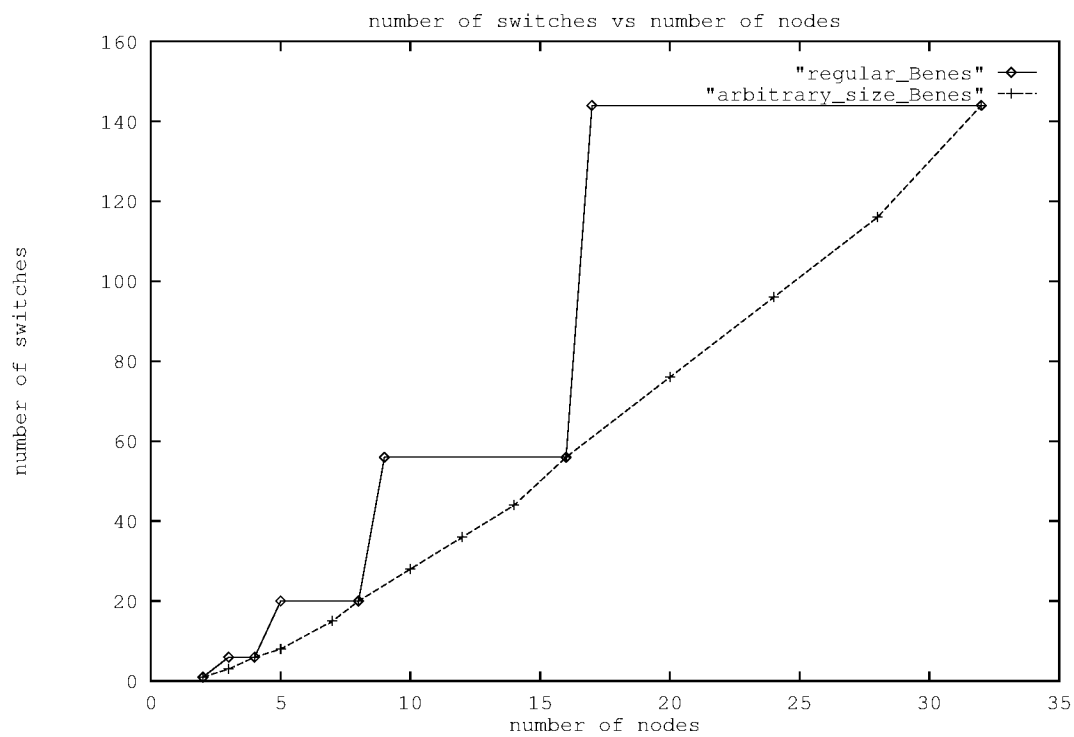
Figure 6: Comparisons between AS-Benes and Benes networks

A Study of Non-Blocking Switching Networks

By CHARLES CLOS

This paper describes a method of designing arrays of crosspoints for use in telephone switching systems in which it will always be possible to establish a connection from an idle inlet to an idle outlet regardless of the number of calls served by the system.

INTRODUCTION

The impact of recent discoveries and developments in the electronic art is being felt in the telephone switching field. This is evidenced by the fact that many laboratories here and abroad have research and development programs for arriving at economic electronic switching systems. In some of these systems, such as the ECASS System,* the role of the switching crossnet array becomes much more important than in present day commercial telephone systems. In that system the common control equipment is less expensive, whereas the crosspoints which assume some of the control functions are more expensive. The requirements for such a system are that the crosspoints be kept at a minimum and yet be able to permit the establishment of as many simultaneous connections through the system as possible. These are opposing requirements and an economical system must of necessity accept a compromise. In the search for this compromise, a convenient starting point is to study the design of crossnet arrays where it is always possible to establish a connection from an idle inlet to an idle outlet regardless of the amount of traffic on the system. Because a simple square array with N inputs, N outputs and N^2 crosspoints meets this requirement, it can be taken as an upper design limit. Hence, this paper considers non-blocking arrays where less than N^2 crosspoints are required. Specifically, this paper describes for an implicit set of conditions, crossnet arrays of three, five,

---

* Malthaner, W. A., and H. Earle Vaughan, An Experimental Electronically Controlled Switching System. Bell Sys. Tech. J., 31, pp. 443–468, May, 1952.

etc., switching stages where less than N^2 crosspoints are required. It then deals with conditions for obtaining a minimum number of crosspoints, cases where the N inputs and N outputs can not be uniformly assigned to the switches, switching arrays where the inputs do not equal the outputs, and arrays where some or all of the inputs are also outputs.

SQUARE ARRAY

A simple square array having N inputs and N outputs is shown in Fig. 1. The number of crosspoints equals N^2 and any combination of N or less simultaneous connections can exist without blocking between the inputs and the outputs. The number of switching stages, s, is equal to 1. The number of crosspoints, $C(s)$, is:

$$C(1) = N^2 \tag{1}$$
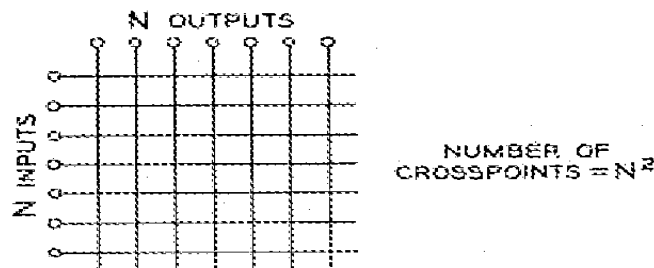
Fig. 1 — Square Array.

THREE-STAGE SWITCHING ARRAY

An array where less than N^2 crosspoints are required is shown in Fig. 2. This array has $N = 36$ inputs and $N = 36$ outputs. There are three switching stages, namely, an input stage (a), an intermediary stage (b), and an output stage (c). In stage (a) there are six 6 x 11 switches; in stage (b) there are eleven 6 x 6 switches; and in stage (c) there are six 6 x 11 switches. In total, there are 1188 crosspoints which are less than the 1296 crosspoints required by equation (1).

Of interest are the derivations of the various quantities and sizes of switches. In stage (a) the number, n, of inputs per switch was assumed to be equal to $N^{1/2}$, thus giving six switches and six inputs per switch. In a similar manner stage (c) was assigned six switches and six outputs per switch. The number of switches required in stage (b) must be sufficient to avoid blocking under the worst set of conditions. The worst case occurs when between a given switch in stage (a) and a given switch in stage (c): (1) five links from the switch in stage (a) to five correspond-

408   THE BELL SYSTEM TECHNICAL JOURNAL, MARCH 1953

ing switches in stage (b) are busy; (2) five links from the switch in stage (c) are busy to five additional switches in stage (b); and (3) a connection is desired between the given switches. Thus eleven switches are required in stage (b). The remaining requirements, namely, eleven verticals per switch in stages (a) and (c) and six by six switches in stage (b) are then easily derived.

The number of crosspoints required for three stages, where $n = N^{1/3}$, is summarized by the following formula:

$$C(3) = (2N^{1/2} - 1)(3N) \tag{2}$$

$$= 6N^{3/2} - 3N \tag{2a}$$

In Table I it may be noted that the number of crosspoints is less than N^2 for all cases of $N \geqq 36$.

PRINCIPLE INVOLVED

The principle involved for determining the number of switches required in the intermediary stage is illustrated in Fig. 3. The figure is for a specific case from which one can generalize for n inputs on a given input switch and m outputs on a given output switch. In the figure it is desired to establish a connection from input B to output H. A sufficient number of intermediary switches are required to permit the $(n - 1)$ inputs other than B on the particular input switch and the $(m - 1)$ outputs other than H on the particular output switch to have connections to separate intermediary switches plus one more switch for the desired connection between B and H. Thus $n + m - 1$ intermediary switches are required.

TABLE I — CROSSPOINTS FOR SEVERAL VALUES OF N

N	Square Array N^2	Three-Stage Array $6N^{3/2} - 3N$
4	16	36
9	81	135
16	256	336
25	625	675
36	1,296	1,188
49	2,401	1,911
64	4,096	2,880
81	6,561	4,131
100	10,000	5,700
1,000	1,000,000	186,737
10,000	100,000,000	5,970,000

NON-BLOCKING SWITCHING SYSTEMS   409
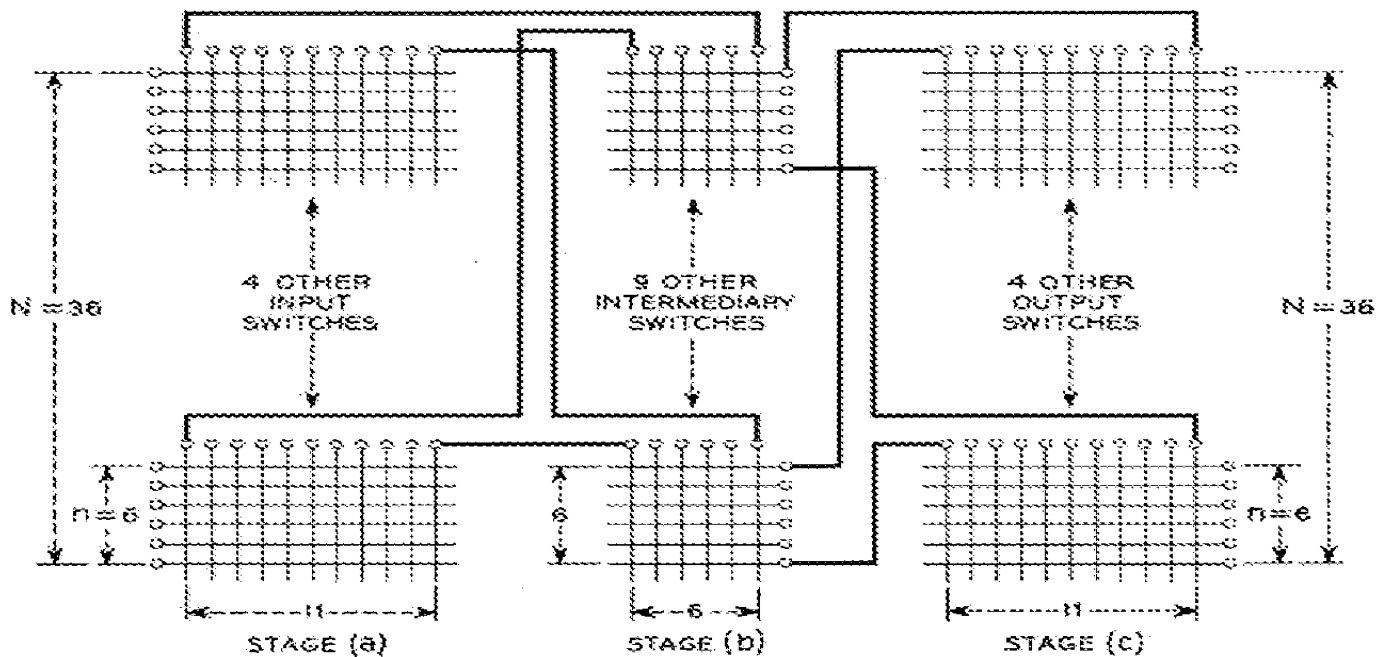
NUMBER OF CROSSPOINTS = $6N^{3/2} - 3N$ (1188 CROSSPOINTS WHEN N = 36)

Fig. 2 — Three-stage switching array.

FIVE-STAGE SWITCHING ARRAY

A five-stage switching array is illustrated in Fig. 4. The analysis of this array can be made in the following manner. Each input and output switch is assumed to have $n = N^{1/3}$ inputs or outputs, respectively. Connection between a given input switch and a given output switch
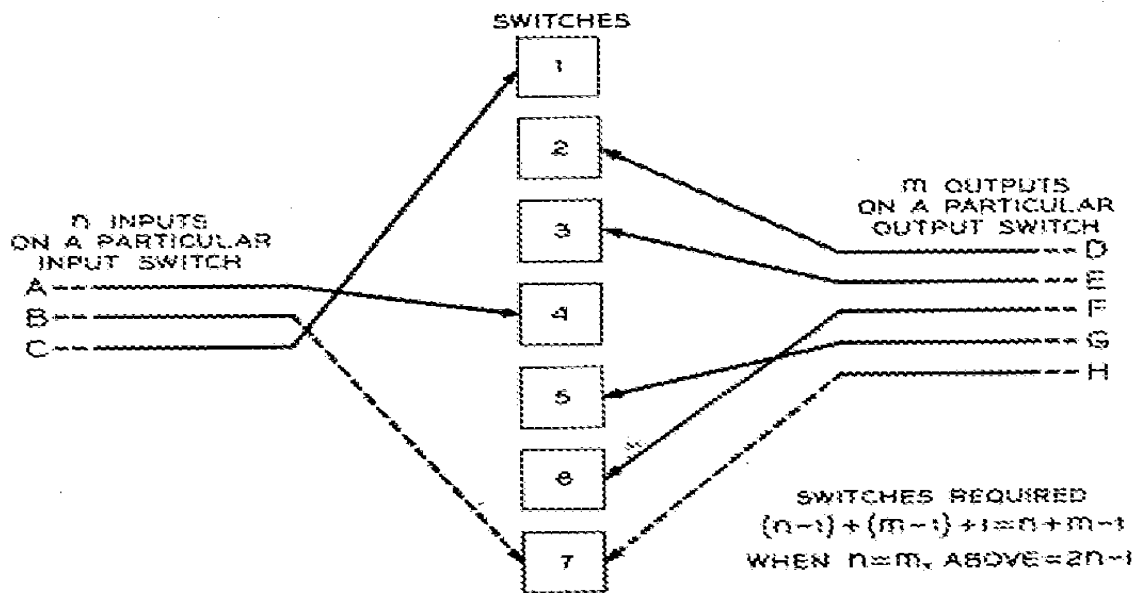
Fig. 3 — Principle involved.

is made via levels, a level consisting of three intermediary switching stages. The number of levels required is $(2N^{1/3} - 1)$. Each level has $N^{2/3}$ inputs and the same number of outputs. The number of crosspoints for a three-stage non-blocking array for $N^{2/3}$ inputs and $N^{3/3}$ outputs can be obtained from equation (2) by substituting $N^{2/3}$ for N in that equation. The total number of crosspoints required for the five-stage array is:

$$C(5) = (2N^{1/3} - 1)^2 \, 3N^{2/3} + (2N^{1/3} - 1) \, 2N \qquad (3)$$

$$= 16N^{4/3} - 14N + 3N^{2/3} \qquad (3a)$$

The number of crosspoints required for several sizes of the five-stage array is given in Table II. The results are compared to the square and three-stage arrays.

SEVEN-STAGE SWITCHING ARRAY

A seven-stage switching array can be analyzed by considering paths requiring five intermediary switching stages as paths via switching aggregates. The number of such aggregates is $(2N^{1/4} - 1)$. Each aggregate has $N^{3/4}$ inputs and a like number of outputs. From equation (3) the crosspoints for each aggregate can be obtained by substituting
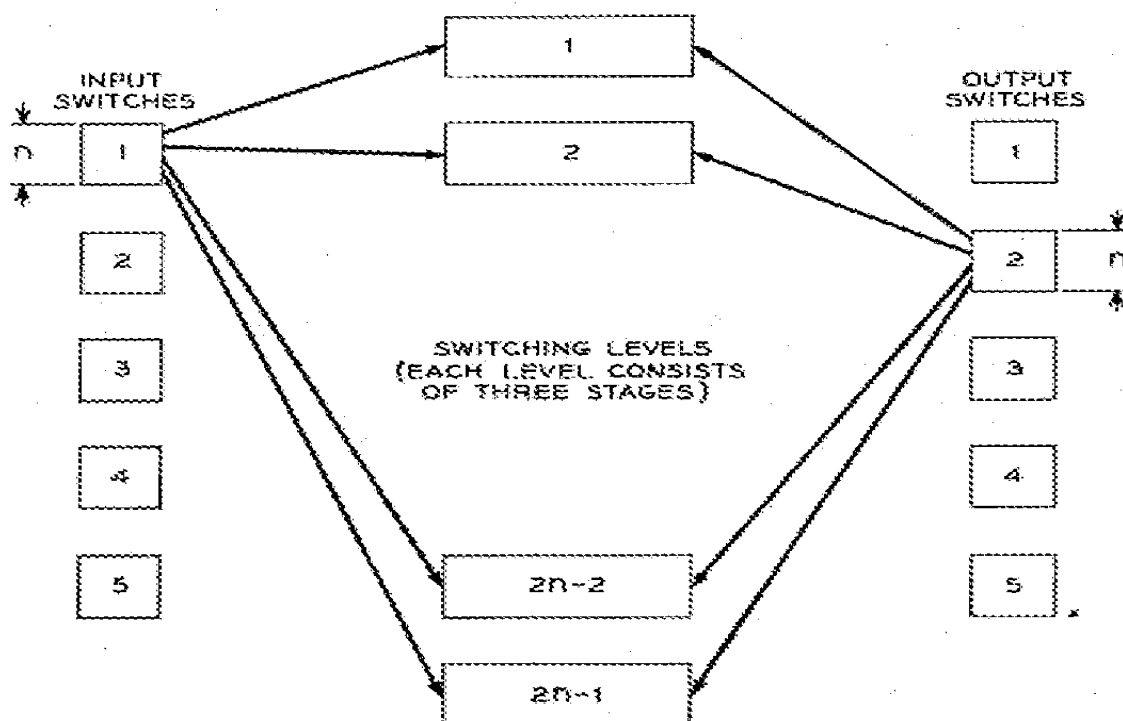
Fig. 4 — Five-stage switching array.

NON-BLOCKING SWITCHING SYSTEMS   411

TABLE II — CROSSPOINTS FOR SEVERAL VALUES OF N

N	Square Array	Three-Stage Array	Five-Stage Array
64	4,096	2,880	3,248
729	531,441	115,911	95,013
1,000	1,000,000	186,737	146,300
10,000	100,000,000	5,970,000	3,434,488

$N^{3/4}$ for N in that equation. The total number of crosspoints required for the seven-stage array is:

$$C(7) = (2N^{1/4} - 1)^3 \, 3N^{1/2} + (2N^{1/4} - 1)^2 \, 2N^{3/4} + (2N^{1/3} - 1)2N \quad (4)$$

$$= 36N^{5/4} - 46N + 20N^{3/4} - 3N^{1/2} \quad (4a)$$

GENERAL MULTI-STAGE SWITCHING ARRAY

Equations (1), (2a), (3a) and (4a) are herewith tabulated as a series of polynomials together with the next polynomial:

$$C(1) = N^2 \quad (1)$$

$$C(3) = 6N^{3/2} - 3N \quad (2a)$$

$$C(5) = 16N^{4/3} - 14N + 3N^{2/3} \quad (3a)$$

$$C(7) = 36N^{5/4} - 46N + 20N^{3/4} - 3N^{1/2} \quad (4a)$$

$$C(9) = 76N^{6/5} - 130N + 86N^{4/5} - 26N^{3/5} + 3N^{2/5} \quad (5)$$

These polynomials can be determined for any number of switching stages from the following formula where s is an odd integer:

$$C(s) = 2\sum_{k=2}^{\frac{s+1}{2}} N^{\frac{2k}{s+1}} \left(2N^{\frac{2}{s+1}} - 1 \right)^{\frac{s+3}{2}-k} + N^{\frac{4}{s+1}} \left(2N^{\frac{2}{s+1}} - 1 \right)^{\frac{s-1}{2}} \quad (6)$$

An alternative expression equivalent to equation (6) has been suggested by S. O. Rice and J. Riordan. The recurrence relation used in individually deriving the foregoing polynomials can be used to directly derive the following formula:

$$C(2t + 1) = \frac{n^2(2n - 1)}{n - 1} [(5n - 3)(2n - 1)^{t-1} - 2n^t] \quad (6a)$$

where $s = 2t + 1$

$$N = n^{t+1}$$

Table III gives comparative numbers of crosspoints for various num-

412   THE BELL SYSTEM TECHNICAL JOURNAL, MARCH 1953

TABLE III — CROSSPOINTS FOR VARIOUS NUMBERS OF SWITCHING
STAGES, s, AND VALUES OF N

N	$s = 1$	$s = 3$	$s = 5$	$s = 7$	$s = 9$
100	10,000	5,700	6,092	7,386	9,121
200	40,000	16,370	16,017	18,898	23,210
500	250,000	65,582	56,685	64,165	78,058
1,000	1,000,000	186,737	146,300	159,904	192,571
2,000	4,000,000	530,656	375,651	395,340	470,292
5,000	25,000,000	2,106,320	1,298,858	1,295,294	1,511,331
10,000	100,000,000	5,970,000	3,308,487	3,159,700	3,625,165
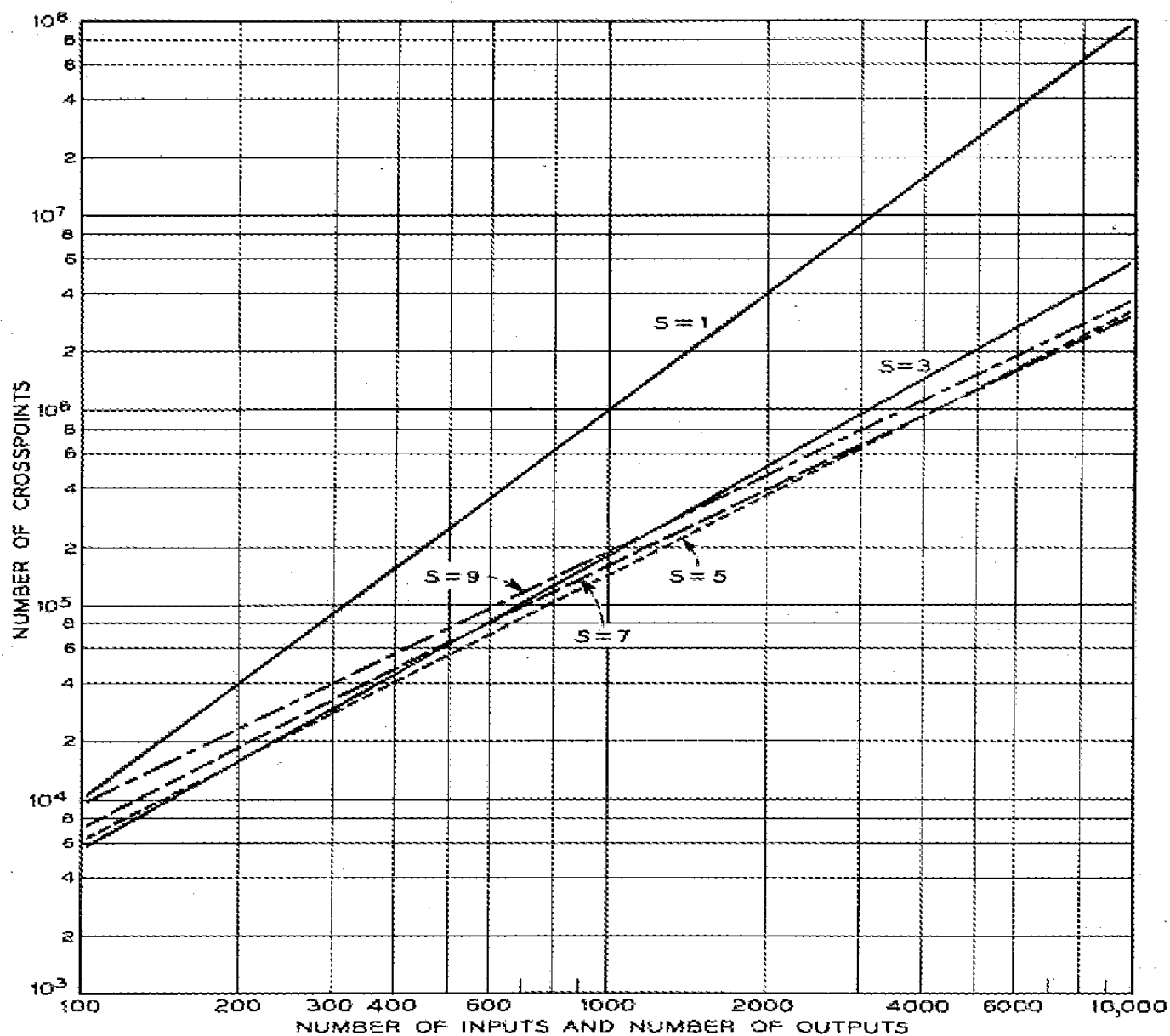
Fig. 5 — Crosspoints versus switching stages.

bers of switching stages and sizes of N. The data of Table III are plotted on Figure 5. The series of curves appear to be bounded by an envelope, representing a minimum of crosspoints. The next section dealing with minima indicates that points exist below this envelope.

MOST FAVORABLE SIZE OF INPUT AND OUTPUT SWITCHES IN THE THREE-STAGE ARRAY

The foregoing derivations were for implicit relationships between n and N, namely, n being the $\left(\dfrac{s+1}{2}\right)th$ root of N. To obtain minimum number of crosspoints a more general relationship is required. For the three stage switching array this is:

$$C(3) = (2n - 1)\left(2N + \frac{N^2}{n^2}\right) \tag{7}$$

When $n = N^{1/2}$ equation (7) reduces to equation (2).

For a given value of N, the minimum number of crosspoints occurs when $dC/dn = 0$ which gives:

$$2n^3 - nN + N = 0 \tag{8}$$

This equation has the following two pairs of integral values:

$$n = 2, \qquad N = 16 \qquad \text{and} \qquad n = 3, \qquad N = 27$$

As N approaches large values equation (8) can be approximated by:

$$N \doteq 2n^2 \tag{9}$$

Graphs of equations (8) and (9) are shown in Fig. 6. In Table IV the numbers of crosspoints are based on the nearest integral values of n for given values of N.

Where comparisons can be made, Table IV indicates fewer crosspoints than does Table I. This fact can be realized in another manner. By eliminating n in equations (7) and (9), the result for large values of N is:

$$C(3) \doteq 4 \, (2)^{1/2} N^{3/2} - 4N \tag{10}$$

Equation (10) indicates fewer crosspoints than does equation (2).

MOST FAVORABLE SWITCH SIZES IN THE FIVE-STAGE ARRAY

If n be the number of inputs per input switch and outputs per output switch, and m be the number of inputs per switch in the second stage

414   THE BELL SYSTEM TECHNICAL JOURNAL, MARCH 1953
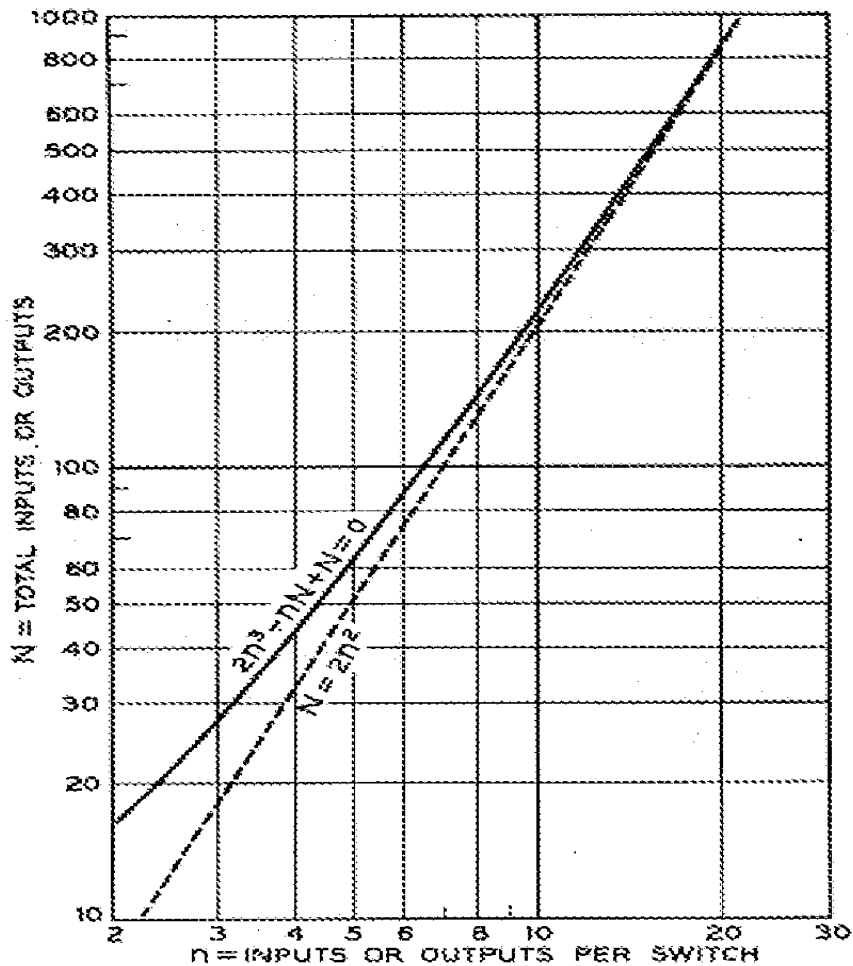
Fig. 6 — Relationship between N and n for minima in crosspoints. Three-stage array.

TABLE IV — CROSSPOINTS FOR SEVERAL VALUES OF N

N	Nearest Integral Value of n	Number of Crosspoints	
		N^2	Equation (7)
16	2	256	288
27	3	729	675
40	4	1,600	1,280
44	4	1,936	1,463
55	5	3,025	2,079
60	5	3,600	2,376
65	5	4,225	2,691
78	6	6,084	3,575
84	6	7,056	4,004
98	7	9,604	5,096
105	7	11,025	5,655

and outputs per switch in the fourth stage, then the following equation gives the total number of crosspoints:

$$C(5) = (2n - 1)\left[2N + (2m - 1)\left(\frac{2N}{n} + \frac{N^2}{n^3 m^2}\right)\right] \qquad (11)$$

The partial derivative of this equation with respect to m when set equal to zero yields:

$$n = \frac{N(m - 1)}{2m^3} \qquad (12)$$

The partial derivative of this equation with respect to n when set equal to zero yields the following equation:

$$N = \frac{nm^2(2n^2 + 2m - 1)}{(2m - 1)(n - 1)} \qquad (13)$$

Equations (12) and (13) can be solved for n and m in terms of given values of N. For example for $N = 240$, we obtain $n = 6.81$ and $m = 3.56$.

SEARCH FOR THE SMALLEST N FOR A GIVEN n FOR THE THREE-
STAGE ARRAY

For a given value of n, equation (7) furnishes a means for locating that size of three-stage switching array which has N^2 or fewer crosspoints. This can be done by setting equation (7) equal to N^2:

$$N^2 = (2n - 1)\left(2N + \frac{N^2}{n^2}\right) \qquad (14)$$

and solving for N in terms of n. The solution is:

$$N \geqq \frac{2n^2(2n - 1)}{(n - 1)^2} \qquad (15)$$

Minimum values of N for given values of n are listed in Table V. This table also lists the next highest N exactly divisible by n. From this table it appears that when $N = 24$, we have the smallest switching array for which it may be possible to have less than N^2 crosspoints. However for $N = 25$, as shown in Table I, equation (2) gives more than N^2 crosspoints. The problem is one of finding an array for $N = 25$ with fewer than N^2 crosspoints. For this and all cases beyond, the next section indicates that it is profitable to consider situations where N is not exactly divisible by n.

CASES IN THE THREE-STAGE SWITCHING ARRAY WHERE $N \equiv r(\text{MOD } n)$

Table I indicated that for $N = 25$ and $n = 5$ a total of 675 crosspoints were required. A square array requires only 625. Fig. 7 shows a layout of switches where $N = 25$ and $n = 3$. In this case one input is left over when 25 inputs are divided into threes. The lone input requires three paths to the intermediary switches. This is in accordance with Fig. 3. The lone output also requires three paths to the intermediary switches. Also from Fig. 3, the lone input to the lone output requires only one path. Hence there must be one switch capable of connecting the lone input to the lone output. The number of crosspoints required is 615 which is less than the 625 required by the square array. This scheme can be extended to any case where $N = kn + r$, where the remainder, r, is an integer greater than zero but less than n. The formula for the number of crosspoints where k input and k output switches of size n and one input and output switch of size r are used is:

$$C = 2(2n - 1)(N - r) + 2(n + r - 1)r + (n - r)\left(\frac{N - r}{n}\right)^2$$
$$+ (n + r - 1)\left(\frac{N - r}{n} + 1\right)^2 - n + r \tag{16}$$

I. G. Wilson has pointed out that for a lone input the crosspoints in the intermediary switches can be used to isolate its possible connections hence no crosspoints are required in the input stage. This likewise applies for a lone output. With this modification the array in Fig. 7 requires six fewer crosspoints. For this case, when $r = 1$, the number of crosspoints is:

$$C = 2(2n - 1)(N - 1) + (n - 1)\left(\frac{N - 1}{n}\right)^2$$
$$+ n\left(\frac{N - 1}{n} + 1\right)^2 - n + 1 \tag{16a}$$

TABLE V — MINIMUM VALUES OF N FOR GIVEN VALUES OF n

n	N per Equation 15	$N = 0 \ (\text{mod } n)$
2	24	24
3	22.5	24
4	24.9	28
5	28.1	30
6	31.7	36

NON-BLOCKING SWITCHING SYSTEMS   417

J. Riordan has found a more efficient arrangement for cases where $N = kn + r$. In place of using k switches of size n and one switch of size r, he proposes that $(k + 1 - n + r)$ switches of size n and $(n - r)$ switches of size $n - 1$ be used. For this case the number of crosspoints is:

$$C = 2(2n - 1)(k + 1 - n + r)n + 2(2n - 2)(n - r)(n - 1)$$
$$+ (2n - 3)(k + 1)^2 + 2(k + 1)(k + 1 - n + r) \qquad (17)$$
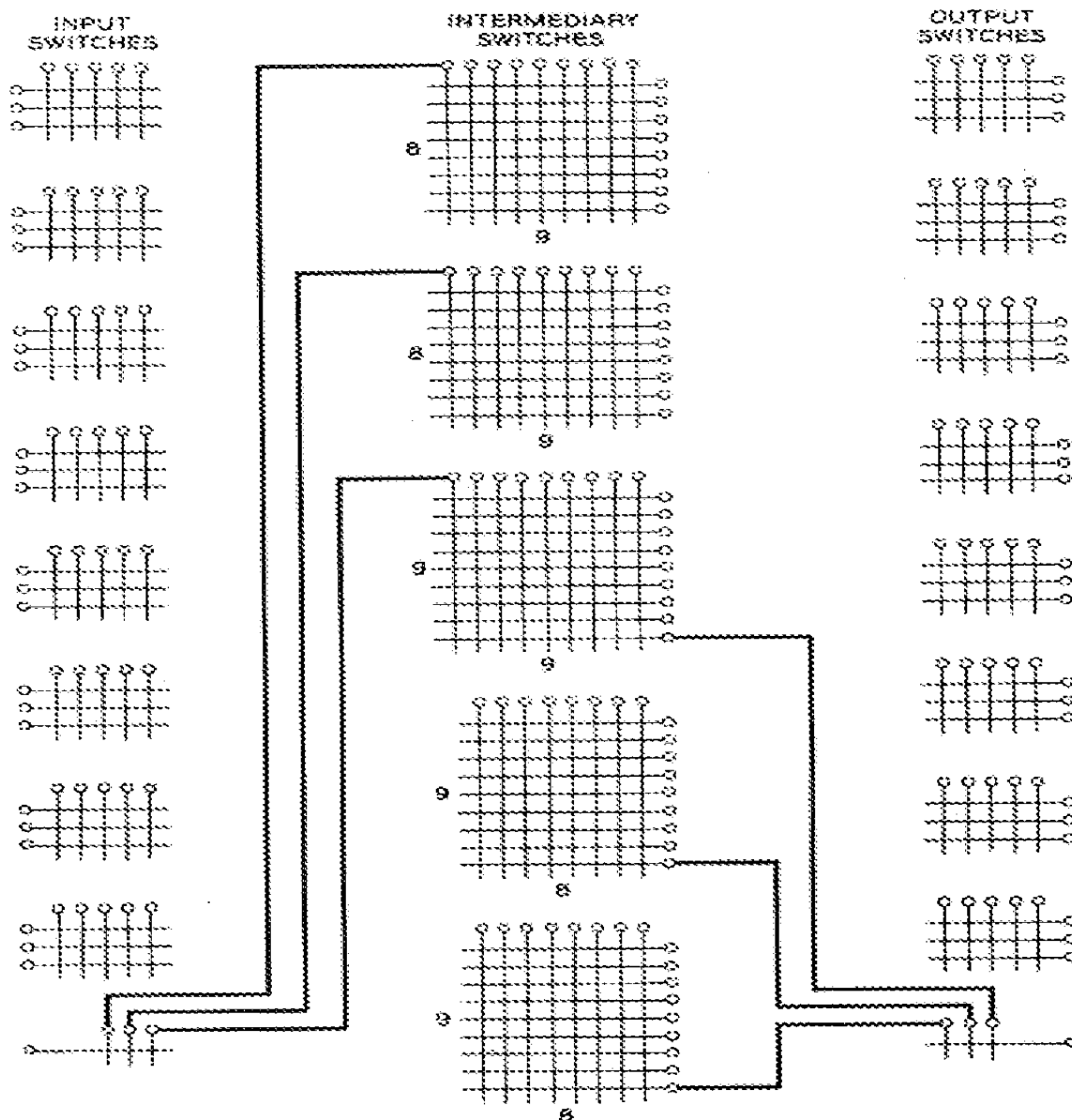
Fig. 7 — Three-stage array. $25 \equiv 1 \pmod 3$. An equivalent arrangement is to provide two 8 x 8 and three 9 x 9 intermediary switches. Two of the 9 x 9 switches need only 80 crosspoints.

418   THE BELL SYSTEM TECHNICAL JOURNAL, MARCH 1953

TABLE VI — CROSSPOINTS FOR VARIOUS VALUES OF N AND n

N	Square Array	Three-Stage Array			
		$n = 2$	$n = 3$	$n = 4$	$n = 5$
23	529*	540	530	556	----
24	578	578	580*	588	----
25	625	625	609*	633	----
26	----	663	643*	687	----
27	----	716	675*	701	----
28	----	756	730*	735	----
29	----	813	766*	788	----
30	----	855	800*	824	864
31	----	----	861	860*	911
32	----	----	899	896*	951
33	----	----	925*	957	991
34	----	----	1002	995*	1031
35	----	----	1042	1033*	1071
36	----	----	1080	1071*	1128
37	----	----	1153	1140*	1170
38	----	----	1195	1180*	1212
39	----	----	1235	1220*	1254
40	----	----	1314	1260*	1296
		$n = 4$	$n = 5$	$n = 6$	$n = 7$	$n = 8$
50	1819	1800*	1879	----	----
60	2415	2376*	2430	----	----
70	3164	3024*	3056	----	----
80	3920	3744*	3764	----	----
90	----	4536	4455*	4499	----
100	----	5400	5291*	5315	----
110	----	----	6199	6100	6156
120	----	----	7040	7044	6975*
130	----	----	8076	7923*	7947
140	----	----	----	8840*	8860
150	----	----	----	9968	9811*
160	----	----	----	10979	10800*

* Minimum values.

Equation (17) is identical to equation (16) when $r = n - 1$. There are two cases, namely, when $n = 2$ and $n = 3$ where equation (16a) gives fewer crosspoints than does equation (17).

SEARCH FOR THE MINIMUM NUMBER OF CROSSPOINTS BETWEEN $N = 23$ AND $N = 160$

The equations of the preceding sections furnish a means for searching for minimum crossnet arrays. Table VI shows the results of such a search up to $N = 160$. Results are indicated in unit steps from $N = 23$ to $N = 40$ and for every tenth interval thereafter. At $N = 161$, a five-stage array requires the fewest crosspoints.

Table VI was computed by the use of finite differences. The equations

NON BLOCKING SWITCHING SYSTEMS   419

were:

$$C[(k+1)n] \quad - C(kn) \quad = (2n-1)(2n+2k+1) \qquad (18)$$

$$C(kn+r+1) - C(kn+r) = 2(k+3n-1) \qquad (19)$$

$$C(kn+1) \quad - C(kn) \quad = 2kn+1 \qquad (19a)$$

Equation (18) was derived from equation (7) with N being replaced by $(k+1)n$ and by kn as required. Equation (19) was derived from equation (17) with r being replaced by $r+1$ as required. This equation applies for all values of n greater than 3 and for the particular case of $n = 3$ and $r = 2$. Equation (19a) was derived from Equations (16a) and (7) and is for the particular case of $r = 1$, when $n = 2$ and $n = 3$.

SEARCH FOR THE MINIMUM NUMBER OF CROSSPOINTS FOR $N = 240$

For a case where N is large enough to require five-switching stages, the search for the minimum number of crosspoints should be based on equations (12) and (13) and on the use of Table VI. The method is suggested by means of Table VII. The data in a previous section indicate

TABLE VII — CROSSPOINTS FOR $N = 240$ AND VARIOUS VALUES OF n

n	Input and Output Stages			Intermediary Stages				Total Crosspoints
	No. of Switches	Size of Switches	Crosspoints	No. of Levels	Inputs and Outputs	m	Crosspoints	
2	120	2 x 3	1,440	3	120 x 120*	8	20,925	22,365
3	80	3 x 5	2,400	5	80 x 80*	5	18,720	21,120
4	60	4 x 7	3,360	7	60 x 60*	5	16,632	19,992
5	48	5 x 9	4,320	9	48 x 48	4	15,120	19,440
6	40	6 x 11	5,280	11	40 x 40*	4	13,860	19,140
7	{30	7 x 13	5,460	3	30 x 35	3	1,826}	19,369
	5	6 x 12	720	11	35 x 35*	4	11,363}	
8	20	8 x 15	7,200	15	30 x 30*	3	12,000	19,200
9	{24	9 x 17	7,344	2	24 x 27	3	1,230}	19,467
	3	8 x 16	768	15	27 x 27*	3	10,125}	
10	24	10 x 19	9,120	19	24 x 24*	3	10,640	19,760
11	{20	11 x 21	9,240	2	20 x 22	—	880}	20,116
	2	10 x 20	800	19	22 x 22	—	9,196}	
12	20	12 x 23	11,040	23	20 x 20	—	9,200	20,240

Crosspoints per equation (3) five-stage array.   20,596
Crosspoints per equation (2) three-stage array.   21,824
Crosspoints per equation (1) square array.   57,800

* See Table VI for minimum number of crosspoints.

420   THE BELL SYSTEM TECHNICAL JOURNAL, MARCH 1953

that a minimum should occur for $N = 240$, when $n = 6.81$ and $m = 3.56$. In Table VII the minimum occurs when $n = 6$ and $m = 4$. It fails to occur at $n = 7$ because 240 is not exactly divisible by 7. Except for this situation, the minimum would have occurred as predicted.

RECTANGULAR ARRAY

Referring to Fig. 1, if there were N inputs and M outputs, a simple rectangular array would result which would be capable of sustaining up to N or M, whichever is the lesser, simultaneous connections without blocking. The number of crosspoints is:

$$C(1) = NM \qquad (20)$$

N INPUTS AND M OUTPUTS IN A THREE-STAGE ARRAY

For the case of a three-stage switching array with N inputs and M outputs, let there be n inputs per input switch and m outputs per output switch. A particular input to be able to connect without blocking under the worst set of conditions to a particular output will require $(n - 1) + (m - 1) + 1$ available paths. Thus by providing for that many intermediary switches, a non-blocking switching array is obtained. The number of crosspoints is:

$$C(3) = (n + m - 1)\left[N + M + \frac{NM}{nm}\right] \qquad (21)$$

Differentiating this equation first with respect to n and then to m yields two partial differential equations whose solution indicates that a minimum is reached when $n = m$. Replacing m by n in equation (21), the equation for the number of crosspoints becomes:

$$C(3) = (2n - 1)\left[N + M + \frac{NM}{n^2}\right] \qquad (22)$$

Solving for the minimum number of crosspoints gives the following expression:

$$n^3 - \frac{NM}{N + M} n + \frac{NM}{N + M} = 0 \qquad (23)$$

When $N = M$ this equation reduces to equation (8).
The three-way relationships of n, N and M are shown in Fig. 8.

NON-BLOCKING SWITCHING SYSTEMS   421
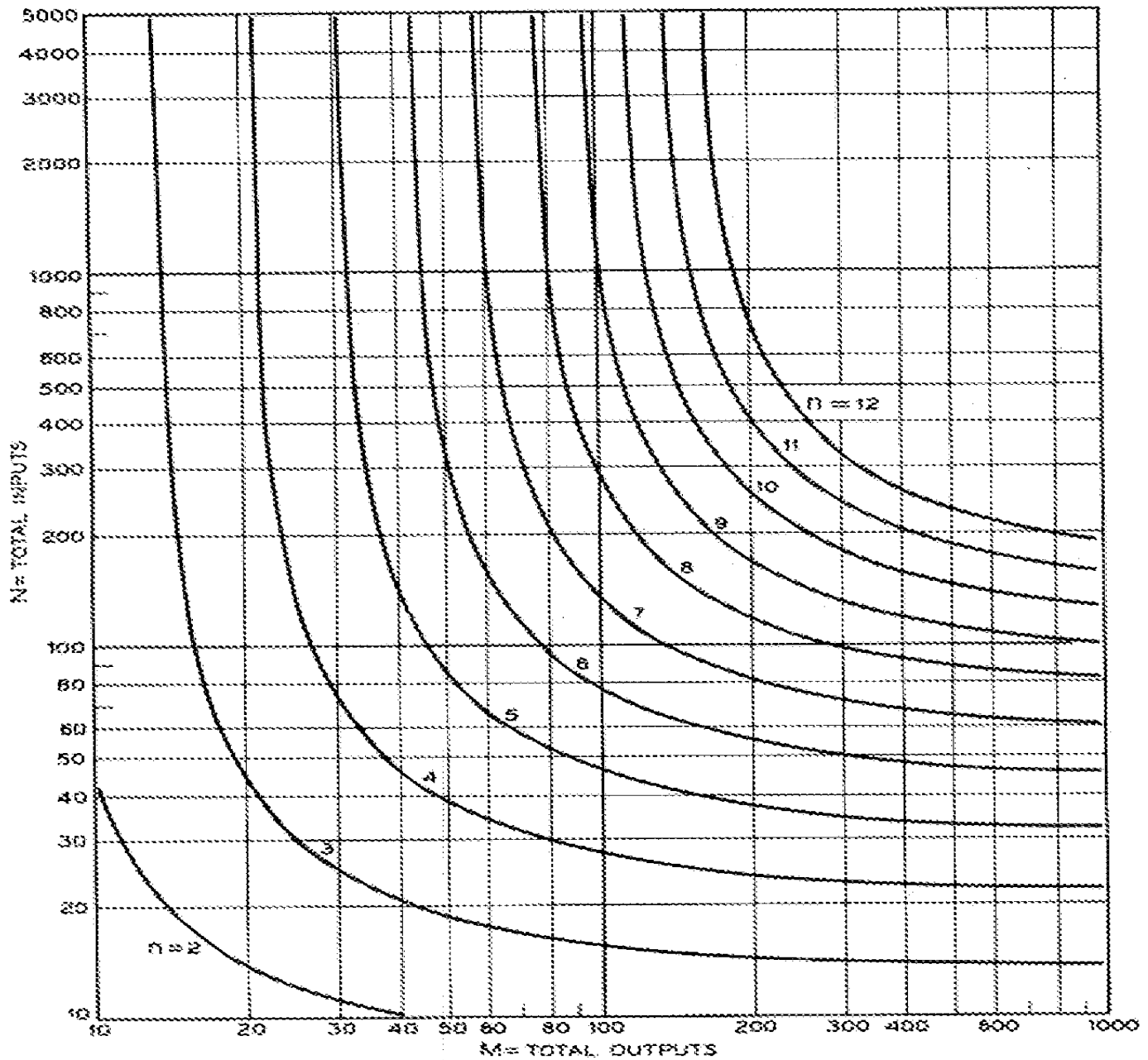
Fig. 8 — Relationship of n to N inputs and M outputs for a minimum in crosspoints in a three stage array.

TRIANGULAR ARRAY

If a case exists where all inputs are also the outputs, then an arrangement such as is shown in Fig. 9 can be used. The crosspoints in the intermediary switches permit connections between all switches on the left hand side. For connections between two trunks on the same switch it is assumed that one of the links to an intermediary switch can

be used to establish the connection but without affecting any of the crosspoints on the intermediary switch. The number of crosspoints for this case is:

$$C = (2n - 1)\left(T + \frac{T^3}{2n^2} - \frac{T}{2n}\right) \tag{24}$$

where T = number of two-way trunks.

By differentiation, conditions for obtaining minimum numbers of crosspoints can be determined. The arrangement can also be extended to cases where extra switching stages are required.

ONE-WAY INCOMING, ONE-WAY OUTGOING AND TWO-WAY TRUNKS

A combination of the triangular array of Fig. 9 and of unequal inputs and outputs is shown in Fig. 10. In this figure, one-way incoming,
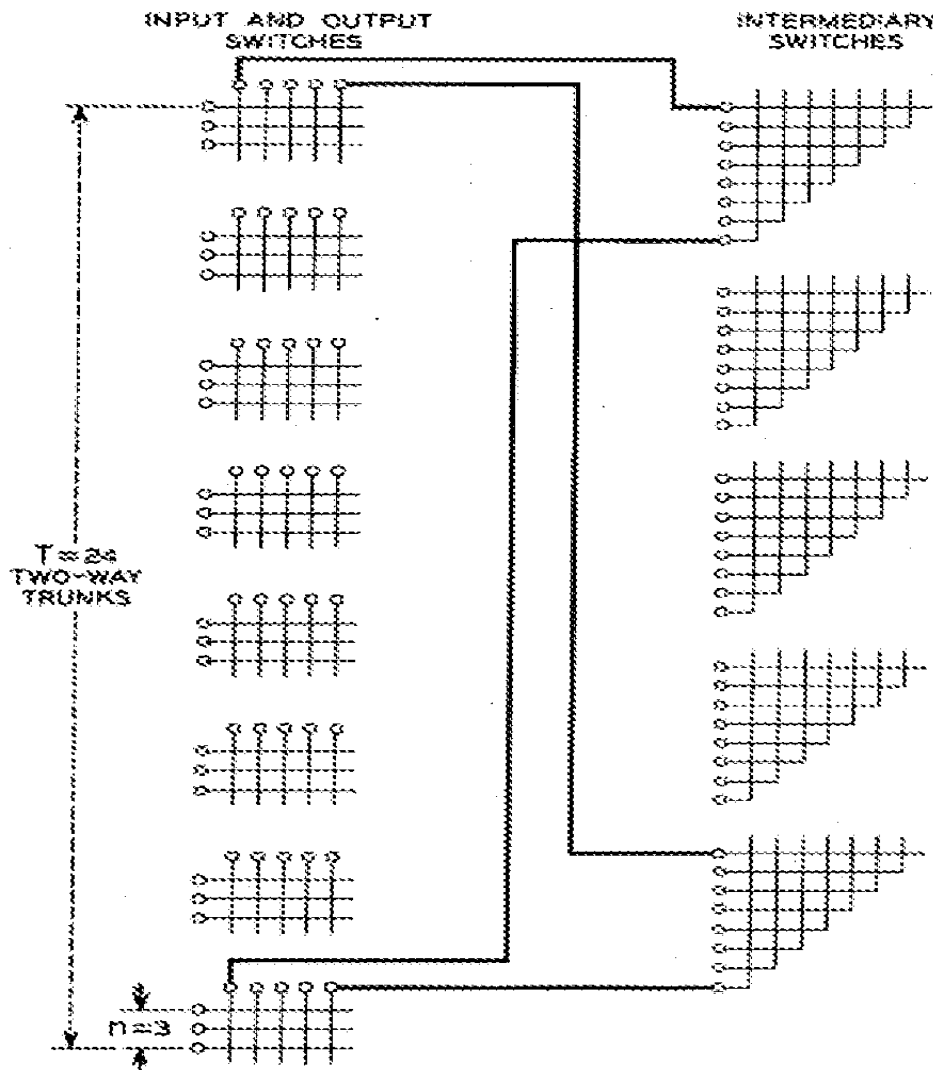
Fig. 9 — Triangular array.

one-way outgoing and two-way trunks can be freely interconnected without blocking. The number of crosspoints for this case is:

$$C = (2n - 1)\left[N + T + M + \frac{NT}{n^2} + \frac{MT}{n^2} + \frac{T^2}{2n^2} - \frac{T}{2n}\right] \quad (25)$$

The comments concerning the triangular array also apply for this case.

COMPARISON WITH EXISTING NETWORKS

Few existing crossnet arrays are non-blocking. An example is the four-wire intertoll trunk concentrating system. In one of its standard sizes 4,000 crosspoints are required for 100 incoming trunks and 40 outgoing intertoll trunks. From Fig. 8, for $N = 100$ and $M = 40$ it may be noted that the nearest integral value for n is 5. By substituting this value in equation (22), a non-blocking three-stage switching array of 2,700 cross-
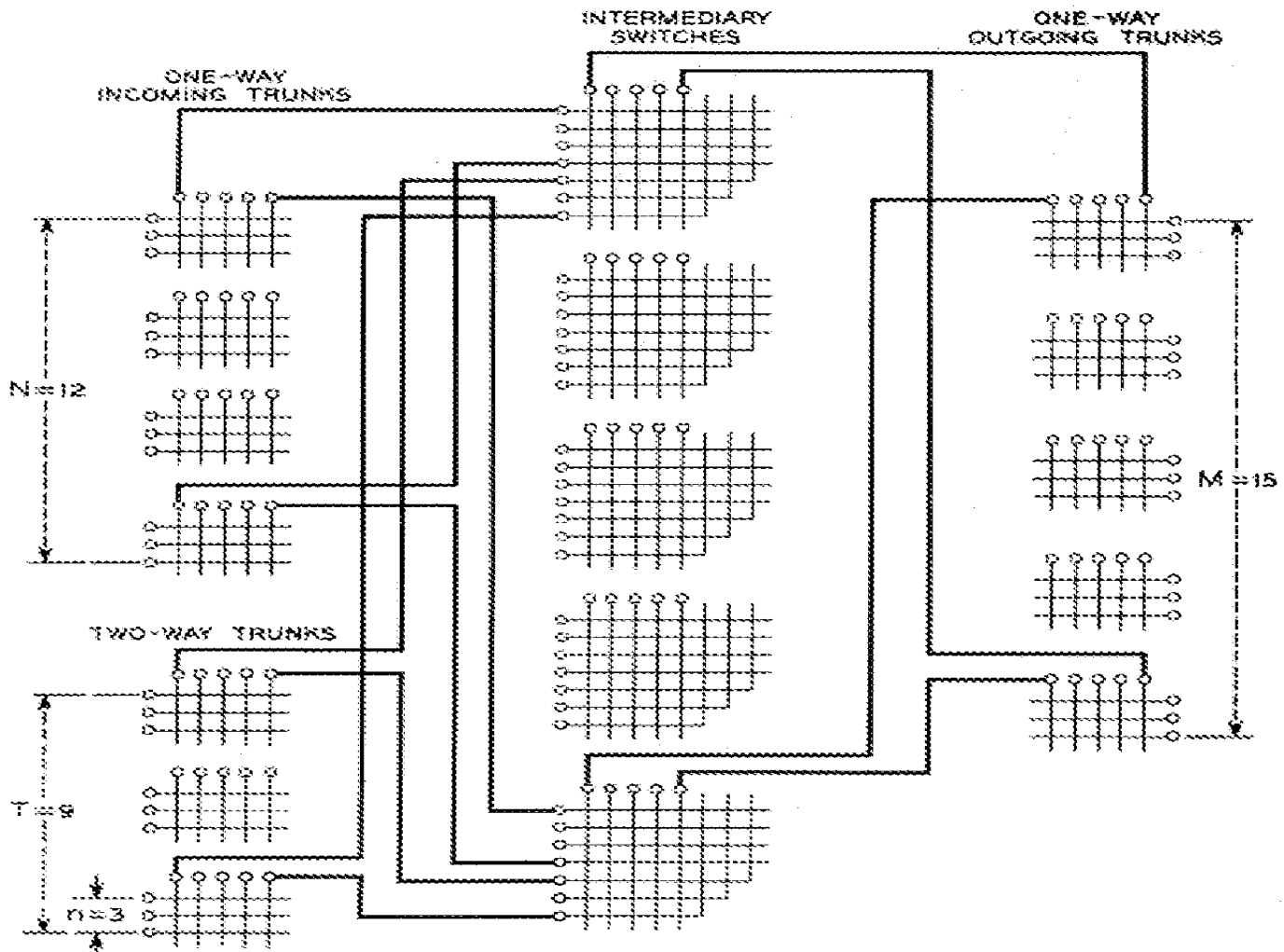
Fig. 10 — One-way incoming, one-way outgoing and two-way trunks.

points is found which could be used for the concentrating switch. In this case the new approach to the switching network problem may prove to be of value.

Comparisons with existing arrays having blocking are likely to be unfavorable because the grades of service are not the same. For instance, a No. 1 crossbar district-to-office layout of 1,000 district junctors and 1,000 trunks requires 80,000 crosspoints. This layout can handle 708 erlangs with a blocking loss of 0.0030. The minimum number of crosspoints with a non-blocking array is slightly less than 138,000. This, however, can handle 1,000 erlangs without blocking. By introducing blocking into the design methods described in this paper, a more favorable comparison with existing arrays having blocking can be made. This can be done by omitting certain of the paths. If done to an array requiring 1,000 inputs and 1,000 outputs a layout can be obtained requiring 79,900 crosspoints with a blocking loss of 0.0022 for a load of 708 erlangs. For this example, at least, it appears that the new design methods may prove to be valuable especially for use in the development of electronic switching systems where the control mechanism may not be dependent upon the particular switching array used.

CONCLUSION

In present day commercial telephone systems the use of non-blocking switching networks is rare. This may be due to the large number of crosspoints required. With the design methods described herein, a wider use of non-blocking networks may occur in future developments. For the usual case of networks with blocking, new systems have generally been designed by an indirect process. Several types and sizes of switching arrays are studied until the most economical one for a given level of blocking is found. With the new design methods, a straightforward approach is possible. Fig. 5 indicates that a region of minimum values exists. By first designing a non-blocking system with a reasonable number of switching stages and then omitting certain of the paths, the designer can arrive at a network with a given level of blocking and be very close to a minimum in crosspoints. The possibility of the adoption of this direct design method is important.

ACKNOWLEDGEMENT

The r-Truncated Benes Networks and Their Randomized Routing Algorithms

Hoda El-Sayed and Abdou Youssef
Department of Electrical Engineering and Computer Science
The George Washington University
Washington, D.C. 20052

Abstract

Benes networks have the potential for balanced traffic, fewer conflicts, and can route any permutation in one pass due to their multiplicity of paths. Omega networks , on the other hand, have fast set-up and low hardware cost, but could take more than one pass to route a permutation. This paper introduces a new class of networks referred to as r-truncated Benes, which is a Benes network with r randomization stages eliminated. Using randomized routing, we will show that r-truncated Benes networks is an excellent trade-off between Omega and Benes networks. In particular, it will be shown that the 1-truncated Benes network outperforms Omega and is also superior to Benes and other truncated Benes networks in cost and performance.

1. INTRODUCTION

The study of interconnection networks has become one of the most popular research areas in parallel processing, as communications overhead is one of the most important factors affecting the performance of parallel computer systems. Multistage interconnection networks (MINs) [6,4,16] can be divided into three classes: blocking, nonblocking, and rearrangeable networks [5]. In blocking networks, simultaneous connections of more than one terminal pair, input and output, may result in conflicts over communication links. Examples of blocking networks are Omega[7] and baseline networks. Nonblocking networks, such as certain Clos [3] networks, can handle all possible one-to-one connections without conflict. A network is rearrangeable if it can perform all possible connections between inputs and outputs by rearranging its existing connections so that a connection path for a new input-output pair can be established. Benes network is a well-known network that belongs to this class [2]. There are two main disadvantages of Bene/Clos networks namely, high hardware cost and slow routing speed. Although Benes/Clos networks have received less attention due to their disadvantages, recent advances in

technology and routing techniques have largely eliminated those disadvantages, thus making Benes/Clos networks competitive alternatives worthy of further study. Specifically, large crossbar switches can be built affordably due to the advances in VLSI technology. An implementation of optical Clos network has been recently reported by Lin, Krile, and Walkup [11]. In routing, a recent study by Youssef [18] has developed efficient universal randomized self-routing algorithms for Clos networks.

The two main disadvantages of the Clos/Benes networks have been Several other parallel algorithms were developed that take less time. Lee developed a Benes control algorithm that takes O(N) parallel time, which is still slow [8]. Nassimi and Sahni came up with a self-routing parallel algorithm for fast set-up in switches that takes O(logN) time; however, the algorithm does not work for all permutations [13]. Lenfant followed another approach that restricts the algorithm to a set of frequently used bijections. Hence the algorithm still does not realize all classes of permutation [9]. Lev , Pippenger, and Valiant implemented an algorithm that takes $O(N\log^2 N)$ serial time for one processor and $O(\log^2 N)$ parallel time [10]. Later, Nassimi and Sahni developed another parallel routing algorithm for Benes networks. The algorithm routes Lenfant class of permutations and takes $O(\log^2 N)$, and does realize all permutations. Youssef and Arden introduced a new approach for fast routing control. The algorithm self-routes several classes of frequently used permutations, and takes $O(\log^2 N)$ time to route arbitrary permutations on Benes-Clos networks [17]. Raghavendra and Boppana developed a self-routing algorithm for self-routing the linear class of permutations, but does not realize all permutations[15]. Mitra and Cieslak [12] introduced a randomized routing scheme on extended Omega networks. Their algorithm, however, uses single randomization and in this paper we use multiple randomization on r-truncated Benes networks.

Omega networks are more frequently used due to their fast routing and low hardware cost. Those networks, however, are blocking and therefore could take more than one pass (which means longer delay) in routing a permutation.
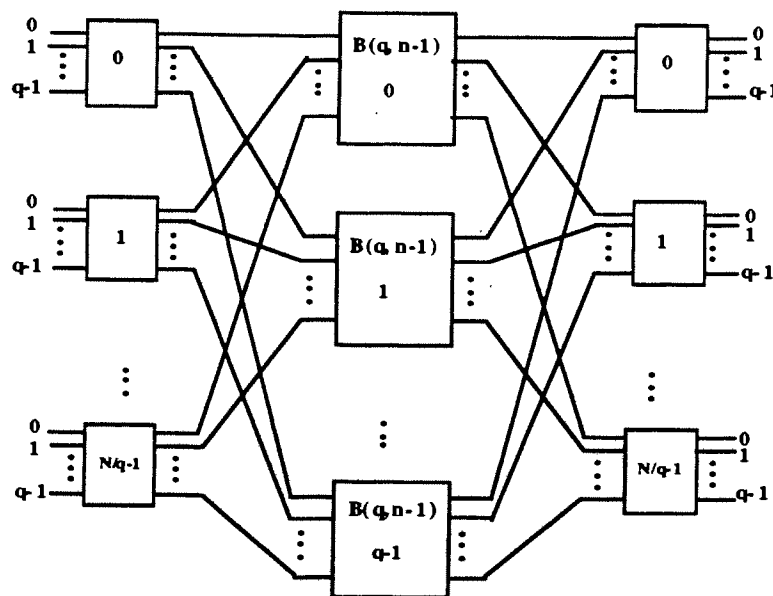
Figure 1. Benes Network B(q , n)

This is mainly due to their single path property, which leaves no room for randomization. On the other hand, Benes networks realize all permutations in single passes. Benes networks[2], however, have more stages than Omega networks[7]. This has given rise to having multiple paths between any pair of nodes in a Benes network, which allows randomization. On the other hand, the increased number of stages caused Benes networks to have longer routing delay and more hardware cost than Omega networks. Due to those conflicting properties, cost of hardware and the ability of randomization, we are motivated to study a new network. This network, which is called here the r-truncated Benes network, is a trade-off between Omega and Benes networks.

Our performance evaluation shows that with only one stage of randomization, an r-truncated Benes network,, with r = 1, is superior to comparable Omega networks. The rest of the paper is organized as follows. The next section gives a brief overview of r-truncated Benes networks. Section 3 will discuss the routing algorithm used on r-truncated Benes networks. Section 4 will present the details of the performance analysis. Section 5 contains the conclusions and some ideas for future work.

2. OVERVIEW OF R-TRUNCATED BENES NETWORK

A Benes network, $B(q,n)$, has $N\ (= q^n)$ inputs and N output terminals, and consists of $2n-1$ stages of N/q crossbar switches each, where each switch is size $q \times q$ crossbar. Such network can be defined recursively. For $n = 1$, $B(q,1)$ is a single $q \times q$ crossbar. For $n \geq 2$, the recursive structure is shown in Figure 1 specifically, the middle stage consists of q copies of $B(q,n-1)$ numbered $0,1,...,q-1$ from top to bottom; the first stage connects the i-th output port of the j-th switch (in the first column) to the j-th input of the i-th $B(q,n-1)$ of the middle stage; and the last stage is the mirror image of the first stage. The stages are numbered from 1 to $2n-1$ from left to right.

In this paper a high-performance routing algorithm for r-truncated Benes networks is developed through randomized self-routing algorithms. In this randomized approach, switches that are in stages from 1 to $n-1$ are set randomly. On the other hand, switches that are in the stages n through $2n-1$ of the network are set according to a self-routing mechanism using the destination address. It is known that after a message crosses the first $n-1$ stages, the message can be self-routed using the destination address. This can be illustrated by considering a three stage Benes network $B(q,2)$ where the input or output y of a switch x in any stage has a global label $[xy]$.

r-Truncated Benes Network
Using Multiple Randomized Circuit Switching
On Random Permutations With N = 4096 Processors
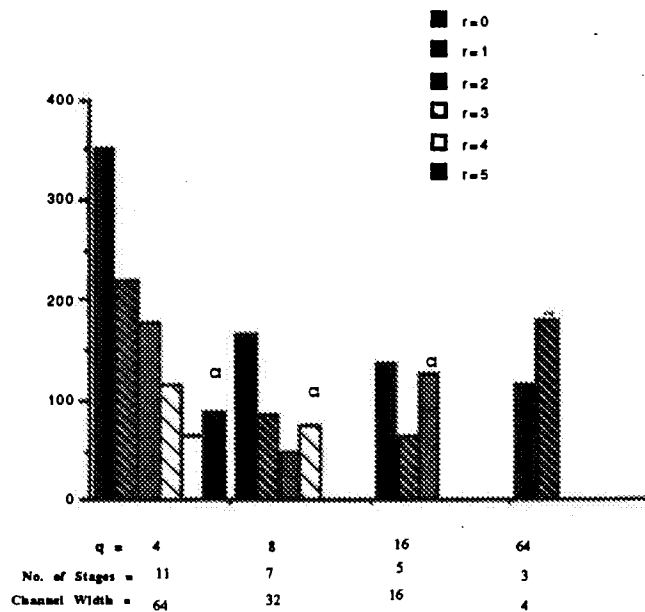
Figure 2. r-Truncated Benes Network Using Multiple Randomized Circuit Switching On Random Permutations With N = 1024 Processors

Suppose that a message is to be sent from output port $[xy]$ in the first stage, to the output terminal $[x'y']$ in the last stage. The message first enters the input port $[yx]$ in the middle stage, then, using the digit x' of the destination address $[x'y']$, the message exits through output port $[yx']$ in the middle stage. Afterwards, it enters the input port $[x'y]$ of the last stage and then using digit y' of the destination address $[x'y']$, it reaches output port $[x'y']$ of the last stage, which is the desired destination. Consequently, to go from any input terminal $[vu]$ of $B(q,2)$ to any output terminal $[x'y']$, we can select a digit z', $0 \leq z' \leq q\text{-}1$, and form the control tag $z'x'y'$ to find a path from $[vu]$ to $[x'y']$ in $B(q,2)$ [17]. The same is applied generally to any number of stages in $B(q,n)$. To go from a source s to a destination d, s forms a control tag $c_1c_2...c_{2n-1}$ where the part $c_1c_2...c_{n-1}$ is selected uniformly randomly and is called the *randprt*, and the part

$c_nc_{n+1}...c_{2n-1}$ is the destination address d in q-ary, and is called the *fixedprt*. Every c_i is a q-ary digit and is used by stage i to exit the paths through output port c_i of the appropriate switch.

A truncated Benes network is derived mainly from a Benes network and we will refer to it as an r-truncated Benes $B(q,n,r)$ network. By deleting the first r stages of a Benes(q,n), where $0 \leq r \leq n\text{-}1$, we obtain the r-truncated Benes network. When $r = n\text{-}1$, the network becomes Omega, and when $r = 0$, the network is a Benes network. Thus, in a 5-stage Benes network B(2,3), with a 2x2 switch size, the first two stages are for randomization.

3. RANDOMIZED ROUTING ON R-TRUNCATED BENES

78

r-Truncated Benes Network
Using Multiple Randomized Circuit Switching
On Random Permutations With N = 1024 Processors
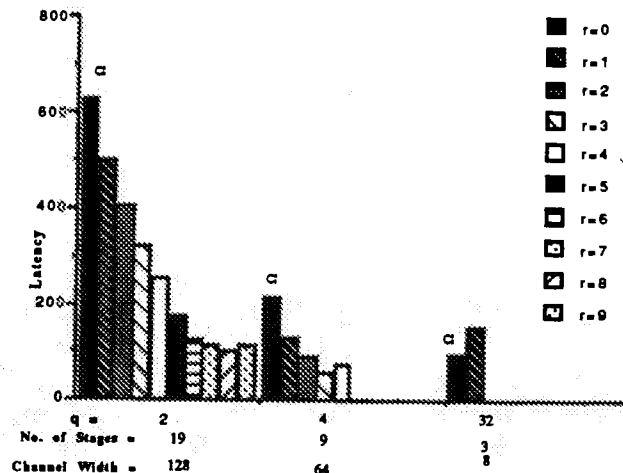
Figure 3. r-Truncated Benes Network Using Multiple Randomized Circuit Switching On Random Permutations With N = 4096 Processors

Multiple randomized circuit-switched routing is used with different values of r (number of stages to be truncated) to determine the optimal value of r. In this randomization approach, switches that are in stages 1 to $n-1$ $-r$ are set randomly and r is any value $0 \leq r \leq n-1$. The number of stages for randomization in an r-truncated Benes network is $n-1-r$ as opposed to $n-1$ stages in a Benes network. On the other hand, switches that are in the remaining stages of the network are set according to a self-routing mechanism using the destination address. In multiple randomization, new output ports are selected randomly for the stages $1 \leq stage \leq n-1$ for each message after each failed attempt to send the message due to a conflict. Randomized routing is performed on SIMD routing (permutations) for different values of r.

4. PERFORMANCE EVALUATION

The following parameters are used in the simulation:
N = machine size (number of input/output processors)
L = length of the message (in flits)
M = message size (in bits)
p = number of pins on each side of a switch
q = number of ports on each side of a switch (qxq is the logical switch size)
W = width of the channel (i.e., number of wires per channel, where each wire
            holds one bit)
Note that $p = qw$ and $L = M/W = Mq/p$

For practical considerations, we have considered the following values for $p, N,$ and M :
* $p = 256$ pins
* $N = 1024$ and 4096
* $M = 128$ bits for $N = 1024$, and $M = 64$ bits for $N = 4096$.

Accordingly, $L = \frac{q}{2}$ for $N = 1024$, and $L = \frac{q}{4}$ for $N = 4096$. Also recall that $(N = q^n)$

Using this set simulation set up, we study permutation routing using randomized routing on r-truncated Benes networks to determine the optimal value of r. This study does not only compare randomized routing on Omega versus Benes, but also quantifies the trade-off and identifies the optimal hybrid.

In the simulation, latencies are measured for several permutations with respect to the different parameter values of $q, n, L,$ and r. Each permutation is performed using multiple randomized circuit switching. Figures 2 and 3 show the average latencies determined for several different random permutations when $N = 1024$ and $N = 4096$, respectively, as a function of the different values of r (truncated stages) for different networks. With $N = 1024$, the network is capable of having 3-stages, 9-stages, and 19-stages, with different channel width and switch sizes. From the graph in figure 2, where the permutations are randomly chosen with $N = 1024$, the following observations are made. The latencies decrease as the number of stages decreases for $r = 0$ to $r = n-2$, and then moves up when $r = n - 1$ (corresponding to Omega). For

the 3-stage network, r can be either 0 or 1. The latency is lower when $r = 0$, than when $r = 1$. As mentioned before, when $r = 0$, the network is a Benes network, while with $r = 1$, the network becomes an Omega. In other words, keeping the randomized stage will give better performance in the network. For the 9-stage network, r can have different values from 0 up to 4. The latency has the lowest value when $r = 3$, that is, when having, only one stage for randomization. For the 19-stage network , r ranges from 0 to 9. It is obvious from the graph that the latency decreases as the number of eliminated stages increases (r values). However, the latency has the lowest value when $r = 8$, which corresponds to only one stage for randomization. The optimal design parameter with $N = 1024$ on random permutations is with a 4x4 switch size in a 9-stage network and the optimal r is with $r = 3$.

From the graph of figure 4, the following observations are made. In the 3-stage network, $r = 0$ has lower latency than when $r = 1$, which is the Omega. For the 5-stage network, $r = 1$ has the lowest latency, that is, with one stage for randomization. In the 7-stage network, $r = 2$ has the lowest latency, that is lower than the Omega when $r = 3$. In the 11-stage network, $r = 4$ has the lowest latency, which is with one stage of randomization.

The optimal design parameter is with an 8x8 switch size of 7-stages, and the optimal value of r is with $r = 2$.

5. CONCLUSIONS

In this paper we introduced and studied a new network called the r-truncated Benes network as a trade-off between Benes networks and Omega networks. Simulations results show that with only one stage of randomization, the message delay is even lower than the Omega network and the r-truncated network becomes superior to Omega as well as other r-truncated Benes for r > 1. This is in addition to the multiplicity of paths in the r-truncated Benes which can also offer fault tolerance. This demonstrates that the traffic balancing and performance benefit from randomization is fully achieved with one stage of randomization. Adding more stages, simply increases the latency without offering any additional performance benefits. Therefore, it is highly recomended that high-performance. It was also observed that when using random permutations for *1024* processors, the 4x4 switch size is optimal, and for *4096* processors, the 8x8 switch size is optimal. Randomized routing has also been used on r-truncated Benes networks using uniform MIMD traffic, and our preliminary results show that Omega performs better than Benes and r-truncated Benes network. That is due to the fact that the network traffic is already uniform and randomization will be of no use in this case (as randomization provides natural traffic load balancing). It is anticipated, however,

that when using non-uniform MIMD traffic, r-truncated Benes network will perform better than Omega.

REFERENCES

[1]   G.B. Adams III, "Fault-Tolerant Multistage Interconnection Networks, " *Computer*, vol. 20, no. 6, pp. 14-27, June 1987.

[2] V.B. Benes, *Mathematical theory on connecting networks and telephone traffic*, Acad. Press, NY, 1965.

[3] C. Clos, "A Study of Non-blocking Switching Networks," *Bell Sys. Tech Jourr*, vol. 32, pp. 406-424, 1953.

[4] T. Feng, " A Survey of Interconnection Networks," *Computer* vol. 14, pp. 12-27, 1981.

[5] K. Hwang and Briggs, *Computer architecture and parallel processing*," McGraw-Hill, NY, 1984.

[6]   Kai Hwang, *Advanced computer architecture*, McGraw-Hill, NY, 1993.

[7] D. H. Lawrie, "Access and Alignment of Data in an Array Processor," *IEEE TC*, C-24, pp. 1145-1155, 1975.

[8] K.Y.Lee, "A new Benes network control algorithm," *IEEE TC.*, vol.c-36, no. 6, pp. 768-772, June 1987.

[9] J. Lenfant, "Parallel permutation of data: A Benes network control algorithm for frequently used permutations, " *IEEE TC.*, vol. c-27, no. 7, pp. 637-647, July 1978.

[10] G.F. Lev, N. Pippenger, and L.G. Valiant, " A fast parallel algorithm for routing in permutation networks, " *IEEE TC.* vol. c-30, no. 2, pp. 93-100, Feb. 1981.

[11]   Shing-Hong Lin, T.F.Krile, and J.F. Walkup, "Two-dimensional Optical Interconnection Network and its Uses," *Applied Optics*, vol. 27, no. 9, 1988.

[12] D. Mitra and R.A. Cieslak, "Randomized parallel communications on an extension of the Omega Network", *J. ACM*, vol. 34, no. 4, pp. 802-824, Oct. 1987.

[13] D. Nassimi and S. Sahni, "A self-routing Benes network and parallel permutation algorithms, " *IEEE TC.*, vol. c-30, no.5, pp. 332-340, May 1981.

[14] D. Nassimi and S. Sahni, "Parallel algorithms to set up the Benes permutation network, "*IEEE Trans. Comput.*, vol.c-31, no.2, pp. 148-154, Feb. 1982.

[15] C.S. Raghavendra and R.V. Boppana, "On self-routing in Benes and Shuffle-Exchange networks, " *IEEE TC .*, vol. 40, no. 9, pp. 1057-1064, Sept. 1991.

[16] C. Wu and T. Feng, "On a class of multistage interconnection networks," IEEE Trans. Comput., vol. c-29, pp.694-704, Aug. 1980.

[17] A. Youssef and B. Arden, "A new approach to fast control of $r^2 x r^2$ 3-stage Benes networks of rxr crossbar switches," Proc. 17th Intl'l Symp. Comp. Arch., Seattle, pp.50-59, May 1990.

[18] A. Youssef, "Randomized Self-Routing Algorithms for Clos Networks," *Computers & Electrical Engineering, a International Journal*, vol. 19, no.6, pp.419-429, 1993.

Balancing Interconnect and Computation in a Reconfigurable Computing Array
(or, why you don't really want 100% LUT utilization)

André DeHon

Berkeley Reconfigurable, Architectures, Software, and Systems
Computer Science Division
University of California at Berkeley
Berkeley, CA 94720-1776
<andre@acm.org>

Abstract

FPGA users often view the ability of an FPGA to route designs with high LUT (gate) utilization as a feature, leading them to demand high gate utilization from vendors. We present initial evidence from a hierarchical array design showing that high LUT utilization is **not** directly correlated with efficient silicon usage. Rather, since interconnect resources consume most of the area on these devices (often 80-90%), we can achieve more area efficient designs by allowing some LUTs to go unused—allowing us to use the dominant resource, interconnect, more efficiently. This extends the "Sea-of-gates" philosophy, familiar to mask programmable gate arrays, to FPGAs. Also introduced in this work is an algorithm for "depopulating" the gates in a hierarchical network to match the limited wiring resources.

1 Introduction

The ability of an FPGA to support designs with high LUT usage is regularly touted as a feature. However, high routability across a variety of designs comes at a large expense in interconnect costs. Since interconnect is the dominant area component in FPGA designs, simply adding interconnect to achieve high LUT utilization is not always area efficient. In this paper, we ask:

- *Is an FPGA with higher LUT usage more area efficient than one with lower LUT utilization?*

- That is: *Is LUT usability directly correlated with area efficiency?*

Our results to date suggest that this is often not the case— achieving high LUT utilization can often come at the expense of greater area than alternatives with lower LUT utilization. While additional interconnect allows us to use LUTs more heavily, it often causes us to use the interconnect itself less efficiently.

To answer this question, we proceed as follows:
1. Define an interconnect model which allows us to vary the richness of the interconnect.
2. Define a series of area models on top of the interconnect model to estimate design areas.
3. Develop an algorithm for mapping to the limited wiring resources in a particular instance of the interconnect model.

4. Map circuits to a range of points in the interconnect space, and assess their total area and utilization.
5. Examine relationship between LUT utilization and area.

2 Relation to Prior Work

Most traditional FPGA interconnect assessments have been limited to detailed population effects [1] [15]. In particular, they let the absolute amount of interconnect (*i.e.* number of wiring channels or switches) float while assessing how closely a given population scheme allows detailed routing to approach the limit implied by global routing. They also assume that the target is to fully populate the LUTs in a region of the interconnect.

Instead, we take the viewpoint that a given FPGA family will have to have a fixed interconnect scheme and we must assess the goodness of this scheme. To make maximum use of the fixed interconnect, in regions of higher interconnect requirements where the design is more richly connected than the FPGA, we may have to use the physical LUTs in the device sparsely resulting in a depopulated LUT placement. This represents a "Sea-of-Gates" usage philosophy as first explored for FPGAs in University of Washington's Triptych design [4].

For the sake of illustration, consider a design which has a small, but heavily interconnected controller taking up 20% of the LUTs in the design. The rest of the design is a more regular datapath which does not tax interconnect requirements. If we demanded full population, we would look at the interconnect resources necessary to fully pack the controller, and those requirements would set the requirements for the entire array. However, the datapath portion of the chip would not need all of this interconnect and consequently would end up with much unused interconnect. Alternately, we can spread out the controller, ignoring some LUTs in its region of placement, so that the whole FPGA can be built with less interconnect. Now, the controller may take up 30% of the device resources since it cannot use device LUTs 100% efficiently, but the whole device is smaller since it requires less interconnect.

Recently, NTT argued for more wires and less LUTs [17], and HP argued for rich interconnect which will meet or exceed the requirements of logic netlists [2]. Earlier Triptych showed density advantages over traditional alternatives with partially populated designs [4]. The NTT paper examined two points in the space, while HP and University of Washington each justified a single design point. In this paper, we build a model which allows us to explore the tradeoff space more broadly than a few isolated design points. The model is based on a hierarchical network design and captures the dominant switch and wire effects dictating wire area. This generalization, of course, comes at the cost of modeling the design space more abstractly than a particular, detailed FPGA design.

Figure 1: Tree of Meshes

We will be using a hierarchical interconnect scheme as the basis of our area model. Agarwal and Lewis's HFPGA [1] and Lai and Wang's hierarchical interconnect [11] are the most similar interconnect schemes proposed for FPGA interconnect. As noted above neither of these studies made an attempt to fix the wiring resources independent of the benchmark being studied as we are doing here. To permit a broad study of interconnect richness, our interconnect scheme is also defined in a more stylized manner as detailed in the next section.

3   Interconnect Model

The key requirements for our interconnect model is that it:
- represent interconnect richness in a parameterized way
- allows definition of a reasonable area model

To meet these goals, we start with a hierarchical model based on Leighton's Tree of Meshes [13] or Leiserson's Fat Trees [14]. That is, we build a tree like interconnect where the bandwidth grows toward the root of the tree (See Figure 1). We use two parameters to describe a given interconnect scheme:
1. c = the number of base channels at the leaves of the tree
2. p (α) = the growth rate of interconnect toward the root

Note that we realize p by using one of two kinds of stages in the tree of meshes:

- non-compressing (2:1) stages where the root wires are simply equal to the sum of root wires from the two children so there is no net bandwidth reduction

- compressing (1:1) stages where the root wires are the same as each of the root wires from the children, so that only half of the total children wires can be routed upward

By selecting a progression of these stages we can approach any bandwidth growth rate (See Figure 4).

If we use a repeating pattern of stage growths, we approximate a geometric bandwidth growth rate. That is, a subtree of size $2 \cdot n$ has 2^p times as much bandwidth at its root as a subtree of size n, or every tree level has $\alpha = 2^p$ more wires than its immediate children. This is roughly the model implied by Rent's Rule [12] ($IO = c \cdot N^p$). More precisely, it represents a bifurcator as defined by Bhatt and Leighton [3] (See Figure 2).

Intuitively, p represents the locality in interconnect requirements. If most connections are purely local and only a few connections come in from outside of a local region p will be small. If every gate in a region had a unique signal coming from outside the region, then $p \to 1.0$. So think of p as describing how rich our interconnect needs to be. If $p = 1$, we are effectively building a crossbar with no restrictions. If $p = 0$, we are building a 1D systolic array or pure binary tree whose IO bandwidth does not grows as the array grows.

Figure 2: (F,α)-bifurcator

4   Area Effects

For our basic area model, we perform a straightforward layout of the elements shown in Figure 4. That is, we have:
- Logic Block of size A_{pe}
- Switches of size A_{sw}
- Wires of pitch WP

Each subtree is built hierarchically by composing the two children subtrees and the new root channel. Channel widths are determined by either the area required to hold the switches or the width implied by the wire channels, depending on which is greater. We assume a dedicated layer for each of horizontal and vertical interconnect. The result is the "cartoon" VLSI layout as shown in Figure 3.

Figure 3: "Cartoon" Layout of Hierarchical Interconnect

Note that the number of base channels (c) is 3 in all these examples.

Figure 4: Programming Growth for Tree of Meshes

Figure 5: c choose k LUT Input Population ($c = 5, k = 3$)

Typical values for an SRAM programmable device:[1]

- $A_{pe} = 40\text{K}\lambda^2$ — this would hold 16 memory bits for a 4-LUT ($16 \times 1.2\text{K}\lambda^2/\text{SRAM-bit} \approx 20\text{K}\lambda^2$) plus a the LUT multiplexor and optional output flip-flop ($13\text{K}\lambda^2$ in [5], $15\text{K}\lambda^2$ in [8]).
- $A_{sw} = 2.5\text{K}\lambda^2$ for a pass transistor switch (including its dedicated SRAM programming bit) — to model mask or antifuse programmable devices, we would use a much smaller size for this parameter.
- $WP = 8\lambda$ for the metal 2 or metal 3 wire trace and spacing

We assume the channels are populated with c choose k input selectors [7] on the input and have a fully populated output connection (See Figure 5). Switch boxes are either fully populated or linearly populated (see Figure 6) with switches.

Figure 7 shows cartoon layouts for 3 different choices of p, highlighting the area implied by each choice. Two things we can observe immediately from this simple model comparison:

- For reasonable parameters, interconnect requirements dominate logic block area; *e.g.* at $c = 6$, $p = 0.67$, a design with 1024 LUTS has only 5% of its area in LUTs (estimated area per LUT including interconnect is $\approx 750\text{K}\lambda^2$) — while this is a simple area model, the area and ratio are not atypical of real FPGA devices; they are also consistent with prior studies (*e.g.* 6% for 600 4-LUT design in [5]).

Figure 6: Linear Switchbox Population for Hierarchical Interconnect

$p = 0.50$	$p = 0.67$	$p = 0.75$
Rel. Area = 0.25	Rel. Area = 0.37	Rel. Area = 1.00

Figure 7: Effects of p on Area at 1K LUTs

---

[1]λ = half the minimum feature size for a VLSI process. Assuming linear scaling of all features, λ^2 area should be the same across processes.

Figure 8: "Pulling" design up tree to match fixed wire schedule

- Interconnect parameter richness has a large effect on total area.
  To further build intuition, let's assume for a moment that a design can be perfectly characterized by a growth exponent p. If the growth exponent for the interconnect matches the growth of the design ($p_{interconnect} = p_{design}$), then the network will require minimum area. What happens if these two are not perfectly matched? There are two cases:

  - $p_{interconnect} > p_{design}$ — we have more interconnect than necessarily. The design can use all the LUTs in the network, but the network has more wires. As a result, the area per LUT is larger than the matched case—that is, mapping the design on the richer interconnect takes more area than the matched design case.

  - $p_{interconnect} < p_{design}$ — we have less interconnect than necessary. We cannot pack the design into a minimum number of LUTs in order to fit the design. Instead we must pull the design up the tree, effectively depopulating the logic blocks, until the tree provides adequate connectivity for the design (See Figure 8). As a result, we have leaves in the tree which are not fully utilized. As we will see, this also takes more area than the matched design case.

Figure 10 shows the area overhead required to map various designs onto interconnects with various growth factors. As we expect, it shows that the matched interconnect point is the minimum point with no overhead. As we go to greater or lesser interconnect offered by the network, the area overhead grows, often dramatically.

5   Design Requirements

In practice, of course, c and p values are a rough characterization of the interconnect requirements for a real design. With multiple subgraphs of a given size (subtrees at the same height in the tree) we get more than one I/O to subgraph relationship. Further, the growth is seldom perfectly exponential. Finally, even asking if a graph has an (F, α)-bifurcator is an NP-hard problem. So, the bifurcations

Figure 9: I/O versus Partition size graph for $\mathtt{i10}$

we construct are heuristic approximations biased by the tools we employ.

Figure 9 shows the I/O versus subgraph relationship for the one of the IWLS93 benchmark ($\mathtt{i10}$).
- Mapped for area with SIS [16] and Flowmap [6]
- Recursively bisected using a single Eigenvalue spectral partitioner

The recursive bisection approximates the natural bandwidth versus subtree sizes which exist in the design. We see the I/O to subgraph relationship is not 1:1. We also see that the max and average contours can be matched well to a geometric growth rate (*e.g.* Rent's Rule—average $c = 5, p = 0.7$; max $c = 7, p = 0.7$).

The left of Figure 11 shows the I/O versus subgraph relationship for all the IWLS93 benchmarks area mapped to 2000 or fewer LUTs using SIS, Flowmap, and spectral partitioning as above. On the right it shows the distribution of Rent parameter estimates for these benchmarks. Here we see that while we may be able to pick "typical" c and p values, there is a non-trivial spread in interconnect requirements across this set of designs.

6   Mapping to Fixed Wire Schedule

We have now seen that we can define a parameterized interconnect model with a fixed wire schedule. Designs have their own requirements which do not necessarily match the fixed wire schedule available from a device's interconnect. When the device offers more interconnect than a design needs, mapping is easy, we simply place the design on the interconnect and waste some wires. However, if the design has more interconnect needs than the device provides, how do we map the design to the device?

As suggested in Figure 8, we can start with the recursively bipartitioned design and simply pull the whole design up the tree until all the interconnect wires meet or exceed the design requirements. However, keeping the groupings originally implied by the recursive bisection is overly strict. In particular, re-associating the subgraphs based on interconnect availability can achieve tighter packings (See Figure 13). That is, we do not really want a bisection of the LUTs, but a bisection of the total capacity including both interconnect and LUTs. Intuitively, the size of a subgraph is determined by the greater of its LUT requirement and its interconnect requirement relative to the fixed wire schedule of the device.

To attack the problem of regrouping subtrees to fit into the fixed wire schedule, we introduce a dynamic programming algorithm which determines where to split a given subgraph based on the available wire schedule. That is, we start with a linear ordering of LUTs. Then, we ask where we should cut this linear order-

Figure 10: Theory: effects of mismatched between interconnect network p and design requirements

Figure 11: I/O versus Partition size graph for Benchmark Set

```
// size[start,finish] represents the smallest subtree which will
//        contain the set of LUTs between position start and finish
// uniqueio(o,i,j) returns the number of unique nets which appear both in the subrange i→j,
//        and outside of that range
o = order all LUTs
for i=0 to o.length
        size[i,i] ← size(1,unique(o,i,i)) // base case ≡ single LUT subtrees
for len=2 to o.length
        for start=0 to o.length-len // process all subranges of specified length
                minsize=MAX
                end=start+len-1
                isize=uniqueio(o,start,end)
                for mid=start+1 to end // search for best split point
                        msize=1+max(size[start,mid],size[mid+1,end])
                        size=max(msize,iolevel(isize))
                        minsize=min(size,minsize)
                size[start,end]←minsize
// final result is size[0,o.length-1]
```

Figure 12: Dynamic Programming Algorithm to Map to Fixed Wire Schedule

5

Figure 13: Re-associating Subgraph clusters to match Fixed Wire Schedule

ing of LUTs into two subtrees in order to minimize the total area required—typically, minimizing the heights of the two subtrees. Each of the subtrees are then split in a similar manner. To make the decision of where to cut a subtree, we examine all cut points. As long as we have a single linear ordering for LUTs, this is very similar to the optimal parenthesis matching problem. In a similar manner, we can solve this problem with a dynamic programming algorithm.

The dynamic programming algorithm (Figure 12) finds the optimal subtree decomposition **given** the initial LUT ordering. The trick here, and the source of non-optimality, is picking the order of the LUTs. For this we use the 1D spectral ordering based on the second smallest Eigenvalue which Hall shows is the optimal linear arrangement to minimize squared wire lengths [10].

Figure 14 shows why the single linear ordering is non-optimal. Here we see a LUT B placed to minimize its distances to A, C, and D. The order is such as to keep B, C, and D together for cut 3. However, if we take cut 4, then it would be more appropriate to place B next to A since we have already paid for the wires to C and D to exit the left subgroup. However, as long as we are using a single linear ordering, we do not get to make this movement after each cut is made. In general to take proper account of the existing cut, we should reorder each of the subgraphs ignoring ordering constraints originally imposed by the wires which have already been cut.

To avoid this effect, we would have to reorder each subtree after each cut is made. In addition to increasing the complexity of each cut, this would destroy the structure we exploited to apply dynamic programming—that is, the sub-problems would no longer be identical. Of course, since the spectral partitioning does not even give an optimal cut point for the bisection problem, the ordering effect alone is not the only element of non-optimality here.

There is certainly room for algorithmic improvement here. The

Figure 14: Example showing the limitations of a Single Linear Ordering

results are, nonetheless, good enough to give us interesting depopulations as we will see in the next section.

7   Results from Mapping

Putting it all together:

- Start with the area targeted SIS and Flowmap 4-LUT networks for the IWLS93 benchmarks under 2000 4-LUTs.
- Order using the second smallest Eigenvalue.
- Map to fixed schedule with the dynamic programming algorithm; The results are shown in terms of relative number of LUTs in the top left of Figure 15.
- Apply an area cost model such as shown in top right of Figure 15.
- Result is the relative area map shown at the bottom of Figure 15.

Figure 15 shows that there is a minimum area point across the benchmark set. For our linear switch population model, this occurs at $c = 6$, $p = 0.6$. As our theory predicts, too much interconnect and too little interconnect both account for area overheads over the minimum. Notice that the only points where the entire benchmark achieves full utilization are $c = 10$, $p \geq 0.75$ and $p = 0.8$, $c \geq 7$, all points which are above the minimum area point.

Table 1 examines the effects of picking a particular point in the c-p-design space. For each design in the benchmark set, we can compute the c,p-point which has minimum area. We can then look at the overhead area required between the "best" c, p, picked for the individual design, versus the best c, p for the entire benchmark under certain criteria. For the linear switch population case, we see that average overhead between the benchmark minimum and each benchmark's best area is only 23% and that corresponds to an average LUT utilization of 87%. Similarly, we see that picking the smallest point where we get 100% device utilization results in almost 200% area overhead. We see different absolute numbers, but similar trends with other area models.

Given the range of partition ratios and cut sizes we saw in Figure 11, it is not that surprising that the full utilization point is excessive for many designs and leads to many area inefficient implementations. Figure 16 shows a slice in p-space for the single design i10 whose I/O versus subgraph size curve we showed in Figure 9. Notice that even for this single design, the minimum area point does not correspond to full utilization. In fact, the minimum area point is actually only 50% of the area of the full utilization point. So, even for a single design allowed to pick the network parameters c, p which minimizes device area, full LUT utilization does not always correspond to better area utilization. We see here that the effects of varying wire requirements, which we described in Section 2, do actually occur in designs.

In the previous section, we noted that the fixed wire schedule mapping algorithm in use is not optimal. It is worth considering how a "better" algorithm would affect the results presented here. A "better" algorithm could achieve better LUT utilization for the points where depopulation occurs. For the points on the graph where no depopulation occurs, a better algorithm could offer no improvement. As a result, we expect a better algorithm to magnify these effects—making the depopulated designs tighter and take less

Figure 15: Area Utilization Results Mapping Benchmark to Fixed Wire Schedules

Figure 16: p-space slice for i10 showing that area minimization is not directly correlated with high LUT usage

7

Wire Dominated $WP = 8\lambda$, $A_{sw} = 64\lambda^2$

| Minimization | params | | Sigma | Max | LUT |
Objective	C	P	relative area	relative area	Utilization
relative area	6	0.6	1.30	2.99	0.87
max relative area	6	0.65	1.40	1.91	0.93
area with full utilization	10	0.75	3.23	6.94	1.00

Linear $WP = 8\lambda$, $A_{sw} = 2500\lambda^2$

| Minimization | params | | Sigma | Max | LUT |
Objective	C	P	relative area	relative area	Utilization
relative area	6	0.6	1.23	2.84	0.87
max relative area	6	0.65	1.24	2.38	0.89
area with full utilization	10	0.75	2.98	4.87	1.00

Switch Dominated (Quadratic)
$WP = 8\lambda$, $A_{sw} = 2500\lambda^2$

| Minimization | params | | Sigma | Max | LUT |
Objective	C	P	relative area	relative area	Utilization
relative area	6	0.6	1.32	3.50	0.87
max relative area	4	0.65	1.47	2.31	0.49
area with full utilization	10	0.75	4.25	11.5	1.00

Table 1: Compare Effects of Various Network Selection Points

area, while the full utilization designs stay at roughly the same point.

8 Limitations and Future Study

We have only scratched the surface here. As with any CAD effort where we are solving NP-hard problems with heuristic solutions there is a significant tool bias to the results. Flowmap was not attempting to minimize interconnect requirements, and there is a good argument that LUT covering and fixed-wire schedule partitioning should be considered together to get the best results. At the very least, it would be worthwhile to try different LUT mapping strategies to assess how much these results are effected by LUT covering.

The area model used assumes a purely hierarchical, 2-ary interconnect. Two things one would like to explore are (1) the effects of different arity (flattening the tree) and (2) the introduction of shortcut connections (*e.g.* Fat Pyramid [9]). The shortcut connections will tend to reduce the need for bandwidth in the root channel and may shift the balance in interconnect costs. Further, shortcuts appear essential for delay-mapped designs, which we have also not studied here.

We suspect the hierarchical model captures the high-level requirements of any network, but it will be interesting to study these effects more specifically for mesh-based architectures. The key algorithmic enabler needed for both shortcuts and mesh-based architectures is to identify good heuristics for spreading in two dimensions rather than the one-dimensional approach we exploited here.

An important assumption we have made here is that interconnect growth is geometric (power law). The c, p estimates shown in Figures 9 and 11 support the fact that a geometric growth relationship seems fairly reasonable. Nonetheless, we have not directly explored wire-schedules which deviate from strict geometric growth, and there may be better schedules to be found outside of the strict geometric growth space explored here.

We concentrated on global wiring requirements here and have not focussed on detailed switch population. The robustness of the general trends across different area and population models shown in Table 1 suggests that the major effects identified here are independent of the switch population details. While this does show us the relative merits of a given interconnect richness within a particular population model, we cannot, however, make any conclusions about the relative merits of different population schemes without carefully accounting for detailed population effects in both the area model and routability assessment.

9 Conclusions

We see that wires and interconnect are the dominant area components of FPGA devices. We also see that the amount of interconnect needed per LUT varies both among designs and within a single design. Given that this is the case, we cannot use all of our LUTs and all of our interconnect to their full potential all of the time—we must underutilize one resource in order to fully utilize the other. If we focus on LUT utilization, we waste significant interconnect—our dominant area resource. This suggests, instead, it may be more worthwhile for us to focus on interconnect utilization even if it means letting some LUTs go unused. Answering our opening question, we see that higher LUT usage does not imply lower area and that LUT usability is not always directly correlated with area efficiency.

Acknowledgements

benchmark mapping to LUTs and developed the initial BLIF parser used for these experiments.

Feedback from Randy Huang, Nicholas Weaver, John Wawrzynek, and Eylon Caspi helped cleanup early drafts of this paper.

References

[1] Aditya A. Agarwal and David Lewis. Routing Architectures for Hierarchical Field Programmable Gate Arrays. In *Proceedings 1994 IEEE International Conference on Computer Design*, pages 475–478. IEEE, October 1994.

[2] Rick Amerson, Richard Carter, W. Bruce Culbertson, Phil Kuekes, and Greg Snider. Plasma: An FPGA for Million Gate Systems. In *Proceedings of the International Symposium on Field Programmable Gate Arrays*, pages 10–16, February 1996.

[3] Sandeep Bhatt and Frank Thomson Leighton. A Framework for Solving VLSI Graph Layout Problems. *Journal of Computer System Sciences*, 28:300–343, 1984.

[4] Gaetano Borriello, Carl Ebeling, Scott Hauck, and Steven Burns. The Triptych FPGA Architecture. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 3(4):491–501, December 1995.

[5] Stephen D. Brown, Robert J. Francis, Jonathan Rose, and Zvonko G. Vranesic. *Field-Programmable Gate Arrays*. Kluwer Academic Publishers, 101 Philip Drive, Assinippi Park, Norwell, Massachusetts, 02061 USA, 1992.

[6] Jason Cong and Yuzheng Ding. FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs. *IEEE Transactions on Computer-Aided Design*, 13(1):1–12, January 1994.

[7] André DeHon. Entropy, Counting, and Programmable Interconnect. In *Proceedings of the 1996 International Symposium on Field Programmable Gate Arrays*. ACM/SIGDA, February 1996. Extended version available as Transit Note #128 <http://www.ai.mit.edu/projects/transit/transit-notes/tn128.ps.Z>.

[8] André DeHon. Reconfigurable Architectures for General-Purpose Computing. AI Technical Report 1586, MIT Artificial Intelligence Laboratory, 545 Technology Sq., Cambridge, MA 02139, October 1996. Anonymous FTP publications.ai.mit.edu:ai-publications/1996/AITR-1586.ps.Z.

[9] Ronald Greenberg. The Fat-Pyramid and Universal Parallel Computation Independent of Wire Delay. *IEEE Transactions on Computers*, 43(12):1358–1365, December 1994.

[10] Kenneth M. Hall. An r-dimensional Quadratic Placement Algorithm. *Managment Science*, 17(3):219–229, November 1970.

[11] Yen-Tai Lai and Ping-Tsung Wang. Hierarchical Interconnect Structures for Field Programmable Gate Arrays. *IEEE Transactions on VLSI Systems*, 5(2):186–196, June 1997.

[12] B. S. Landman and R. L. Russo. On Pin Versus Block Relationship for Partitions of Logic Circuits. *IEEE Transactions on Computers*, 20:1469–1479, 1971.

[13] Frank Thomson Leighton. New lower bound techniques for VLSI. In *Twethy-Second Annual Symposium on the Foundations of Computer Science*. IEEE, 1981.

[14] Charles E. Leiserson. Fat-Trees: Universal Networks for Hardware Efficient Supercomputing. *IEEE Transactions on Computers*, C-34(10):892–901, October 1985.

[15] Jonathan Rose and Stephen Brown. Flexibility of Interconnection Structures for Field-Programmable Gate Arrays. *IEEE Journal of Solid-State Circuits*, 26(3):277–282, March 1991.

[16] Ellen M. Sentovich, Kanwar Jit Singh, Luciano Lavagno, Cho Moon, Rajeev Murgai, Alexander Saldanha, Hamid Savoj, Paul R. Stephan, Robert K. Brayton, and Alberto Sangiovanni-Vincentelli. SIS: A System for Sequential Circuit Synthesis. UCB/ERL M92/41, University of California, Berkeley, May 1992.

[17] Atsushi Takahara, Toshiaki Miyazaki, Takahiro Murooka, Masaru Katayama, Kazuhiro Hayashi, Akihiro Tsutsui, Takaki Ichimori, and Ken nosuke Fukami. More Wires and Fewer LUTs: A Design Methodology for FPGAs. In *Proceedings of the 1998 International Symposium on Field-Programmable Gate Arrays*, pages 12–19, 1998.

A Mapped Benchmarks Statistics used for Experiment

Design	Mapped LUTs	Max c	Max p	Avg c	Avg p
5xp1	83	7	0.35	6	0.42
9sym	177	6	0.57	6	0.63
9symml	80	6	0.45	5	0.47
C1355	74	7	0.64	5	0.67
C17	2	0	0	0	0
C1908	136	7	0.56	5	0.56
C2670	218	6	0.7	4	0.75
C3540	382	7	0.54	6	0.56
C432	79	7	0.53	6	0.57
C499	74	7	0.63	5	0.67
C5315	590	7	0.66	5	0.69
C6288	522	8	0.44	6	0.47
C7552	723	7	0.63	5	0.68
C880	116	6	0.65	5	0.65
a	1	0	0	0	0
alu2	279	8	0.49	6	0.54
apex1	799	8	0.62	7	0.6
apex3	900	9	0.54	6	0.58
apex6	258	6	0.71	5	0.68
apex7	108	6	0.61	4	0.64
b1	4	4	0.40	3	0.67
b12	377	7	0.55	7	0.46
b9	56	6	0.62	4	0.69
bbara	34	6	0.43	5	0.44
bbsse	70	8	0.32	6	0.36
bbtas	9	5	0.25	4	0.3
beecount	21	6	0.43	5	0.49
c8	87	7	0.54	5	0.58
cc	33	5	0.65	4	0.73
cht	68	5	0.71	4	0.69
clip	243	7	0.53	6	0.44
clmb	369	2	1	2	1
cm138a	9	5	0.44	5	0.48
cm150a	15	5	0.67	4	0.75
cm151a	8	5	0.53	4	0.63
cm152a	11	5	0.41	5	0.43
cm162a	14	6	0.49	5	0.56
cm163a	12	5	0.60	5	0.65
cm42a	10	5	0.45	5	0.45
cm82a	4	4	0.5	4	0.5
cm85a	12	5	0.42	5	0.44
cmb	19	6	0.49	5	0.53
comp	45	6	0.52	5	0.57
con1	6	5	0.31	5	0.33
count	37	6	0.58	4	0.64
cps	821	9	0.62	6	0.67
cse	134	6	0.57	5	0.58
cu	24	6	0.51	5	0.56
daio	3	0	0	0	0
dalu	502	8	0.55	6	0.59
decod	18	5	0.49	4	0.52
dk14	51	7	0.35	7	0.24
dk15	31	6	0.39	6	0.22
dk16	171	8	0.42	6	0.47
dk17	30	6	0.38	5	0.41
dk27	5	5	0.15	4	0.35
dk512	25	6	0.17	5	0.22
donfile	1	0	0	0	0
duke2	274	8	0.48	6	0.53
e64	386	7	0.6	5	0.64
ex1	164	6	0.6	5	0.64
ex2	90	6	0.5	5	0.56
ex3	33	6	0.45	5	0.52
ex4	41	7	0.36	6	0.42
ex5	27	6	0.36	5	0.44
ex6	71	6	0.51	6	0.4
ex7	36	6	0.51	5	0.54
example2	139	6	0.69	4	0.74
f51m	98	7	0.53	5	0.58
frg1	282	6	0.62	6	0.51
frg2	744	8	0.61	5	0.65
i1	19	5	0.66	4	0.74
i10	906	7	0.68	5	0.71
i2	77	6	0.82	5	0.86
i3	46	5	0.88	5	0.89
i4	97	5	0.77	4	0.82
i5	153	6	0.67	4	0.75
i6	144	5	0.76	4	0.78
i7	215	6	0.71	4	0.77
i9	347	7	0.63	5	0.65
keyb	209	8	0.43	6	0.48
kirkman	133	8	0.35	6	0.37
lal	67	7	0.56	5	0.6
ldd	50	7	0.46	5	0.5
lion	3	0	0	0	0
lion9	5	0	0	5	0.1
majority	4	5	0.13	5	0.17
mark1	52	7	0.41	4	0.66
mc	9	5	0.21	4	0.33
misex1	24	6	0.37	5	0.41
misex2	54	6	0.57	5	0.59
mm30a	327	6	0.57	6	0.47
mm4a	118	9	0.28	7	0.31
mm9a	96	6	0.48	6	0.38
mm9b	120	6	0.51	5	0.53
modulo12	1	0	0	0	0
mult16a	32	5	0.36	4	0.36
mult16b	31	5	0.25	4	0.16
mult32a	64	5	0.39	4	0.38
mult32b	62	6	0.41	5	0.43
mux	45	6	0.51	6	0.47
my_adder	32	4	0.61	4	0.61
o64	46	5	0.84	4	0.88
opus	50	6	0.51	5	0.57
pair	567	8	0.6	5	0.65
parity	5	5	0.73	5	0.75
pcle	23	5	0.5	4	0.59
pcler8	33	5	0.6	4	0.68
planet	410	9	0.45	6	0.5
planet1	410	9	0.45	6	0.5
pm1	28	6	0.54	4	0.61
rd53	36	6	0.38	5	0.4
rd73	190	6	0.56	5	0.57
rd84	405	7	0.63	5	0.67
rot	467	7	0.65	5	0.71
s1	317	8	0.52	6	0.56
s1196	226	7	0.53	6	0.45
s1238	253	7	0.52	7	0.46
s1423	162	6	0.51	5	0.40
s1488	289	7	0.55	4	0.76
s1494	295	7	0.59	6	0.51
s1a	6	1	1	1	1
s208	27	6	0.34	5	0.29
s27	6	5	0.20	0	0
s298	40	6	0.35	5	0.41
s344	41	6	0.34	4	0.44
s349	41	6	0.34	4	0.44
s382	53	6	0.44	5	0.48
s386	74	7	0.32	6	0.36
s400	53	6	0.38	5	0.42
s420	62	7	0.31	5	0.39
s444	53	6	0.37	5	0.39
s510	105	7	0.45	6	0.47
s526	84	7	0.39	5	0.41
s526n	90	7	0.39	5	0.42
s5378	522	7	0.63	5	0.66
s641	79	5	0.6	4	0.68
s713	80	6	0.61	5	0.67
s8	1	0	0	0	0
s820	166	7	0.48	6	0.52
s832	169	7	0.48	6	0.52
s838	124	7	0.44	5	0.5
s9234	439	8	0.56	5	0.63
s953	182	7	0.5	6	0.54
sand	406	7	0.57	5	0.61
sao2	121	7	0.47	6	0.5
sbc	332	7	0.57	5	0.6
scf	663	9	0.53	6	0.57
sct	69	7	0.48	5	0.52
shiftreg	2	0	0	0	0
sqrt8	45	7	0.46	6	0.37
sqrt8ml	40	6	0.58	5	0.47
squar5	50	8	0.27	6	0.34
sse	70	8	0.32	6	0.36
styr	341	7	0.59	7	0.47
t481	735	8	0.55	6	0.6
table3	513	9	0.52	7	0.55
table5	522	7	0.66	7	0.53
tav	12	5	0.23	4	0.50
tbk	616	9	0.5	6	0.54
tcon	16	4	0.78	3	0.88
term1	246	8	0.48	6	0.52
train11	19	6	0.48	4	0.76
train4	3	0	0	0	0
ttt2	198	9	0.42	6	0.45
unreg	32	5	0.67	5	0.7
vda	517	8	0.55	6	0.58
vg2	277	9	0.42	6	0.47
x1	761	9	0.48	6	0.57
x2	23	6	0.39	5	0.53
x3	441	7	0.61	5	0.6
x4	294	7	0.61	5	0.63
xor5	21	6	0.38	5	0.46
z4ml	80	7	0.39	5	0.41
alu4	1756	8	0.58	6	0.63
apex4	1284	7	0.69	5	0.72
apex5	1241	9	0.63	6	0.66
cordic	1381	9	0.62	8	0.52
dsip	1175	7	0.65	6	0.64
ex5p	1348	9	0.61	6	0.64
i8	1242	8	0.57	5	0.60
k2	1138	8	0.69	6	0.58
seq	2004	10	0.53	7	0.58

B Lsize and Level

When mapping to a hierarchical array, or any array for that matter, one problem to address is how we count area used. Do we charge the design for the smallest tree hierarchy used? If so, we only get a logarithmic estimation of size. Designs which are slightly larger than a tree stage are charged the full cost of the next tree level. This could skew measures as ± 1 LUT at a power-of-two boundary has a big difference in metric, but elsewhere near factor-of-two differences hardly matter. For the data shown here, we have counted size in terms of the span of LUTs used (*lsize* — See adjacent diagram). That is, if we number the tree LUTs in a linear order; we pack starting at LUT 0 and use the position of the highest placed LUT to account for the capacity used. The LUTs above the last used subtree are all free. Intuitively, if we consume all of a subtree of size 128 and one more subtree of size 64, we still have a subtree of size 64 available for additional logic, so we charge the design to be only of lsize 192. In practice, when we use level as a metric instead of lsize, we see similar trends to those reported here but a larger benchmark-wide mismatch penalty, especially when requiring full population, due to the logarithmic granularity effects.

Level = 3

Lsize = 5

10

HYPERTREE
A MULTIPROCESSOR INTERCONNECTION TOPOLOGY

by

James R. Goodman and Carlo H. Sequin

Computer Sciences Technical Report #427

April 1981

HYPERTREE,
A MULTIPROCESSOR INTERCONNECTION TOPOLOGY

James R. Goodman and Carlo H. Séquin

Computer Science Division
Department of Electrical Engineering and Computer Sciences
University of California
Berkeley, California, 94720

ABSTRACT

A new interconnection topology for incrementally expansible multiprocessor systems is described which combines the easy expansibility of tree structures with the compactness of an n-dimensional hypercube. The addition of n-cube links to the binary tree structure provides direct paths between nodes which have frequent data exchange in algorithms such as sorting and Fast Fourier Transforms. The derivation of a family of such Hypertree structures is outlined, and the basic properties such as average path length, uniformity of the distribution of message traffic and routing algorithms are analyzed.

KEYWORDS: Multiprocessors, Communication networks, Message traffic, Routing algorithms, Tree structure, Hypercube.

1. INTRODUCTION

One of the problems that is still impeding the emergence of general purpose multiprocessors is interprocessor communication. For effective cooperation of several processors on the same task, or for fast access to distributed data, high communications bandwidth is typically required. If all processors are simply connected onto one and the same bus, this shared resource becomes a bottleneck preventing simultaneous communication between different pairs of processors, and the effective throughput of the system may actually go down as the number of processors is increased. A suitable interconnection network is thus needed which provides as much bandwidth as possible between any pair of processors. The classical crossbar switch between separate banks of processors and banks of memories, requires a high pin-to-logic ratio, and is therefore not very amenable to VLSI exploitation. One possible approach is to combine one processor and its memory with one node of the switching network, thus creating a regular network of computers.

1.1. Classical Switching Networks

Many people have addressed the problem of switching networks. Much work has been done in particular to allow multiple, simultaneous connections between processor banks and memory banks to permit sharing of data or concurrent cooperation on the same tasks. Among the better known networks are lattice structures [Bouknight 72], the flip net [Batcher 76], the Omega net [Lawrie 75], the indirect n-cube [Pease 77] [Benes 65], the perfect shuffle [Golomb 61] [Stone 71], the augmented data manipulator [Feng 74], the de Bruijn net [Schlumberger 74] [de Bruijn 46], the generalized connection network [Thompson 78], the Banyan partitioner [Goke 73], and the n-cube network [Batcher 76] [Siegel 77].

- 2 -

It is understood that one of the chief properties of many of these networks is the efficient interconnection of nodes in the n-dimensional hypercube, or n-cube, configuration. The n-cube is particularly compact. The worst case distance between any two nodes is only n, the dimension of the structure, and nodes which differ in their node addresses by only one bit are direct neighbors. In addition, this logical structure is extremely useful because of the wide range of algorithms that fit it particularly well. For many problems such as Fast Fourier Transforms or sorting, it is possible to map the logical structure of the problem directly onto the physical structure, and large quantities of data are exchanged between nearest neighbor pairs.

For the more general switching network, where many arbitrary pairs of elements may wish to communicate simultaneously, the n-cube topology is also suitable. The routing algorithm to reach a particular node is trivial: For every bit in the target node address which differs from the current node address, a corresponding link that complements that bit has to be traversed.

1.2. Tree-Structured Networks

The emergence of a technology for very large scale integration (VLSI), which within five years will allow the fabrication of a single silicon chip containing approximately one million active devices, also changes the constraints on a multiprocessor interconnection network. With the advances of this technology, the active devices themselves get smaller, faster and consume less power. Unfortunately, at the same time the disparity increases between the speeds of signals completely internal to the chip and signals which have to pass through the package pins. Proper systems partitioning onto several chips becomes ever more important, and high bandwidth signal paths should be kept completely internal to the silicon chips as far as possible. Thus a proper VLSI building block may be a self-contained computer on a single integrated circuit [Sequin 79].

- 3 -

Once a complete computer fits on a chip, such computers can be placed at all interconnection points of a switching network. This makes particular sense with VLSI technology since the switch by itself tends to have a low logic-to-pin ratio, making it unattractive to VLSI. The switching network then is no longer the link between a discrete bank with p processors and a separate bank with m memory modules, and we no longer need to consider only switching networks with p connection points on one side and m ports on the other. A whole set of networks such as lattices or trees can now also be considered. Since we believe that future computing systems should be easily expansible these latter structures look even more attractive. It is therefore not surprising that recently a lot of interest has been generated in tree-structured networks. [Swan 77] [Despain 78] [Mago 79] [Browning 79].

1.3. Requirements for a new structure

The disadvantage of the previously mentioned n-cube network, from this point of view, is the fact that it is not truly expansible. Whenever the number of nodes grows beyond a power of two, all nodes have to be changed since they have to be provided with an additional port. Thus the "module" of this network is not a constant, predefinable building block. Moreover, a useful expansion of this structure has to occur by doubling the number of nodes. An incompletely populated n-cube lacks some of the above mentioned properties which make the n-cube attractive in the first place. Tree structures are expansible in a natural way, and even unbalanced trees still retain most of the properties that make trees attractive.

The number of package pins does not grow at the same rate as the number of active devices within. The package periphery thus represents another physical bottleneck, and we should look at interconnection topologies which need relatively few ports per node. If the number of ports per node has to be limited,
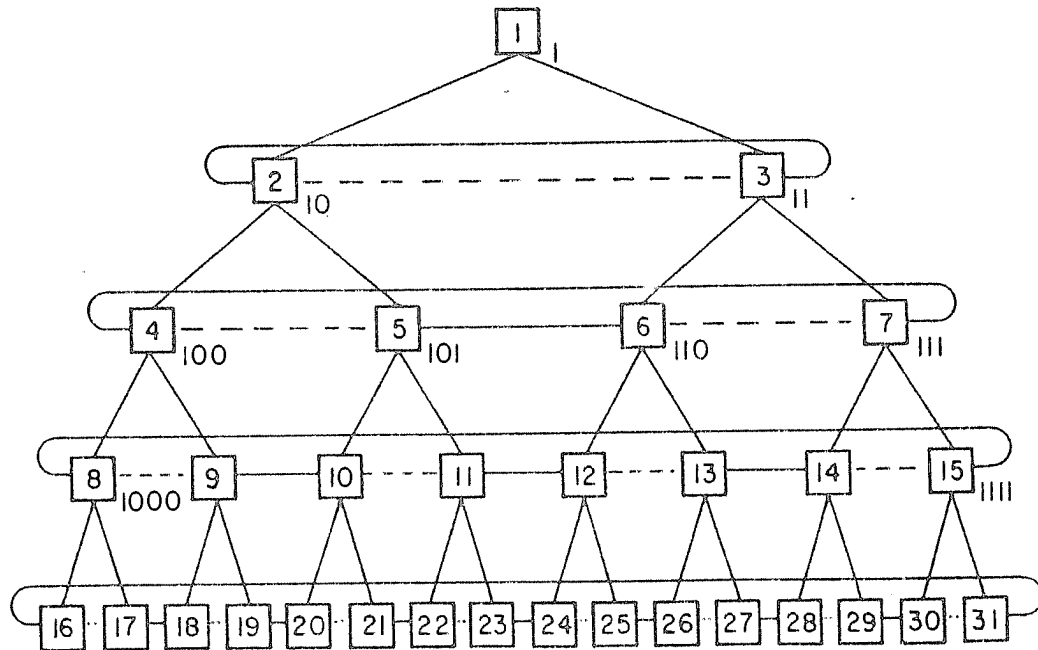
Figure 1. Binary tree with full-ring connections. When the dashed branches are omitted, a half-ring tree is obtained. The chosen numbering scheme gives all nodes on the same level addresses of the same length. The root of the tree has node address "1", and the children of node x have node adresses $2x$ and $2x+1$ respectively.

- 4 -

a binary tree structure, requiring only 3 ports per node, looks particularly attractive. Additional links, however, are required to provide redundant paths which are the basis for a system with some fault tolerance in message routing. At the same time these links can be used to reduce the average distance between nodes and to provide a more uniform message density in all links. An extensive search for the optimal placement of these additional links has shown the half-ring and full-ring binary trees (see Fig 1) to be attractive contenders, primarily because of their simple routing algorithms [Sequin 78].

For tasks in which remote leaves have to communicate extensively with one another, however, the ringed tree structures have their disadvantages. There is a lack of direct long distance connections at the lower levels of the tree. This forces messages between remote leaves to travel rather high up in the tree in order to reach their target along the shortest paths, which in turn can lead to serious congestions of some of the links below the third level of the tree. It is conjectured that a more optimal placement of the additional links could allevi-ate those problems. Hypertree is the result of the search for such a structure. It combines the best features of the binary tree and of the n-dimensional hyper-cube.

2. THE HYPERTREE STRUCTURE

The basic skeleton of Hypertree is a binary tree structure. In the following we will assume that the nodes are numbered as shown in Fig. 1. The root has node address 1. Addresses of left and right children are formed by appending a "0" and "1" respectively to the address of the parent node; i.e. the children of node x are numbered $2x$ and $2x+1$. As in the half-ring or full-ring structures, additional links in Hypertree are horizontal, and connect nodes which lie in the same level of the tree. In particular, they are chosen to be a set of n-cube con-nections, connecting nodes which differ by only one bit in their addresses. We

- 5 -

will discuss the cases of nodes with four and five ports, permitting connections of one and two additional horizontal links at each node, respectively. Thus each level of the tree can accommodate either one or two sets of n-cube connections, and the resulting structures will be called Hypertree I and II. It should be obvious from the following description how the concept can be expanded to more than two sets of n-cube connections per level.

2.1. Selection of n-cube Interconnections

With only one or two ports per node available for the n-cube connections, a choice has to be made as to which set should be chosen at each level. In a heuristic manner we start at the root of the binary tree and progress in a top down manner. We select at each level the set(s) of connections from which the most benefit can be derived: For each pair of nodes for which the addresses differ in exactly one bit, the path length, expressed in number of links, is calculated in the Hypertree structure existing above this level. Figures 2 and 3 show these distances for a binary tree. The longest connection(s) on each level are circled, and in these places a direct n-cube interconnection is introduced. Any such connection will effectively reduce the number directly below it to the value 3 , because the corresponding connections in this lower level can now be made by traveling up one level, accross the n-cube connection and down again. Correspondingly the number two levels below an n-cube connection can be no larger than 5, and so on. This has to be taken into account in determining the longest path between n-cube pairs on each level. This path is then bypassed by the addition of the corresponding set of n-cube connections. While it is clear that this selection procedure will result in a local optimization, it will be shown later, that the selected n-cube interconnections also optimize the overall tree structure in a global sense.
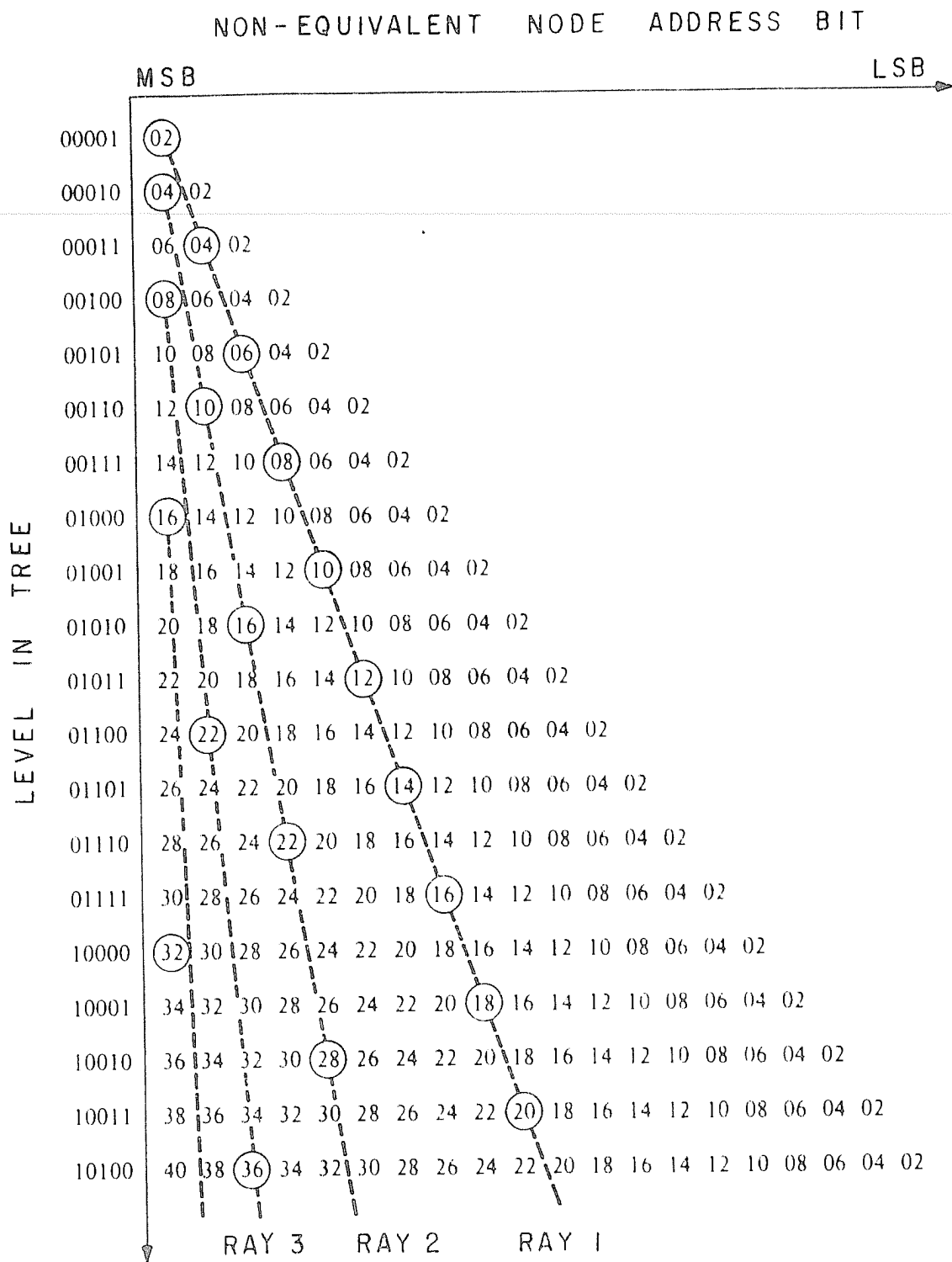
Figure 2. Selection of n-cube connections in Hypertree I. Shown are the distances between nodes on the same level which differ only by one bit in their addresses for the ordinary binary tree. Each circle represents a set of n-cube connections, chosen in such a manner that the longest distance between a pair of nodes with Hamming distance 1 at that level gets reduced to one.

NON−EQUIVALENT   NODE   ADDRESS   BIT

MSB   LSB

LEVEL   IN   TREE

	MSB															LSB
00001	(02)															
00010	(04)	(02)														
00011	(06)	(04)	02													
00100	08	(06)	(04)	02												
00101	(10)	08	06	(04)	02											
00110	12	(10)	(08)	06	04	02										
00111	14	12	10	(08)	(06)	04	02									
01000	(16)	14	12	10	08	(06)	04	02								
01001	18	(16)	(14)	12	10	08	06	04	02							
01010	20	18	16	(14)	12	10	(08)	06	04	02						
01011	22	20	18	16	(14)	12	10	(08)	06	04	02					
01100	(24)	22	20	18	16	(14)	12	10	08	06	04	02				
01101	26	(24)	22	20	18	16	14	12	(10)	08	06	04	02			
01110	28	26	(24)	22	20	18	16	14	12	(10)	08	06	04	02		
01111	30	28	26	(24)	22	20	(18)	16	14	12	10	08	06	04	02	
10000	32	30	28	26	24	22	20	(18)	16	14	(12)	10	08	06	04	02
10001	34	32	30	28	(26)	24	22	20	18	16	14	(12)	10	08	06	04	02
10010	(36)	34	32	30	28	(26)	24	22	20	18	16	14	12	10	08	06	04	02
10011	38	(36)	34	32	30	28	26	24	22	20	18	16	(14)	12	10	08	06	04	02
10100	40	38	36	34	32	30	28	26	(24)	22	20	18	16	(14)	12	10	08	06	04	02

Figure 3. Selection of n-cube connections in Hypertree II. Each circle represents a set of n-cube connections as in Fig. 2.

- 6 -

In Hypertree II there are occasionally levels where a choice exists for the placement of the horizontal link, when two longest paths are of equal length. However, even from the links which are placed with no free choice, a regular pattern of these interconnects appears, and the choice is always made in order to follow this pattern. For Hypertree I all n-cube connections fall on slanted lines or "rays", starting at the location of the most significant bit on levels which are a power of two (Fig. 2). The slope of subsequent rays doubles and is equal to twice the level number on which it starts. For Hypertree II the pattern is more complicated. All n-cube connections appear in diagonal pairs, which themselves fall again on slanted lines (see Fig 3). The resulting interconnection pattern for a small Hypertree I is shown in Fig. 4.

2.2. Formal Description of Hypertree I

If we number the levels downward, starting with the root as 0, we notice that the level number, n, for each node is the number of binary digits after the most significant one in its address. An inspection of Fig. 2 shows that the number of consecutive trailing zeros, z, in the level number, expressed in binary, defines the ray number,

$$r = z + 1, \tag{1}$$

on which the set of n-cube connections on this particular level lies, e.g. on level 10100 the n-cube connections belong to ray 3. The specific bit b which is affected by the n-cube connections at level, m, can then be determined by the intersection of the proper ray with this level. For ray r,

$$b = \frac{m}{2^r} + \frac{1}{2} \tag{2}$$

where b is the bit number, counting from the left with the redundant, most significant "1" in the node address counted as bit 0.

Several observations can be made which follow an inspection of Fig. 2. "Ray
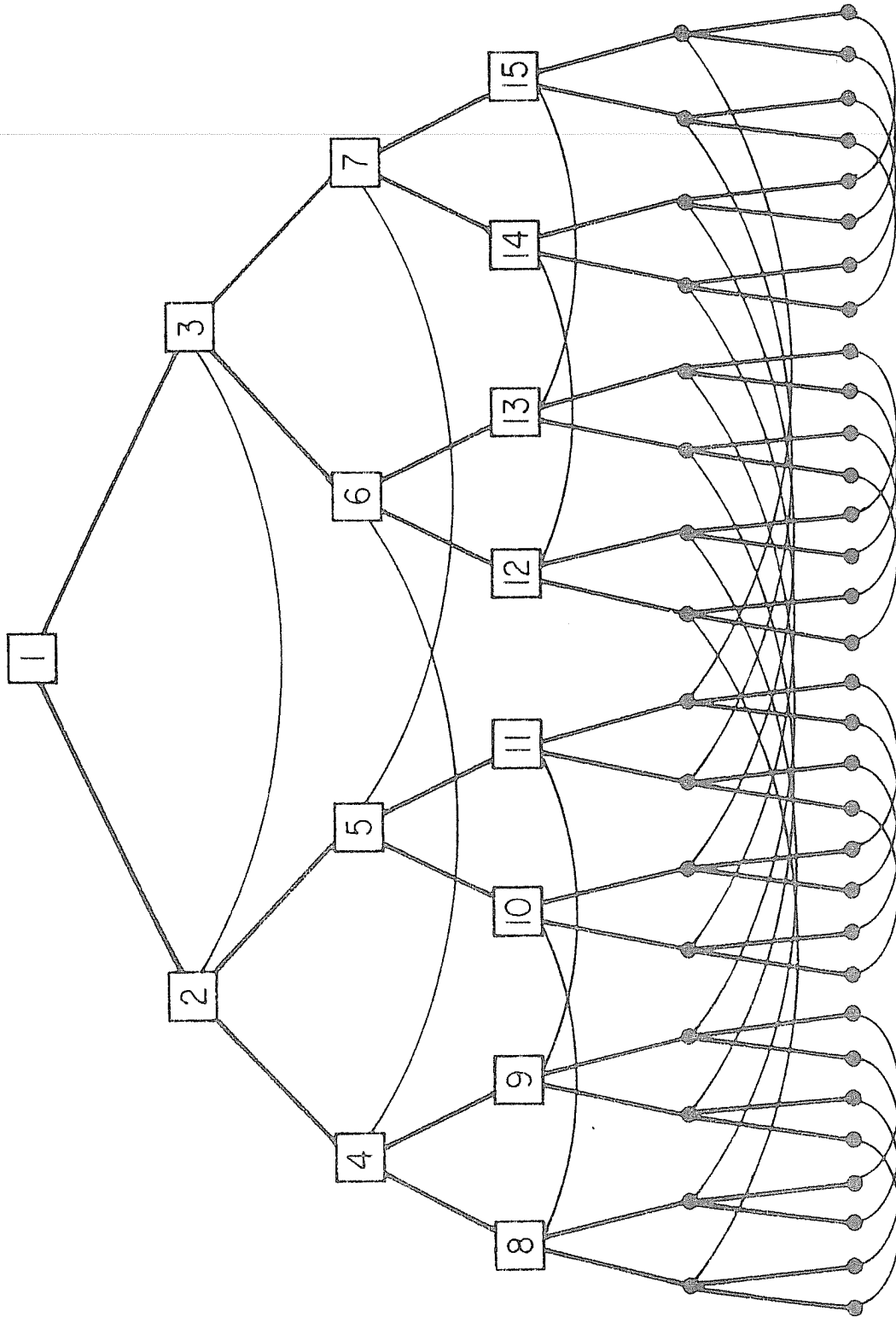
Figure 4. Interconnections in Hypertree 1.

level	Hypertree I		Hypertree II	
	above	whole	above	whole
1	1	1	1	1
10	2	2	1	1
11	3	3	2	2
100	4	3	3	3
101	5	4	3	3
110	6	5	4	3
111	7	5	5	3
1000	8	6	5	4
1001	9	7	6	5
1010	10	7	7	5
1011	11	8	7	5
1100	12	9	8	5
1101	13	9	9	6
1110	14	10	9	7
1111	15	11	10	7
10000	16	11	11	7
10001	17	12	11	7
10010	18	13	12	8
10011	19	13	13	9
10100	20	14	13	9

Table I. Worst case distances between nodes with Hamming distance 1 for Hypertree I and II. Two cases are considered: "above", where messages only travel above the two nodes, as is appropriate for leaf-to-leaf traffic, and "whole", where messages can find the best path above or below, assuming an infinitely large tree.

- 7 -

1" goes through the middle bit of the significant part of the node address on every level with an odd number of significant bits (excluding the redundant leading one). This means that the n-cube connections always complement a bit in the more significant half of the node address, while the structure relies upon the skeleton of the binary tree for changes in the less significant half. In addition it turns out that for a tree of any size, exactly one representative of each such an n-cube connection is found in the lower half of the tree. Since the lower half of the tree also contains all the necessary connections to make any change in the least significant half of the address, it follows that messages originated at a leaf node need never travel higher than half the height of the tree in order to reach any other leaf.

3. PROPERTIES OF HYPERTREE

In this section some of the relevant properties of Hypertree as an interconnection network will be discussed and compared to other previously described multiprocessor topologies.

3.1. Worst Case Distances

One important measure of the power of an interconnection network is the distance messages must travel in the network. It is advantageous to make this parameter as low as possible, since it will not only reduce travelling time for messages, but also minimize message density in the links. Absolute worst case distances between any two nodes in Hypertree I can be based on the observation made in section 2.2: it is equal to m, the number of levels above the lower of the two nodes. Worst case distances between n-cube pairs can be determined by inspection of Fig. 2 and 3 and are summarized in Table I. For Hypertree I and II worst case distances are listed for a) the case where only connections higher in the tree can be used, as is the case for leaf-to-leaf communication, and b) the

- 8 -

case where we assume to be in the middle of a large tree, so that messages can find suitable n-cube connections above as well as below the current level. For Hypertree I the longest distance is 1/2 and 1/3 of the distance in a simple binary tree, and for Hypertree II they are reduced to 1/3 and 1/5, respectively.

3.2. Average Distances

Of even more importance may be the *average* path length travelled by all messages. In order to obtain a meaningful comparison between different networks, some normalization has to be made, in particular if networks between processors with different numbers of ports per node are considered. With no limits on the number of ports, a fully interconnected network could be designed, which would lead to an average distance of one. In order to take into account realistically the limitations in the number of pins and in the amount of power available to drive communication lines, Despain [Despain 79] has proposed that, in the context of single-chip computers, a constant communications bandwidth per node should be assumed. Under this assumption, the bandwidth available through each port is then B/p, where p is the numer of ports and B is the total bandwidth available from that processor. Using the same idea, we propose that the average message path length, L, be normalized by multiplying it with the number of ports, p, in order to permit a meaningful comparison of graphs of different degrees. Thus the normalized average message path length, L′, then becomes

$$L' \equiv L \times p. \tag{3}$$

Another factor influencing the average message path length is the distribution of pairs of communicating nodes. In the absence of any specific information about the communication patterns required by particular task, one might assume a uniform distribution in which all nodes send messages with equal probability to all other nodes. More appropriately, for tree-structured networks

Structure	L	L′
Binary Tree	$2n - 2 + \dfrac{2}{N}$	$6n - 6 + \dfrac{6}{N}$
Half-Ring$^\circ$	$2n - \dfrac{139}{32} + \dfrac{31}{4N}$	$8n - \dfrac{139}{8} + \dfrac{31}{N}$
Full Ring†	$2n - 5 + \dfrac{8}{N}$	$10n - 25 + \dfrac{40}{N}$
n-Cube	$\dfrac{n}{2}$	$\dfrac{n^2}{2}$
Hypertree I†	$\dfrac{5n}{4} - \dfrac{4}{3} + \dfrac{4}{3N} - \dfrac{n \bmod 2}{12}$	$5n - \dfrac{16}{3} + \dfrac{16}{3N} - \dfrac{n \bmod 2}{3}$

Table II. Average distance between nodes for various networks as a function of the number of leaf nodes $N \equiv 2^n$. L is the average distance between every pair of nodes (including each node with itself). L' is the normalized distance obtained by multiplying the average distance by the maximum number of ports per node. traffic.

$^\circ$ for n > 2.

† for n > 1

- 9 -

where all I/O points and all secondary memory are connected to the leaves of the tree [Despain 78], a set of messages running between all pairs of *leaf* nodes will be studied.

3.2.1. Leaf-to-leaf traffic

Table II shows the average distances between leaves for the binary tree, the half-ring tree, the full-ring tree, Hypertree I, and the average distance between all nodes for the n-cube. The n-cube is assumed to have $N \equiv 2^n$ processors, and all tree structures have $N \equiv 2^n$ leaves and thus a total of $2N - 1$ processors. The derivation of the average path length is trivial for the binary tree and the n-cube, but for Hypertree it is rather tedious and thus deferred to appendix A. Average distances for the half-ring and full-ring trees have been determined through an exhaustive count on the computer. Table II includes the normalized average distances, L', for the same structures, which are also plotted in Fig. 5 as a function of n. Note that the normalized average path length in Hypertree is always shorter than in the binary tree, and eventually becomes even less than that of the n-cube.

3.2.2. N-cube nearest neighbors

The n-cube interconnection, i. e. the paths between nodes that have a Hamming distance of 1, can play an important role in problems such as Fast Fourier Transforms and sorting. These connections are the basis for many interconnection networks in SIMD architectures [Batcher 73] [Siegel 76]. It is therefore worthwhile to study these special interconnection paths in the above structures. For balanced binary trees we use the node numbering scheme of Fig. 1, so that the leaf nodes are numbered from 2^n to $2^{n+1} - 1$. We then define n-cube nearest neighbors to be those pairs of leaf nodes which have a Hamming distance of one, and assume uniform traffic among them. The average distance between those
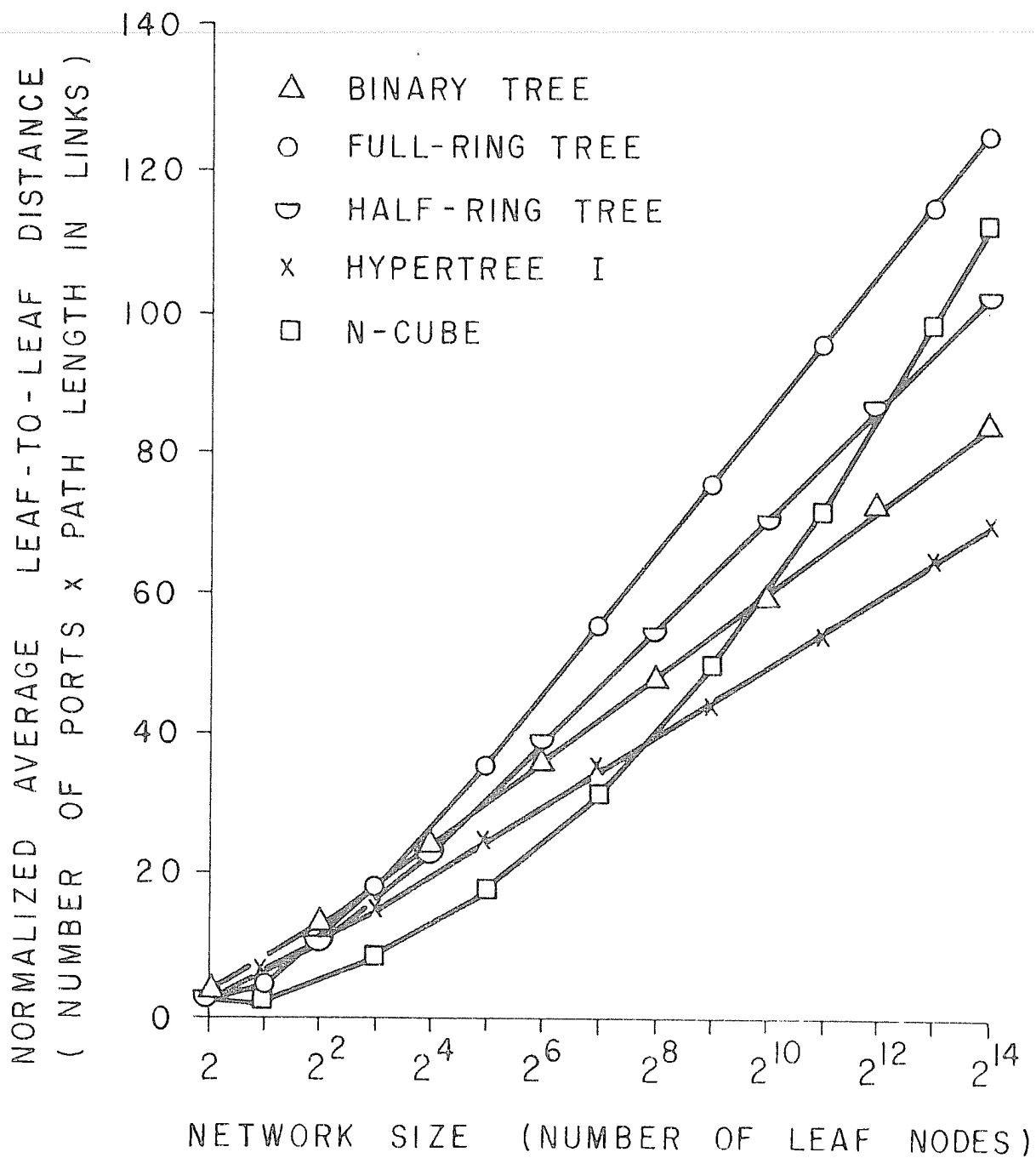
Figure 5. Average path length in various structures as a function of the number of processors. It is assumed that every leaf sends with equal likelihood to every leaf.

- 10 -

n-cube nearest neighbor nodes for the binary tree turns out to be n. For Hyper-tree I, this value is n/2, while for the n-cube, of course, it is 1. Normalizing these numbers as before with the number of ports per node gives:

binary tree: 3n

Hypertree I: 2n

n-cube: n

Thus the Hypertree I structure is only a factor of two worse than the n-cube, which, in this particular case, is the ideal structure. For the binary tree the above numbers are valid also for n-cube nearest neighbors within the tree, while for Hypertree the number given is only an upper bound; paths through the lower parts of the tree may provide a more direct connection.

3.2.3. Consideration of locality in traffic

It seems reasonable to assume that an efficient and practical multiproces-sor system will exhibit much greater traffic over short distances than over long communication paths, since interaction among small clusters of processors may cause a large portion of the total traffic. This may result from the fact that tasks that can be broken up into smaller subtasks would normally be assigned to neighboring processors. Furthermore, the way that multiple processors are interconnected often reflects the need of the communication patterns in the first place and takes into consideration the fact that communication costs increase with distance. Under such conditions, the effective average message path length may be much shorter than the numbers given in Table II.

Without such locality in the message traffic, the amount of message com-munication handled per processor will drop off in most multiprocessor networks

as the number of nodes increases. This becomes clear from the following analysis: We assume a system with N nodes with a uniform message density close to the limit defined by the bandwidth of the links so that throughput is limited by communication bandwidth. Given the constant bandwidth assumption, if the number of nodes in such a system doubles, and we assume that all nodes communicate equally with all others, then the number of messages sent among the processors quadruples, while the number of links in the system has only doubled. Since the number of nodes that a processor can address has also doubled, the message rate between any specific pair of processors can be at best half the rate in the original system. This rate however can only be achieved if the average path length remains the same. Any increase in the path length will result in a further reduction in the message rate, so that each processor is actually sending and receiving fewer messages. With sufficient locality the longer paths obtain a much lower weight, and thus the increase in average distance can become insignificant.

We have been unable to derive a locality function for any of the enhanced tree structures for which a closed form expression of the average path length could be determined. To get some qualitative understanding of how locality affects the increase of path length with increasing tree size, we will take a look at some crude and extremely simplistic models.

As one extreme, we can assume that communication between pairs of nearest neighbors is most important. In order to provide highest bandwidth for this case, within the constraints of constant total bandwidth per node, the degree of the graph of the interconnection network should be chosen as low as possible. In the extreme this would lead to a set of isolated pairs of nodes, and the average path length then remains constant at the value 1.

In order to get some approximation for a more practical situation, we will

- 12 -

assume an arbitrary network in which each node sends one half of its maximum possible message traffic to nodes which are one link away. Half that many messages are sent to nodes at a distance of two links, and in general an amount of traffic $T(d)=T_{max}2^{-d}$ is going to nodes at distance d away from the sender. For a graph of infinite extension, the average message path length then becomes 2, while for graphs with a finite diameter k, the average path length will be $2-2^{-k}$. Thus, with the above assumtions on locality in the message traffic, the average path length is almost independent on the size of the network. Thus a primary goal must be to try to map a problem onto the topology of the network, so that the direct connections are used as often as possible, or alternatively, to provide a network with a structure of the local interconnections for which this is easily possible for many important problems.

3.3. Routing Algorithms

One of the desirable requirements for a large network of processors is that messages can be routed by each intermediate processor without total knowledge of all the details of the network, since the storage of that information within each node can use up an exorbitant amount of memory space. Both, the n-cube and the binary tree have simple routing algorithms which involve only the addresses of the current location of the message and of the target processor. In the n-cube, links are selected which reduce the Hamming distance by one, until the target is reached. In the binary tree a message is routed upwards in the tree (towards the root) until the target node is a descendent of the current node; from there it is routed down. Both the half-ring and full-ring tree structures also have simple routing algorithms which are optimal in the sense that they always find the shortest path [Sequin 78]. Both follow basically the binary tree algorithm, but use the horizontal links whenever the path length can be reduced.

- 13 -

3.3.1. A simple routing algorithm

Similar algorithms exist for the Hypertree structures. They are optimal for messages between leaf nodes in a balanced tree. A simple routing algorithm works as follows:

As in the basic binary tree, messages are routed upwards in the tree until the target is a direct descendent of the current position. In addition, whenever the n- cube connections at a particular level reduce the Hamming distance between the current position and the target, the corresponding link is traversed.

This algorithm will result in a path in which all useful n-cube connections have been traversed before the message has reached the highest point of its path, even though any particular n-cube connection could also have been utilized during the downward phase of the message routing.

3.3.2. Optimal routing algorithm

There are circumstances under which the simple routing algorithm will not lead to the shortest path.

1) For messages between non-leaf nodes, the use of n-cube connections below either source or target node may lead to a shorter path. The simple algorithm will not find these paths since it makes no assumptions about n-cube links below the current node.

2) A similar observation can be made even for traffic between leaf nodes, if the tree is unbalanced. If the target lies below the source, an n-cube link below the target could be used for the shortest possible connection. The simple algorithm may search much higher in the tree for a corresponding interconnection.

3) If the the Hypertree structure is incomplete, and a particular n-cube link is missing, the corresponding bit could be complemented by taking another n-cube link of the same set on the way down. The above algorithm in its simplest form

is not aware of this possibility.

Under the assumptions of a balanced, complete Hypertree, we were able to derive a routing algorithm which always selects a path of minimum distance. It is too complex, to be described here, but the point should be made that an optimum algorithm exists which uses only local information and knowledge of the size of the tree.

An algorithm able to find the optimal path in an unbalanced tree must have knowledge about the extension of the frontier, i.e. the position of the leaves, of the tree. For the same reason, in order to find the shortest path, it would also need global knowledge of any missing links, be it because the tree is incomplete by design or because certain links have failed arbitrarily. Simulation studies have demonstrated that in balanced trees with up to eleven levels (4095 nodes), the simple routing algorithm yields an average path length for all pairs of nodes which is never more than 0.42% greater than the optimal path length. The slight potential improvement must be weighed against a much more complicated algo-rithm, requiring more detailed global information about the network, and a more complicated routing controller.

3.4. Message Density

Another major goal in the design of an efficient network topology is to dis-tribute traffic as evenly as possible over all existing links. The basic binary tree and the fullring tree both have serious deficiencies in that respect. The binary tree suffers intolerable congestion near the root, since roughly half of all traffic must pass through the highest links in the tree if no locality exists. A similar bottleneck exists in the horizontal links on the third level of the full-ring tree if the simplest routing algorithm is used. Some of the same kind of bottlenecks exist even in the Hypertree, but to a much lesser degree, since, as has been shown in Section 2.2, leaf-to-leaf messages never go more than half way up the

- 15 -

tree. The fact that messages are kept in the lower, wider parts of the tree, reduces congestion considerably.

Fig. 6 shows the maximum number of messages routed through the busiest link for the n-cube, the binary tree, half-ring and full-ring tree, and Hypertree as a function of tree size. The data has been normalized by multiplying the derived values with the number of ports per node. For the binary tree, the busiest links are the two top ones, while for the n-cube, all links carry an equal amount of traffic. For Hypertree I with 2^n leaf nodes the horizontal links $\frac{n+1}{2}$ levels below the root are the busiest, if n is odd. If n is even, the load is equally heavy on horzontal links at level $\frac{n}{2}+1$ and on the vertical links immediately above that level. The full-ring tree has a seroius bottleneck at the horizontal links on level 3, and the half-ring has almost as much congestion on the vertical links above level 3. Of all the described tree structures, the Hypertree stucture is clearly the best in that respect. While in the other tree structures the maximum traffic density grows roughly as the square of the number of nodes in the tree, in Hypertree I the growth rate is only $N^{1.5}$.

It should be pointed out, that even for n-cube, communication density grows beyond any fixed bound, making large systems inefficient without the reliance on locality in the message traffic. Similar trends as demonstrated for average path length exist, making the maximum message density rather independent of the size of the network with certain models of locality. Since it is not obvious what a proper model for locality is unless the mapping of a particular application onto the network has been worked out in detail, specific calculations of maximum traffic densities other than with uniform message traffic have not been pursued at this point.
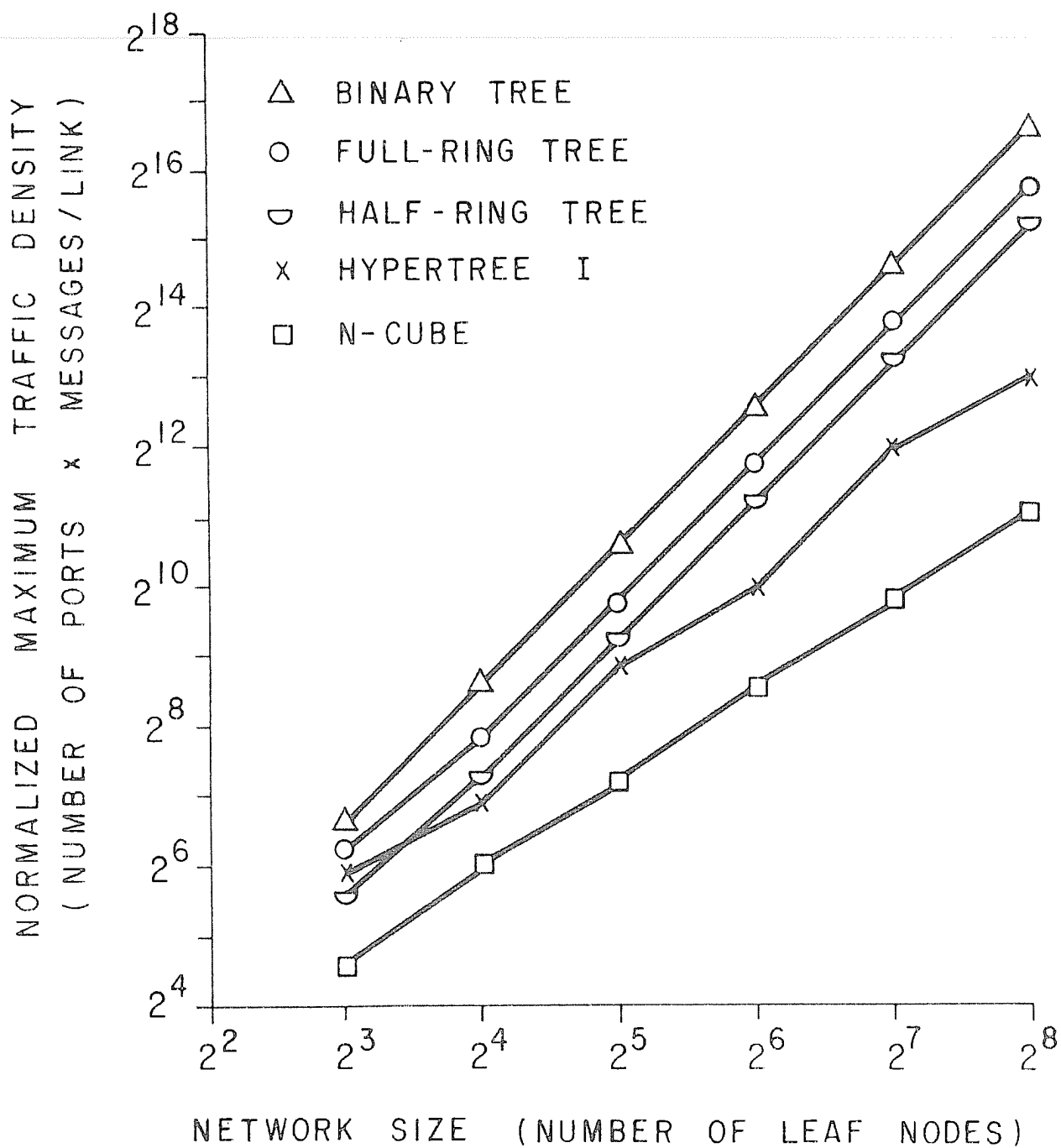
Figure 6. Maximum number of messages routed through a single link as a function of network size for uniform leaf-to-leaf traffic. Numbers have been normalized by multiplying them with the number of ports per node required.

- 16 -

4. PRACTICAL CONSIDERATIONS

4.1. Expansibility

Among the important parameter of a multiprocessor system are its modularity, expansibility, and specifically the smallest increment by which the system can be expanded in a useful way. It is generally unreasonable to demand that a system must remain balanced in all stages of expansion, since this may imply that its size must be increased in large steps, i.e. powers of two. The shortcomings of n-cube with respect to modularity and incremental expansibility have been discussed in Section 1.3.

Binary trees also require a doubling of size in order to remain balanced. However, tree structures do not lose too much of their attractiveness if they are not perfectly balanced. The routing algorithms discussed still reach their target in an efficient way even if the tree is somewhat unbalanced, as long as the skeleton of the binary tree is still present. Although routing in such a system may not be optimal for all circumstances, such a structure still has its usefulness if it reflects the nature of the problems that it is handling, and provides the links as required by the nature of the communication patterns in that context.

Two basic approaches could readily be imagined by which Hypertree is expanded in a "natural" way. If the particular installation is a single, cohesive system, additional nodes should be placed on the last incomplete level, before a new level is started. The imbalance will then never exceed one level, an amount which causes no problems for the routing algorithm. On the other hand, if the system needs to grow organically with the needs of a widespread and diverse user community, many individual subtrees may develop which are connected through a shared main tree containing the root of the system. In both cases, n-cube interconnects could be added as soon as both the nodes, differing only in the corresponding bit that is to be complemented at that particular level, are

- 17 -

present. However, the sequence of n-cube interconnects on subsequent levels, as derived in section 2.1, is really designed for a single cohesive system. If the system contains many disjoint subtrees, connected in only a few points to a main tree, it may be better to start a new sequence of n-cube interconnects within each subtree, thereby optimizing local communications. This would be worth the price of storing the list of these sets as a function of the level in each routing controller. Since the lack of interconnection between individual subtrees is a consequence of the way the system has been expanded, it can be assumed that it reflects an absence of the need to communicate between nodes belonging to different subtrees. Thus the lack of those horizontal links should have no adverse effect on the overall performance of the system or on the effective average pathlength.

4.2. Fault Tolerance

The requirements for fault tolerance have greatly increased in recent years as systems have become increasingly complex. Certainly an important feature of structures such as those considered here is that it must continue to work correctly, though perhaps with reduced performance, when one or more components have failed. Specifically we expect such a system to continue to operate properly in the presence of failures of a single link or even a single node with all its attached links, as long as that particular node is not involved in the computation, i.e. is neither the source nor the destination for any messages.

The addition of the n-cube links to the binary tree creates multiple disjoint paths between every pair of nodes. The simple algorithm described in the previous section already has some fault tolerance. It can readily cope with missing links during its upward move through the tree, by using the following alternate choices for missing links:

DESIRED
PATH

ALTERNATE
PATH

MISSING
LINK

(a)

DESIRED
PATH

ALTERNATE
PATH
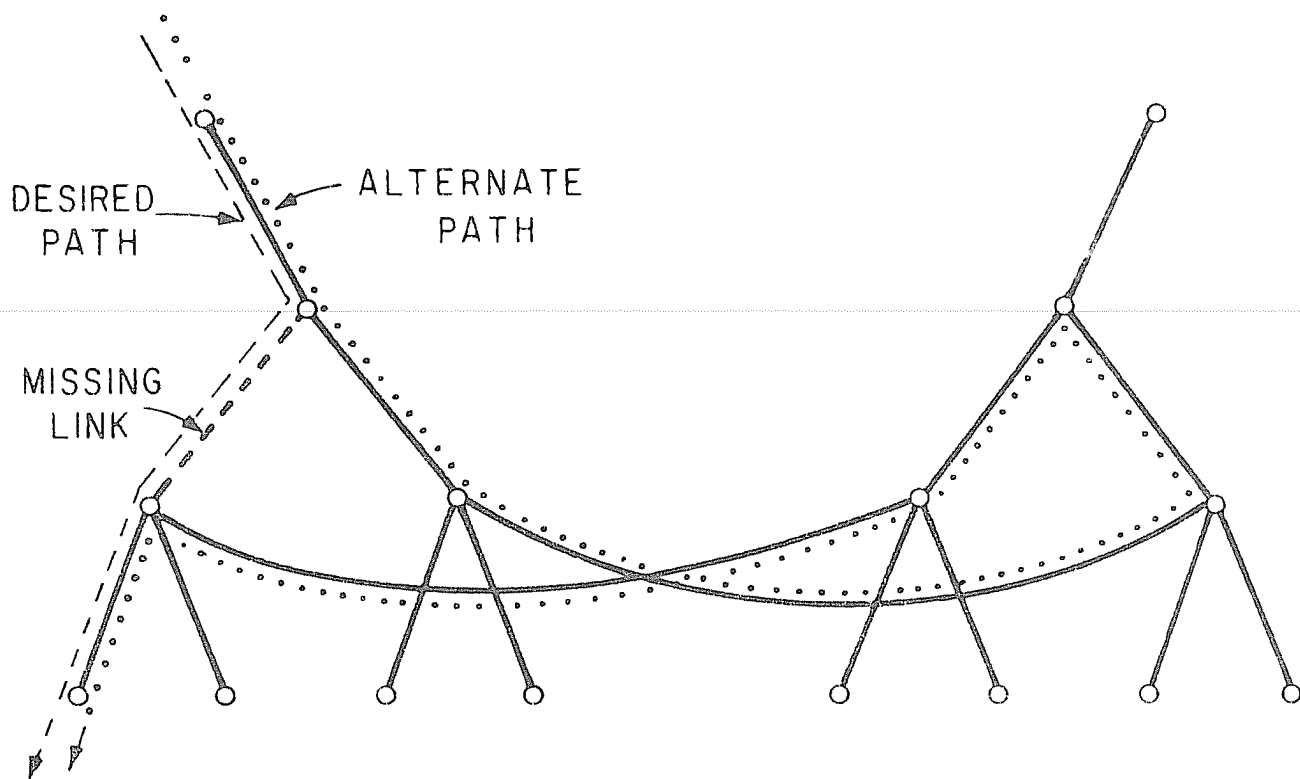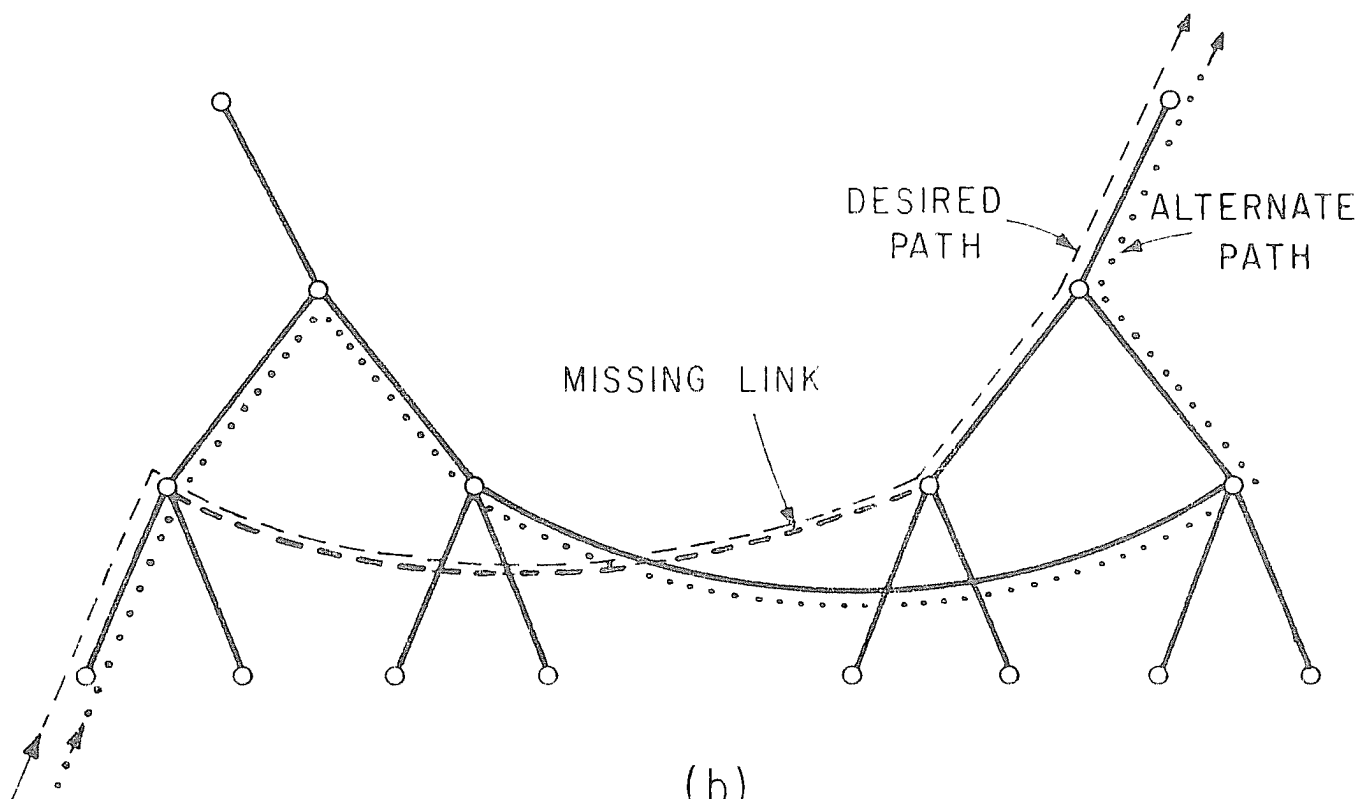
MISSING LINK

(b)

Figure 7. Detours around missing or defective links, a) around a
vertical link in the downward direction, b) around missing n-cube
link

- 18 -

a)   If an n-cube link cannot be passed, go upwards instead.

b)   If an upward link cannot be passed, go across n-cube link instead, followed by an upward move.

Difficulties occur on the downward branch if a missing link is encountered. Because of the lack of redundant paths over short distances, a deviation from the proper descending path can not be corrected by traversing links of the binary tree skeleton only. A detour into another plane of the hypercube has to be made (Fig. 7a). Instead of the missing vertical link, the downward link to the sibling node is taken. After traversing the n-cube link at that level, the message is routed to the sibling node in that part of the tree, and subsequently back to the original path across another n-cube link at the same level. During this whole detour the message has to remember that it is being rerouted and can not simply follow the standard routing algorithm. This extra information has to be carried along in the message header. The decoding and proper treatment of this information will increase the complexity of the routing algorithm.

Additional features can be built into the simple routing algorithm in order to enhance its efficiency in the case of single element failures in balanced hypertrees. The unsuccessful attempt to use an n-cube interconnect during the upward branch could be remembered in the message header, and the corresponding bit could be complemented in the downward branch of the message path. Alternatively a local detour similar to the one described above could be built into the path (Fig. 7b). Through the common parent node the message is shipped to the brother of the node with the unavailable n-cube link, from where the the n-cube traversal is now executed. At the other end, the desired path can be reached again in one or two steps by going through the parent node. The trade-offs between the efficiency of these fault tolerant routing algorithms and the complexity of the necesseray hardware to implement them will have to

- 19 -

be evaluated individually for each realization of such a system.

In summary, this algorithm will successfully route messages in the presence of no more than one failed link or node, provided that the failed node is not itself the source or target. This is true even in somewhat unbalanced trees, provided that the expansion has been performed in a "natural" manner. This implies that a complete binary tree skeleton exists above the two nodes, that each node has a brother node, and that brother nodes are connected to another pair of sibling nodes through two n-cube links. Thus, at least two disjoint paths should have existed before the failure.

While the addition of the long-distance n-cube interconnections is useful for certain algorithms, the lack of redundant paths to nearby nodes, such as the extra links in a full-ring or half-ring tree, may be disadvantageous if there is a lot of locality in the message traffic and also for generating a simple fault-tolerant routing algorithm. If five ports per node can be afforded, it may therefore be advisable to use only one set of hypercube connections per level and reserve the other extra port for a half-ring connection.

5. RELATION TO OTHER NETWORKS

Other people have studied various structures in the context of the questions addressed in this paper. Schlumberger[74] analyses the de Bruijn network, which may be introduced as the state diagram for an n-bit, k-ary shift register, where k is the number of unidirectional links leading in and out of each node. He has shown that the worst case path length for $N=2^n$ nodes is n and that the average path length, when all nodes communicate equally with one another, is slightly less than that. The structure has an elegant routing scheme, requiring only local information, and also has reasonable fault tolerance. By considering the special case of a binary shift register ($k=2$) and making all links bidirectional paths, we obtain graphs for processor with four ports which have many

- 20 -

similarities to the networks discussed here. The de Bruijn network can also be drawn as a binary tree with one additional node and with feedback paths from leave nodes to the higher parts of tree (Fig. 8).

The average distance in this network is about 15 to 25% shorter (for trees with up to eleven levels) than in Hypertree I. The reason for this is that this network uses all of its links for communication and none for input, output or connection to secondary memory. In the tree structures discussed, all leaf nodes have two free ports, while the de Bruijn network has no ports available. For a fair comparison, a fifth port should be added to each node in the de Bruijn network, and this network should then be compared to Hypertree II, not Hypertree I. It turns out that the average path length of Hypertree II is indeed slightly shorter than that of the de Bruijn graph. Alternatively, if we use the two ports at the leaves of Hypertree I for additional connections, and place a perfect shuffle network [Stone 71] at the bottom of the tree, the average path length drops to 5 to 10% below that of the de Bruijn graph.

However, the additional paths in the de Bruijn network break the nice symmetric properties which are inherent in the other tree structures. It is not apparent that algorithms can be derived to take advantage of this unusual network, and it therefore appears to show less promise than Hypertree.

Pease[Pease 77] has proposed an "Indirect Binary n-Cube" array, a structure which has been suggested numerous times in various contexts [Beizer 62] [Benes 65] [Lawrie 75] [Goke 73], for use with microprocessors. In this structure, each stage implements one dimension of the n-cube interconnection, i. e., each level provides the exchange corresponding to one bit in the node address. If each level of edges in the simple binary tree is thought of as one "stage" of a switching network, it will be seen that the binary tree has the same characteristic. The bottom level provides the exchange corresponding to the least
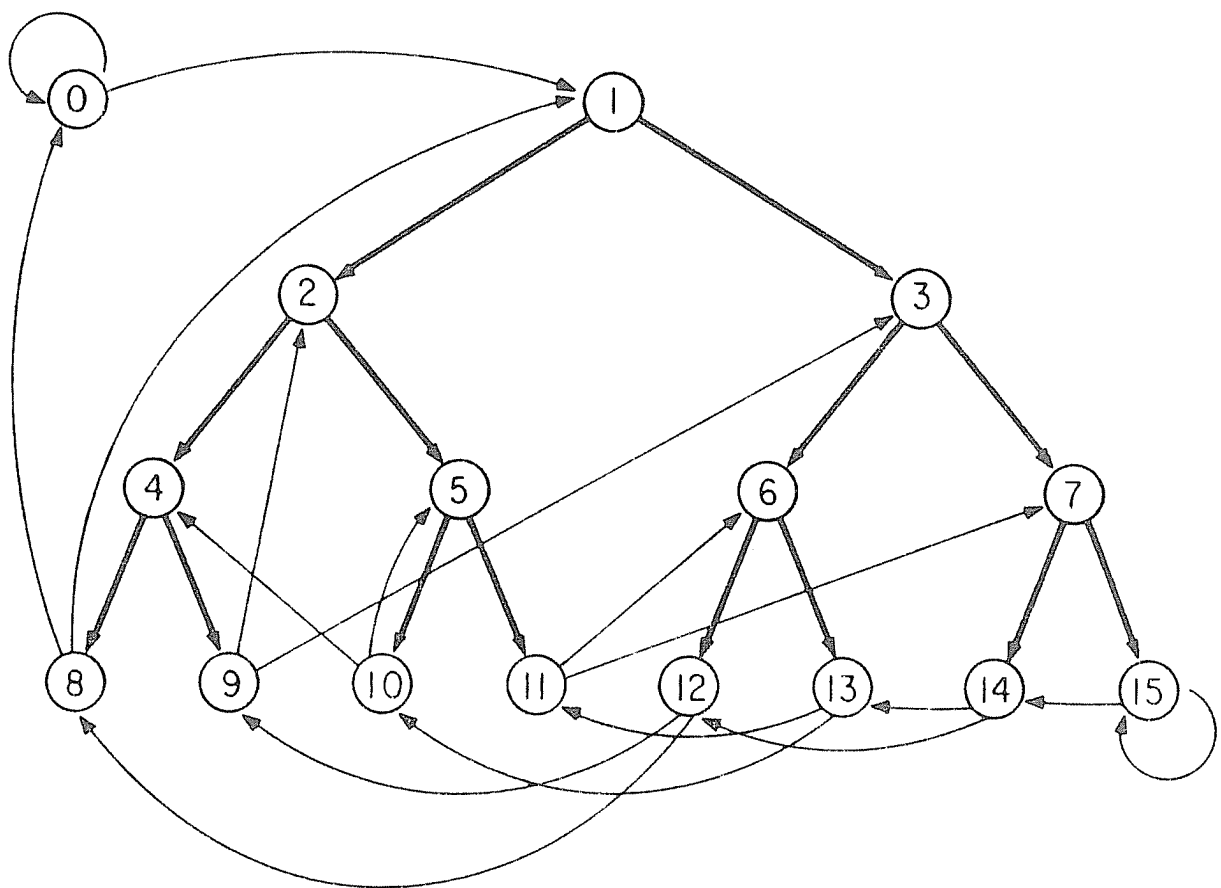
Figure 8. De Bruijn network represented as a binary tree with an extra node and additional feedback links.

- 21 -

significant bit, the next level the next least significant bit, etc. The horizontal connections in Hypertree at each level also provide the exchange corresponding to one bit in the node address. Although Pease did not propose it this way, it would seem possible to implement a network of single-chip computers with switching circuitry in the form of an indirect n-cube. The two ends of the network could be tied together, forming a cylinder, and the links could all be bidirectional. Such a network has some very interesting properties and is being analyzed for its possibilities.

Both of the above mentioned structures contain one property that made them unacceptable in our application: lack of incremental expansibility. A minimum increment in the case of the de Bruijn network requires a doubling of the number of processors, and even worse, a total reconfiguration of the nodes. A minimum increase in the size of the indirect n-cube requires doubling the circumference of the cylinder and increasing its height by one extra stage. Neither structure appears to maintain its nice properties if it is not complete.

6. CONCLUSIONS

A new network topology for multiprocessor systems has been derived in an attempt to combine the best features of easily expansible tree structures and the rather compact n-dimensional hypercube. The two underlying structures permit two distinct logical views of the system. Thus problems which map particularly nicely onto a tree structure can take advantage of the binary tree, while those that can use the symmetry of the n-cube can be assigned to processors in a way that efficiently uses the n-cube links.

The regular structure allows the implementation of simple routing algorithms, which require no detailed knowledge of the network interconnections. With a relatively small additional overhead, a routing algorithm can be con-

- 22 -

structed that is robust enough, so that messages will arrive at the proper node even for grossly unbalanced trees or in the presence of failing nodes or links. This is a requirement for easy expansibility of the system and for graceful degradation in the presence of communication hardware failures.

The network is readily expansible in an incremental way. All nodes have a fixed number of ports regardless of the size of the network. These two properties make this topology particularly attractive for implementations of multi-microprocessor networks of the future, where a complete computer with a substantial amount of memory can fit on a single VLSI chip.

ACKNOWLEDGMENTS

APPENDIX 1: AVERAGE PATH LENGTH IN HYPERTREE

In this section we will derive the expected path length in Hypertree I for a message between arbitrary leaf nodes. This is equivalent to deriving the average path length when all leaf nodes send messages to all leaf nodes. We shall include the useless case of a node sending a message to itself since it results in simpler formulas if it is left in. If desirable it can easily be removed.

For a graph consisting of a balanced, binary tree plus the additional links defined in this paper as Hypertree I, we number the nodes in the natural way: top to bottom, left to right, starting with the root as "node 1" on "level 0". These node addresses, expressed as binary numbers with coefficients x_j, will then take the form

$$X = x_0\, x_1\, x_2 \cdots x_m$$

where m is the level number of the node, and where $x_0 = 1$ and $x_j = 0$ or 1, for $j = 1,2,3, \cdots m$. In particular, for leaf nodes $m = n$, where n is the number of levels below the root.

We now define a "half-way" level $l = \left\lceil \dfrac{m}{2} \right\rceil$ and $a = m \bmod 2$, so that $l = \dfrac{m+a}{2}$. One can convince one-self, e.g. by looking at Fig. 2 in the paper, that any bit in a leaf node address can be complemented by following a path which stays in the lower half of the tree. Some bits get complemented by using the skeleton of the binary tree only, while others make use of one of the horizontal n-cube links. Now consider the path between two leaf nodes X_1 and X_2, and define the *address difference* as

$$Y_{1,2} = 0\, y_1\, y_2\, y_3 \cdots y_n.$$

where y_j, is the modulo-2 sum between x_{j1} and x_{j2}.

We now rename the bits in this address difference so that b_j are bits which can be complemented by staying on the binary tree links in the lower half of the

- 2 -

tree, and h_j are bits for which the shortest path leads across a n-cube link:

$$Y_{1,2} = 0 \left\{ h_0 \, h_1 \cdots h_{l-1} \right\} b_{l-a} \, b_{l-a-1} \cdots b_2 \, b_1.$$

The indices of the bits denote how many levels above the leaf the path has to climb in order to make the complementation of that bit possible. Thus if the bit position to be complemented is among bits b_j, i.e. $b_j = 1$, the the shortest path between the two nodes uses only links of the simple binary tree, going up j levels, then back down, and the minimum distance d is $2j$. If the differing bit is among bits h_j, then the shortest path between the two nodes goes up the binary tree j levels, across the horizontal link, then down through the binary tree, and $d = 2j+1$. While the indices of the bits b_j appear in decending order in $Y_{1,2}$, the bits h_j are an unordered set. It is important to note, however, that every term from h_0 to h_{l-1} appears exactly once for a tree with $n = 2l-a$ levels below the root.

In the general case where the source and the target addresses differ in several bits, the minimum path length d is achieved by climbing the tree to the highest level necessary, then going back down, traversing the horizontal links for the necessary exchange of bits h_j on the appropriate levels, either on the way up or on the way down.

The highest level to which the path must climb is determined by t, the largest value of j for which either h_j or b_j is 1. Then, if we define the number of horizontal links to be traversed as

$$H = \sum_{j=0}^{l-1} h_j \, ,$$

the path length becomes

$$d = 2t + H.$$

For random messages among the leaf nodes, we must now calculate the expected values of t and H.

- 3 -

Calculation of H and t

For the random message case, each bit in the address difference is equally likely to be 0 or 1. Since there are l bits of the relative address that result in a traversal of a horizontal link, one can calculate

$$H = \frac{1}{2} \times l = \frac{m+a}{4}.$$

Each h_j is equally likely to be 0 or 1, and similarly b_{l-a} will be 1 with probability $\frac{1}{2}$. Note that t can reach its maximum value l only if $a = 0$, i.e., m is even, and $b_l = 1$. Thus

$$p(t=l) = \frac{(1-a)}{2}.$$

Thus if $a = 0$, $t < l$ with probability $\frac{1}{2}$. Regardless of the value of a, however, if $t < l$, then $t = l-1$ with probability $\frac{3}{4}$, i.e., when either h_{l-1} or b_{l-1} is 1. Thus

$$p(t=l-1) = \frac{3}{4}[1-p(t=l)] = \frac{3}{4}[1-\frac{1}{2}+\frac{a}{2}] = \frac{3}{8}(a+1)$$

and in general for $j = 1,2,3,\cdots,l-1$,

$$p(t=l-j) = \frac{3}{4}[1-\sum_{i=0}^{j-1} p(t=l-j)] = \frac{3(1+a)}{2\cdot 4^j}.$$

The expected vertical distance t, is then

$$t = \sum_{j=0}^{l} j\cdot p(t=j) = \sum_{j=1}^{l} j\cdot p(t=j)$$

or more explicitly,

$$t = 1\cdot\frac{3(1+a)}{2\cdot 4^{l-1}} + 2\cdot\frac{3(1+a)}{2\cdot 4^{l-2}} + \cdots + (l-2)\cdot\frac{3(1+a)}{2\cdot 4^2} + (l-1)\cdot\frac{3(1+a)}{2\cdot 4} + l\cdot\frac{1-a}{2}$$

which can be expressed as

$$t = \frac{3(1+a)}{2\cdot 4^l}\sum_{j=1}^{l-1} j\, 4^j + l\cdot\frac{1-a}{2}.$$

Now, since it holds in general for $q > 1$

$$\sum_{j=1}^{k-1} j\cdot q^j = \frac{q}{(q-1)^2}\left[(k-1)q^k - kq^{k-1}+1\right]$$

it follows that

- 4 -

$$t = \frac{3(1+a)}{2 \cdot 4^l} \cdot \frac{4}{9} [(l-1)4^l - l \cdot 4^{l-1} + 1] + l \cdot \frac{1-a}{2}.$$

Collecting terms, rearranging and noting that

$$2^a = a + 1$$

we get

$$t = \frac{m}{2} - \frac{2}{3} + \frac{2}{3} 2^{-m} - \frac{a}{6}.$$

Thus the average leaf-to leaf path length in Hypertree I is

$$d = 2t + H = \frac{5m}{4} - \frac{4}{3} + \frac{4}{3} 2^{-m} - \frac{a}{12}$$

This formula has been used to plot the curve in Fig. 5 in the paper.

REFERENCES

[Batcher73]   K.E.Batcher, "STARAN/RADCAP hardware architecture," Proc. 1973
               Sagamore Computer Conf. Parallel Processing, pp.209-227.

[Batcher76]   K.E.Batcher, "The Flip Network in STARAN," 1976 Int. Conf. on
               Parallel Prcessing, Aug. 1976, pp.65-71.

[Benes65]   V.E.Benes, Mathematical Theory of Connecting Networks and Tele-
               phone Traffic. New York: Academic Press, 1965.

[Browning79]   S.A. Browning: "Computations on a Tree of Processors", VLSI
               Conference, CALTECH, Pasadena, CA, Jan. 22-24, 1979, Proceedings

[Bouknight72] W.J.Bouknight,   S.A.Denenberg,   D.E.McIntyre,   J.M.Randall,
               A.H.Sameh, and D.L.Slotnick, "The ILLIAC IV System," Proceedings
               of the IEEE, Vol.60,No.4, Apr. 1972, pp.369-388.

[de Bruijn46] N.G.de Bruijn, "A combinatorial prblem," Koninklijke Nederlands
               Akademie van Wetenschappen, Proceedings, Vol.49 (part 2),
               pp.758-764, 1946.

[Despain78]   A.M.Despain and D.A.Patterson, "The computer as a component,"
               submitted to CACM, 1978.

[Feng74]   T.Feng, "Data Manipulating Functions in Parallel Processors and
               Their Implementations," IEEE Trans. Comput., Vol.C-23, No.3, Mar.
               1974, pp.309-318.

- 5 -

[Goke73]   L.Goke and G.Lipovski, "Banyan Networks for Partitioning Multipro-
                cessor Systems," Proc. 1st Ann. Comput. Architecture Conf., 1973,
                p.21-28.

[Golomb61]   S.W.Golomb, "Permutations by Cutting and Shuffling," SIAM Review,
                Vol.3, Oct.1961, pp.293-297.

[Lawrie75]   D.Lawrie, "Access and alignment in an array processor," IEEE
                Trans. Comput., Vol.C-24, Dec.1975, pp.1145-1155.

[Mago79]   G.A. Mago: "A Cellular Language-directed Computer Architecture",
                VLSI Conference, CALTECH, Pasadena, CA, Jan. 22-24, 1979,
                Proceedings

[Pease77]   M.C.Pease, "The Indirect Binary n-Cube Microprocessor Array,"
                IEEE Trans. Comput., Vol.C-26,No.5, May 1977, pp.458-473.

[Schlumberger74]M.A.Schlumberger, "De Bruijn Communication Networks,"
                Stanford Ph.D. Dissertation, Computer Science Department, Stan-
                ford, California, June 1974.

[Sequin79]   C.H. Sequin: "Single-Chip Computers, the New VLSI Building
                Blocks", VLSI Conference, CALTECH, Pasadena, CA, Jan. 22-24,
                1979, Proceedings

[Siegel77]   H.J.Siegel, "Analysis Techniques for SIMD Machine Interconnection
                Networks and the Effects of Processor Address Masks," IEEE Trans.
                Comput, Vol. C26, No.2, Feb. 1977, pp.153-161.

[Stone71]   H.S.Stone, "Parallel Processing with the Perfect Shuffle," IEEE
                Trans. Comput., Vol.C-20,No.2, Feb.1971, pp.153-161.

[Swan77]   R.J.Swan, S.H.Fuller, and D.P.Siewiorek, "Cm* - A modular, multi-
                microprocessor," 1977 NCC Proceedings, pp. 645-655, June 1977.

[Thompson65]C.Thompson, "Generalized Connection Networks for Parallel Pro-
                cessor Intercommunication," IEEE Trans. Comput., Vol C-27, No.12,
                Dec. 1978, pp.1119-1125.

The Fat-Pyramid and Universal Parallel Computation Independent of Wire Delay

Ronald I. Greenberg[*]

Department of Electrical Engineering
and Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742
rig@umiacs.umd.edu

Abstract

This paper shows that a *fat-pyramid* of area $\Theta(A)$ requires only $O(\log A)$ slowdown to simulate any competing network of area A under very general conditions. The result holds regardless of the processor size (amount of attached memory) and number of processors in the competing network as long as the limitation on total area is met. Furthermore, the result is valid regardless of the relationship between wire length and wire delay. We especially focus on elimination of the common simplifying assumption that unit time suffices to traverse a wire regardless of its length, since the assumption becomes more and more untenable as the size of parallel systems increases. This paper concentrates on simulation using transmission lines (wires along which bits can be pipelined) with the message routing schedule set up off line, but it also discusses the extension to on-line simulation. This paper also examines the capabilities of a fat-pyramid when matched against a substantially larger network and points out the surprising difficulty of doing such a comparison without the unit wire delay assumption.

1   Introduction

This paper shows that the *fat-pyramid* network is a good candidate as the basis for a general-purpose parallel computer, because it can efficiently simulate any network of comparable area under general conditions. The basic structure of the fat-pyramid network was suggested by Charles Leiserson and Tom Cormen and is related to the fat-tree introduced by Leiserson [18]. The fat-pyramid may be viewed as a fat-tree in the style introduced in [12, Sec. 7] (the *butterfly fat-tree*) augmented by hierarchical mesh connections as illustrated in Fig. 1. (A variation on the butterfly-fat-tree has recently been adopted for the network structure of the CM-5 parallel computer of Thinking Machines Corporation [20].)

Ignoring the mesh connections, shown with thick lines, the fat-pyramid may be viewed as based upon a 4-ary tree in which each internal node is replaced by a collection of switches and processors are placed at the leaves. A collection of wires corresponding to an edge in the underlying 4-ary tree is referred to as a channel, and the number of wires in a channel is called its capacity. By using two types of switches with a constant number of inputs and outputs, it is possible to build fat-pyramids with essentially arbitrary channel capacities as illustrated in [19]. In this paper it suffices to make only a simple modification to the network shown in Fig. 1, in which channel capacities double at each level of the 4-ary tree.[1] A precise mathematical

---

[1]The choice of the name "fat-pyramid" for this network stems from the observation that if all channel capacities were equal to one, the result would be a network which has been called the "pyramid" by Tanimoto (and earlier a "recognition cone" by
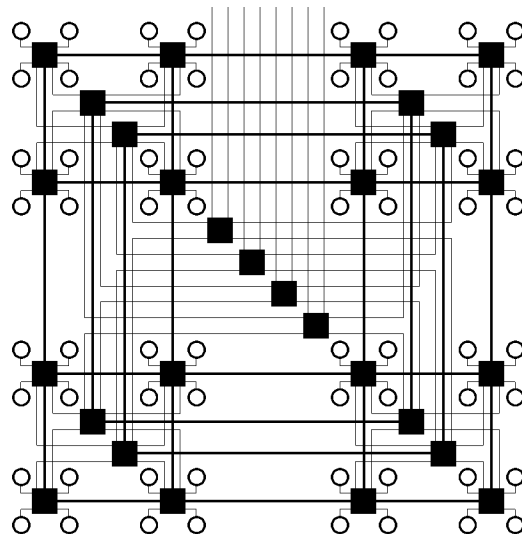
Figure 1: A fat-pyramid. Processors are placed at the leaves, represented by circles; the squares are switches. This network is obtained by superposing hierarchical mesh connections on a butterfly fat-tree. The original fat-tree connections are represented by thin lines and the mesh connections by thick lines. (A different layout of the fat-pyramid is used to obtain results independent of wire delay.)

description of the interconnection pattern for the switches in Fig. 1 can be given as follows. We begin with a collection of ordinary two-dimensional meshes at levels $0, 1, \ldots, \log_2 \sqrt{n/4}$ representing distance from the leaves in the underlying tree structure. At level h, there are 2^h copies of a $\sqrt{n/4}/2^h \times \sqrt{n/4}/2^h$ mesh. We then denote a switch by an ordered 4-tuple (h, c, x, y), where h is the level, c is the copy number of the mesh ($0 \leq c < 2^h$) at this level that contains the switch, and x and y specify a mesh position in an ordinary Cartesian coordinate system ($0 \leq x, y < \sqrt{n/4}/2^h$). Then for $0 \leq h < \log_2 \sqrt{n/4}$, switch (h, c, x, y) is connected to $(h+1, c, \lfloor x/2 \rfloor, \lfloor y/2 \rfloor)$ and $(h+1, c+2^h, \lfloor x/2 \rfloor, \lfloor y/2 \rfloor)$.

An important feature of the fat-pyramid is its status as a universal network without the usual assumption that unit delay suffices to traverse a wire of any length. This issue becomes more and more important as we move towards increasingly massive parallelism. In this regard, the fat-pyramid has an advantage over the fat-tree, though these networks are both appropriate to a view in which hardware cost is measured as area or volume under standard VLSI models. By accounting for the difficulty of running the wires in a massive network, these VLSI models provide a more realistic measure of cost than merely bounding processor count and node degree. Though the results in this paper are stated in terms of area in a two-dimensional design space (constant number of chip layers), the extension to three-dimensions is fairly straightforward [9] using the ideas in [11].

The basic mode of operation assumed for fat-pyramids and any other parallel computer will be as in the distributed random-access machine (DRAM) model of Leiserson and Maggs [21]. All memory is local to the processors, and a processor can read, write, and perform arithmetic and logical functions on values stored in its local memory. It can also read and write memory in other processors by routing messages through an underlying network. In the bulk of this paper, messages are viewed as being composed of indivisible "packets", and delay along wires is measured in terms of the time to transmit a complete packet; the end of Section 3 explains why there is limited change to the results if we switch from this "word model" to a "bit model". There is also no need to worry about messages having varying length; we can think of messages as being divided up into packets of standard size and henceforth treat "message" as synonymous with "packet".

Uhr) [25, p. 3]. The addition of mesh connections to the fat-tree is also similar to the introduction of "brother" connections in trees to obtain the X-Tree network [8, 23]. (The term "fat-pyramid" should not be confused with a recent independent use of the term by other authors.)

It is also convenient to assume that operation of competing networks is divided into separate (alternating) phases of intraprocessor computation and interprocessor communication. Thus, to bound asymptotic simulation time, it will suffice to take the maximum of the overheads for simulation of the computation and simulation of the communication. In fact, this approach is valid even if the competing network interleaves computation and communication in a more complicated fashion. The validity of the simplification is established rigorously in [9, Sec. 4.5] in the case of unit wire delay; the case of nonunit delay can be handled by a similar approach of prioritizing instructions and packet deliveries according to their completion times in the competing network. Throughout this paper, we shall say that network A can simulate network B with overhead μ if, for any t, the operations performed by network B in time t can be performed by network A in time $t\mu$.

Intuitively, the fat-pyramid combines the strengths of the fat-tree and the mesh. A fat-tree can efficiently simulate any network of comparable area under the unit wire delay assumption [3, 12, 14, 18]. But the straightforward layout of the fat-tree has wires of length $\Theta(\sqrt{A})$ near the root (and little improvement is possible). If the fat-tree is used to simulate a mesh, any mapping of processors in the mesh to processors in the fat-tree will place on "opposite sides of the root" some processor pairs that are adjacent in the mesh. If the time to transmit a packet is a linear function of wire length, the fat-tree will require $\Omega(\sqrt{A})$ time to route messages between such a pair of processors. But the mesh network could be performing a computation requiring only nearest neighbor communication so that the fat-tree simulation would have polynomial ($\Omega(\sqrt{A})$) overhead, which is much worse than the polylogarithmic overhead attainable in the unit wire delay case. The mesh, on the other hand, is universal in the case of linear wire delay. But if delay is less sensitive to wire length, the mesh may also suffer polynomial slowdown as can be seen by considering simulation of a tree. Since a tree of area A (using the H-tree layout illustrated in e.g. [17]) contains essentially the same number of processors as a mesh of area A, the mapping of tree processors to the mesh will expand some routing path between processors from $O(\log A)$ switches in the tree to $\Theta(\sqrt{A})$ in the mesh. The fat-pyramid, in contrast to the fat-tree or the mesh, can achieve polylogarithmic simulation overhead under essentially any interesting model of wire delay.

The remainder of this paper is organized as follows. Section 2 discusses the fat-tree variation used as the basis for the fat-pyramid and its ability to efficiently simulate any network of comparable area given unit wire delay. Section 3 shows how the fat-pyramid can be used to extend the ideas of Section 2 to nonunit wire delay. Section 4 considers the overhead required by a universal network to simulate a network of larger area; these results are proved only for unit wire delay. Section 5 contains concluding remarks and open questions.

2   Routing and simulation overhead on the fat-tree

To facilitate explanation of the universality properties of the fat-pyramid, we first examine details and properties of the particular variation of the fat-tree used as its basis in this paper. In this section, we retain the assumption of unit wire delay.

We begin by considering the area requirements of the fat-tree. A standard modeling approach involves thinking of network nodes as points in a grid and wires as edge-disjoint paths through the grid, but it is somewhat unrealistic to view the network nodes as occupying constant area. Since we generally want each processor to be capable of addressing each other processor, a fat-tree of area A should have $\Omega(\log A)$ memory per processor.[2]

In order to pack in a maximum number of processors of area $\Theta(\log A)$, we consider a fat-tree with channel capacities doubling at increasing levels of the underlying 4-ary tree as in Fig. 1, except that each circle in that figure is replaced by an H-tree layout of $\log_2 A$ processors. Each such H-tree block is of size $\Theta(\log A) \times \Theta(\log A)$, and we can derive the area of the fat-tree by solving a recurrence relation for the side

---

[2]The requirement is actually $\Omega(\log n)$, where n is the number of processors. But $\log n$ is $\Omega(\log A)$ unless n is $o(A^\epsilon)$ for every $\epsilon > 0$, which would imply that competing networks of the same area could not be simulated in reasonable time since they could have $\omega(A^{1-\epsilon})$ times as many processors. Comments preceding Theorem 2 and in the beginning of the proof of Lemma 3 provide some justification for assuming henceforth that processors occupy a square of area $\Theta(\log A)$; if more area is required to include sophisticated operations in the processor instruction set, the situation can be handled as in [9, 10].

length $S(b)$ of a fat-tree with b H-tree blocks. Since the channel capacities above the H-tree blocks double at every level, we have a channel capacity proportional to $\sqrt{b}$ at the root, and we can write the recurrence

$$S(b) = 2S(b/4) + O(\sqrt{b}) \text{ , and } S(1) = \Theta(\log A) \text{ ,}$$

with solution

$$S(b) = \Theta(\sqrt{b}\log A) \text{ .}$$

Thus, we can build a fat-tree on $A/\log_2 A$ processors of area $\Theta(\log A)$ in area $\Theta(A)$ since that corresponds to b=$A/\log_2^2 A$. (Two notes are in order. First, the recurrence assumes switches of constant area, but switches of area $\Theta(\log A)$ to examine and compare full processor addresses or message priorities can be accommodated in the modified layout discussed in Section 3. Second, because of the need to precisely bound the wire density, we should think of packet transmission as occurring in a bit-serial fashion. We can still think of unit time as the time to transmit a packet of $\Theta(\log A)$ bits along a wire.)

Now we can examine the simulation overhead required by a fat-tree of area A to simulate other networks of area A. We can perform this comparison without placing any restriction on the number or size of processors of the competing network. Here, processor size refers solely to the amount of attached memory; we assume that processors of networks to be compared have the same instruction set and are equally well-engineered to provide the same operations at the same cost in time and space. If the competing network has larger processors than the fat-tree, we can subdivide the memory of these processors and view them as a larger number of smaller processors, so we concentrate on the situation in which the processors of the competing network are no larger than those of the fat-tree. We can use a straightforward geometric mapping of processors of the competing network to fat-tree processors and powerful packet routing results [15, 16] to show that the fat-tree is universal under unit wire delay. For now, only the off-line result is considered, i.e., that there exists an efficient means of scheduling the messages of competing networks on the fat-tree. For this result, the following packet routing lemma suffices; the term *congestion* refers to the maximum number of packets that must cross a single edge (wire) in the network, and *dilation* refers to the maximum number of edges that must be traversed by a single packet.

Lemma 1 ([15]) *For any set of packets with edge-simple paths having congestion c and dilation d, there is a schedule having length $O(c+d)$ and maximum queue size $O(1)$ in which at most one packet traverses each edge of the network at each step.*

We can now proceed with the fat-tree universality result:

Theorem 2 *A fat-tree F of area $\Theta(A)$ can simulate any network of area A with $O(\log A)$ overhead.*

Proof. Given a competing network R of area A, we recursively bisect it in the straightforward geometric fashion, cutting nonsquare pieces in the shorter direction, until we have $A/\log_2 A$ pieces. These pieces of R are mapped to the $n = A/\log_2 A$ processors of F so that the recursive bisection of R matches the natural recursive bisection of F obtained by cutting at the root and then at the roots of the subtrees and so on. This yields at most $\log_2 A$ processors of R whose computation must be simulated by a single processor of F.[3]

Having obtained a computation overhead of $O(\log A)$, we now analyze the communication overhead involved in routing messages of R through F. The message traffic in channels of F can be bounded by using an assumption that is certainly reasonable for VLSI technologies: the number of packets that can leave an area in unit (packet transmission) time is proportional to the perimeter of the area. The perimeter of a piece of R corresponding to a subtree of $n/2^l$ processors in F is $O(\sqrt{A}/2^{l/2})$ (excluding the perimeter of R itself if R is nonsquare). Since the capacity of a channel on top of a subtree of $n/2^l$ processors in F is at least $\sqrt{(n/\log_2 A)}/2^l$ and $n = A/\log_2 A$, the number of packets entering or leaving a subtree in unit time divided

---

[3]A processor that is split by one or more of the bisection lines in R can be mapped to any one of the processors in F to which its pieces would correspond. This does not alter the bounds on computation or communication overhead, because a bisection line can only split a number of processors equal to its length and because the extra number of messages generated by the split processors in unit time is at most proportional to the number of such processors.

by the number of wires in the corresponding channel is $O(\log A)$. In fact, the packets can be routed through F so that there are $O(\log A)$ packets on any single wire. Indeed, this outcome occurs with high probability if each message is routed by picking at random the root switch it should pass through (which fully determines its path) [14, 16].

We have now established that for a set of packets traveling in R during any unit period of time, at most $O(\log A)$ packets must traverse any given wire in F. Furthermore each packet must traverse at most $O(\log A)$ wires to get from its source to its destination in F. In other words, the congestion and dilation are at most $O(\log A)$. Thus, by Lemma 1, in $O(\log A)$ time steps, we can completely route in F all the messages that travel in R during any unit of time.   ∎

3   The fat-pyramid and nonunit wire delay

In this section we consider the effect of dropping the unit wire delay assumption. The general graph layout framework developed by Bhatt and Leighton [6] shows that there is enough room in our fat-tree layouts to build sufficiently large drivers for each wire to keep the wire delay constant in the capacitive model. This section shows that even if this constant switching time is not the dominant determiner of wire delay, an appropriate layout of the fat-pyramid yields a universal network with $O(\log A)$ simulation overhead. The key to this result is that a routing path of length δ in a competing network of area A corresponds to a path of length $O(\delta + \log A)$ in the fat-pyramid.

It should be noted that it is reasonable to assume wire delay to be no worse than linear in wire length, since repeaters (extra switches) can always be used to reduce delay to linear. Linear wire delay would be the correct model if technology could be improved to the point where only speed of light limitations constrain the time to switch a length of wire. Then, the measure of unit time would be much smaller, but linear wire delay would be required of any competing network.

It is also helpful to assume a mild "regularity" condition on the wire delay function. (Similar regularity conditions are used elsewhere in the literature (e.g., [5, 7, 17],[1, p. 280]) in order to obtain results about large classes of functions.) Specifically, let $w(\delta)$ denote the time required to transmit a packet along a wire of length δ; then we seek two properties for the function w. First, w should be nondecreasing, and second it should satisfy the following condition:

Definition: A function w is said to satisfy Condition C1 if there exists a constant c such that

$$\frac{w(\delta + x)}{w(\delta)} \leq \frac{\log_2 \delta + x}{\log_2 \delta}$$

for all $x \geq 0$ and $\delta \geq c$.

It should be noted that Condition C1 is satisfied by most functions likely to be of interest in the context of wire delay. For example, it is satisfied by all functions $w(\delta)$ of the form $c\delta^q \log_2^k \delta$ for constants c, q, and k such that either $q < 1$, or $q = 1$ and $k \leq 0$. One way to see that all of these functions satisfy Condition C1, is to observe that they satisfy a simpler regularity condition C2, which implies C1.

Definition: A function w is said to satisfy Condition C2 if there exists a constant c such that

$$\frac{w(\delta + x)}{w(\delta)} \leq \frac{\delta + x}{\delta}$$

for all $x \geq 0$ and $\delta \geq c$.

Condition C2 implies condition C1 because $1 + x/\delta \leq 1 + x/\log_2 \delta$ for any $x \geq 0$ and $\delta > 0$.

(Without changing the asymptotic results given below, we can actually weaken conditions C1 and C2 in order to admit an even larger class of functions than already mentioned. Specifically we could define conditions C1 and C2 to be that the old conditions are satisfied to within a constant factor. Then the

5
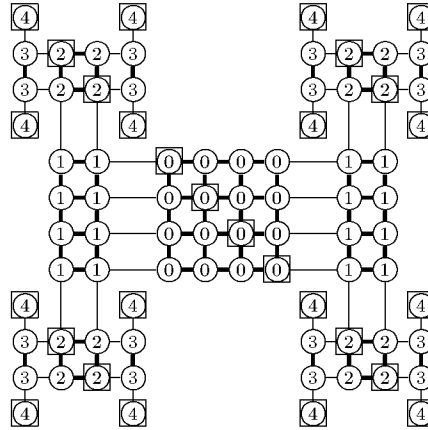
Figure 2: Embedding the fat-pyramid in the tree of meshes. The squares represent switches of the fat-pyramid; laying out the edges connecting switches in the butterfly fat-tree requires using each edge of the tree of meshes at most twice.

conditions are satisfied by any function w satisfying $c_1 \delta^q \log_2^k \delta \le w(\delta) \le c_2 \delta^q \log_2^k \delta$ for sufficiently large δ, with q and k as before and positive constants c_1 and c_2.)

To obtain results independent of wire delay, we must consider a regular layout of a fat-pyramid, that is, one in which the switches at any given level of the tree are regularly spaced throughout the layout. We can produce such a layout by using the "fold-and-squash" technique of Bhatt and Leighton [6, pp. 325–326] and Thompson [26, pp. 36–38]. To see the effect on wire lengths in the fat-pyramid, it is helpful to think of embedding the butterfly fat-tree into the tree of meshes graph, performing the fold-and-squash transformation, and then adding the fat-pyramid's hierarchical mesh connections. We will actually embed just the switches shown as black squares in Fig. 1, while keeping in mind that each such bottom-level switch must be attached in the final layout to four $\Theta(\log A) \times \Theta(\log A)$ H-trees composed of $\log_2 A$ processors of area $\Theta(\log A)$. Fig. 2 illustrates the embedding of switches of the fat-tree into a 4×4 tree of meshes. The fold-and-squash transformation of the tree of meshes is performed by first folding the connections between the mesh at level zero (comprised of all the nodes labeled zero in Fig. 2) and the meshes at level one so that the two smaller meshes fit on a second layer directly over the large mesh. Then the meshes at level two are folded onto a third layer and so forth up to $\log_2(A / \log_2^2 A) = O(\log A)$ levels. Finally, the layers are offset and projected onto the plane as illustrated in Fig. 3. The maximum length of tree-of-meshes edges becomes $O(\log A)$ even with the four $\Theta(\log A) \times \Theta(\log A)$ H-trees clustered around the bottom-level switches of the fat-tree.[4] Finally, only a constant factor increase in area is required to add the fat-pyramid's hierarchical mesh connections. (In addition, since the switches at any given level are separated by a distance of $\Theta(\log A)$, there is room to expand the switches to occupy area $\Theta(\log A)$; the layout can also be massaged to make such switches square if desired.)

In the regular layout of a fat-pyramid of area A, wires connected to a switch h levels up from the H-tree blocks in the underlying 4-ary tree, are of length $O(2^h \log A)$. To see this, observe that each edge of the fat-pyramid that is h levels up is mapped to a path of $O(2^h)$ tree-of-meshes edges.

Messages in the fat-pyramid are routed over the same paths as in the fat-tree, except that we allow each message to take one shortcut via one or two of the new hierarchical mesh edges. More precisely, the routing path is formed by going up fat-tree edges towards a randomly selected switch at the root of the underlying 4-ary tree until a switch is reached that is adjacent horizontally, vertically, or diagonally in the mesh at that level to a switch from which the destination can be reached by going down fat-tree edges. Certainly the dilation is not increased by incorporating such shortcuts, and, in fact, the congestion for any set of messages in the fat-pyramid is also within a constant factor of the congestion in the fat-tree. The congestion in tree

---

[4]An explanation at greater length of the fold-and-squash technique can be found in [11].
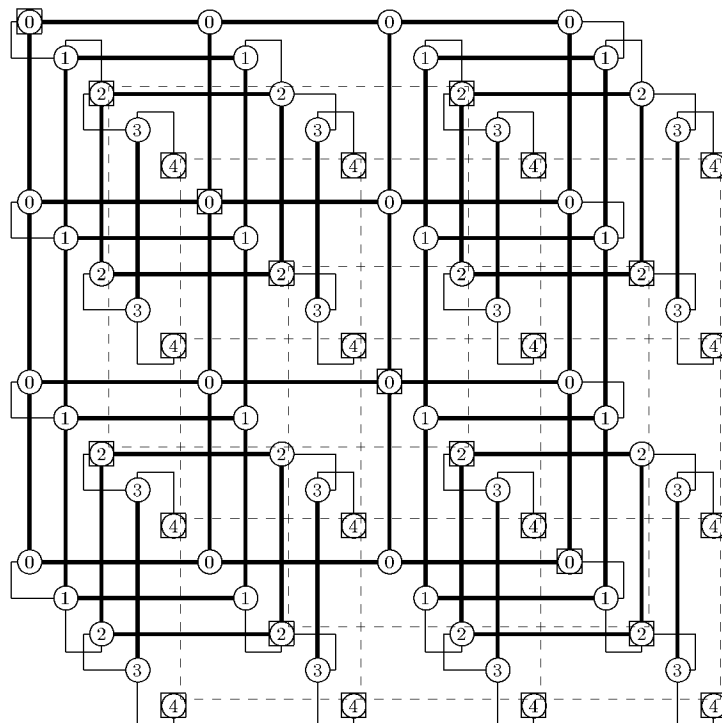
Figure 3: Layout of the fat-pyramid after folding and squashing. As before, the squares represent fat-pyramid switches, and fat-tree edges are routed through the tree of mesh edges shown with thin and thick solid lines. (The connections between the layers obtained through folding, the thin lines, are routed by using an auxiliary row or column located in the direction of the corresponding fold and lying parallel to it.) The insertion of the fat-pyramid's hierarchical mesh connections is illustrated with dashed lines.

edges does not increase, because when a message takes a shortcut, it does not traverse any tree edge it was not already destined to traverse; in each mesh edge, the messages that pass may arrive from only from a constant number of tree edges.

Now we can show that the mapping of competing networks to a fat-pyramid does not stretch any wires by very much.

Lemma 3 *Any network R of area A can be mapped to a fat-pyramid F of area $\Theta(A)$ so that any message following a path of length δ in R travels only $O(\delta + \log A)$ distance in F.*

Proof. First note that we can assume R is square. (If R is not square, it can be converted to a square layout of at most 1.8 times as much area with each wire at most 3 times as long as in the original layout [2].) Now we can map the processors of a square network R to F in a straightforward geometric fashion as in Section 2. Then two processors separated by distance δ in R are at most $\lceil \delta / \log_2 A \rceil$ H-tree blocks apart horizontally or vertically in F. Since two subtrees on $(\lceil \delta / \log_2 A \rceil)^2$ H-tree blocks in the underlying 4-ary tree that are physically adjacent (horizontally, vertically, or diagonally) must suffice to cover such a pair of processors, the routing path connecting these processors needs only to go up $\log_2(\lceil \delta / \log_2 A \rceil)$ levels above the H-trees and use two mesh edges. Since any wire connected to a switch h levels up is of length $O(2^h \log A)$, and the distance traveled within H-tree blocks is $O(\log A)$, the length of the routing path connecting processors at distance δ in R is $O(\delta + \log A)$. ∎

We can now state our main result for nonunit wire delay, relying only on our regularity condition for wire delay. As before, we focus on off-line scheduling in the packet model; extensions are discussed afterwards. It is also important to note that the availability of transmission lines is assumed, so that wires can contain a number of packets equal to the delay of the wire at any given time. Though it is natural to think of a transmission line as a device for pipelining bits, it is only more conservative to think of pipelining packets and to continue measuring delay in terms of packet steps. (Use of transmission lines is occurring in real parallel machines, e.g., see the references in [22].)

Theorem 4 *Using transmission lines, a fat-pyramid F of area $\Theta(A)$ can simulate any network of area A with $O(\log A)$ overhead.*

Proof. To apply the packet routing results of Leighton, Maggs, and Rao [15, 16], we can imagine additional switches on each wire of the fat-pyramid in number equal to the delay for that wire. With the inclusion of these imaginary (and trivial) switches, we can view the routing problem as fitting into the unit wire delay framework; we have simply increased the maximum distance (in terms of switches) that packets must travel.

Now consider any set of messages generated by the competing network in which the maximum *physical* distance that a message travels in the competing network is δ. Let T be the time required to deliver the set of messages in the competing network, and note that $T \geq w(\delta)$. Also, the congestion created by this message set is $O(T \log A)$. Furthermore, the maximum number of fat-pyramid edges which a message must traverse is $2 \log_2 \delta$, each containing at most $w(\delta + \log_2 A)$ real and imaginary switches. (Actually, the number of switches should be $w(O(\delta + \log A))$, but the results below remain valid because w is at most linear.) Thus the communication overhead μ can be bounded as follows, based on Leighton, Maggs, and Rao's result that routing can be performed in time proportional to congestion plus dilation:

$$
\begin{aligned}
\mu \;\;\leq\;\; & O\!\left(\frac{T \log A + w(\delta + \log_2 A) \log \delta}{T} \right) \\[2mm]
\leq\;\; & O\!\left(\frac{T \log A}{T} \right) + O\!\left(\frac{w(\delta + \log_2 A) \log \delta}{w(\delta)} \right) \\[2mm]
\leq\;\; & O(\log A) + O\!\left(\frac{(\log \delta + \log A) \log \delta}{\log \delta} \right) \\[2mm]
\leq\;\; & O(\log A) \;,
\end{aligned}
$$

where the third line follows from regularity condition C1. The computation overhead is also $O(\log A)$ as in Section 2. ∎

The simulation can also be performed on-line, albeit with some loss of efficiency in the case of nonunit wire delay. In the unit delay case, there is no loss of efficiency, due to analysis of packet routing on a "leveled" network [14, 16]. With nonunit wire delay, however, it is not apparent how to make use of the framework of leveled networks. But the algorithm of Leighton, Maggs, and Rao for routing on-line in $O(c + d \log(Nd))$ steps (with high probability) in general networks, where c is congestion, d is dilation, and N ($\leq A$ here) is the number of packets [15], can be applied to yield a simulation overhead of $O(\log^2 A)$.[5] In fact, the overhead can be improved to $O(\log^2 A / \log \log A)$ using a variation on their technique appearing in [13, 24]. (The question of routing in networks without transmission lines is also considered in [13], and the results there may prove useful to the construction of universal networks without transmission lines.)

Finally, it is desirable to consider the overhead in *bit-times* for on-line routing, since on-line routing schemes may need to tack $\Omega(\log A)$ address bits onto messages that are of constant size in the competing network. The overhead in bit-times for nonunit wire delay is $O(\log^2 A)$, which can be shown as in the proof of Theorem 4 by using the following proof that for unit wire delay, $O((c + d + \log N) \log(Nd))$ bit steps suffice to route packets of $\log_2 N$ bits with high probability $(1 - O(\frac{1}{Nd}))$. We begin by assigning each packet an initial (integral) delay chosen randomly and uniformly from the interval $[1, \alpha c]$, for a constant α to be specified later, and consider the "schedule" in which there is no other waiting except that the bits of any given message are sent one after another. This yields a "schedule" of length $O(c + d + \log N)$ bit-steps in which there may be bits from several different messages traversing an edge at a given time. But the probability that more than $\log_2(Nd)$ packets have a bit traversing a given edge at a given bit-step is at most

$$\binom{c}{\log_2(Nd)} \left(\frac{\log_2 N}{\alpha c}\right)^{\log_2(Nd)} \leq \left(\frac{e}{\alpha}\right)^{\log_2(Nd)} .$$

This probability multiplied by the number of choices for an edge and a bit step is less than $\frac{1}{Nd}$ for a sufficiently large constant α. So with high probability, we can obtain a schedule of length $O((c + d + \log N) \log(Nd))$ in which only one bit traverses an edge at a time, by having switches cycle through incoming packets forwarding one bit at a time.

4   Simulating larger networks

This section obtains upper bounds on the time required by a fat-tree to simulate networks that occupy more area but have the same amount of area devoted to processors. The reason for the latter restriction is that for any significant difference in memory, there are computations which can be performed in the larger amount of memory space but not in the smaller amount of memory space. Rather than placing restrictions on the type of computation, it is probably more meaningful to look at restrictions on the way that space is allocated. That is, if the larger network uses the same amount of processor area (including memory) and simply uses more interconnect area, then we can make meaningful comparisons between the networks. As would be expected, simulation difficulty increases as the area of the competing network does, but only up to a certain threshold beyond which extra area does not help the competition.

In this section we return to a reliance on the unit wire delay assumption due to a change in the means of mapping competing networks to the universal network. The results are, of course, applicable to the fat-pyramid as well as the fat-tree, but it is an open problem to show that unit wire delay is unnecessary when a universal network simulates a larger network.

As we open up the issue of restricting the processor area used by competing networks, it may seem natural to ask about situations in which the competing network has less processor area than is allowed for the universal network. Indeed, we could have considered this question earlier when comparing networks of the same total area. But when the processor area of competing networks is so restricted, the best results are obtained by tailoring the universal network to the particular mix of processor and interconnect area, with the most difficult case occurring when the competing network has no less processor area than the universal

---

[5]Actually, the universal network should also be modified to have processors that are larger and fewer (both by a factor of $\Theta(\log A)$) in order to accommodate the necessary queues of $O(\log(Nd))$ packets.

network. Thus, the results given so far are worst-case results for simulating networks of essentially the same total area. In this sense, the universal networks described above are the best known to build in a given area. Rather than digress to networks tailored to particular mixes of processors and interconnect, we now ask how well the networks discussed so far can do when they are actually matched against networks with larger area but no greater processor area.

In what follows, we let A_X represent the area of network X. We are, of course, interested in the case where the competing network R has at least as much area as F, i.e., $A_R \geq A_F$; when $A_R \leq A_F$, our earlier results apply.

We use the same basic strategy as before for demonstrating universality results; that is, we recursively bisect the competing network and map appropriate pieces to the fat-tree processors. But when the competing network may have greater area than processor area, extra care is required to ensure that the decomposition is balanced; that is, when we bisect the area of the competing network, we must also bisect the set of processors of the competing network. Fortunately, we can invoke the general theory developed by Bhatt and Leighton [6] and, in a fashion that is cleaner for our purposes, by Leiserson [18]. (It is not desirable to use this approach when unnecessary due to a "loss of locality" in the mapping, which destroys the results on nonunit wire delay in Section 3.) These results tell us that since the competing network of area A_R has a decomposition using cuts of size $\sqrt{A_R}/2^{l/2}$ at level l, it has a balanced decomposition using cuts of the same size (up to a constant factor). Keeping this fact in mind, we can prove the following theorem:

Theorem 5 *A universal fat-tree of area $O(A_F)$ can simulate any network of total area $A_R \geq A_F$ and processor area at most A_F with $O(\sqrt{A_R/A_F} \log A_F)$ overhead.*

Proof. Using a balanced decomposition as described above for the competing network R, we find that the ratio of messages to channel capacity for a set of messages delivered by R in unit time is

$$O\left(\frac{\sqrt{A_R}/2^{l/2}}{\sqrt{A_F/\log^2 A_F}/2^{l/2}}\right)$$

at level l from the root of the fat-tree. Thus, the communication overhead, as determined by congestion plus dilation, is $O(\sqrt{A_R/A_F} \log A_F)$, which dominates the $O(\log A_F)$ computation overhead.   ■

When the area of the competing network is much larger than the area of the universal fat-tree, we can actually do better than is suggested by Theorem 5. When A_R is $\Omega(A_F^2)$, the competing network is limited more by the restriction on processor area than by its total area. This is true because communication out of a piece of network R is limited not only by the perimeter of that piece but also by the perimeter of the processors in the piece. Thus, only $O(A_F/2^l)$ messages can leave a piece of R at level l in a balanced decomposition. Dividing by fat-tree channel capacity to determine the congestion, we obtain the following result:

Theorem 6 *A universal fat-tree of area $O(A_F)$ can simulate any network having processor area at most A_F with $O(\sqrt{A_F} \log A_F)$ overhead.*   ■

5   Conclusion

This paper has shown that a fat-pyramid network can efficiently simulate any other network built in the same amount of area. The results allow an essentially arbitrary relationship of delay to wire length and allow arbitrary processor size and density in competing networks.

This paper has also obtained bounds on the time required by a universal network to simulate larger networks of the same total processor area. Unfortunately the latter result is not readily extended to the case of nonunit wire delay, due to the use of decomposition trees that are balanced. It is an open question whether or not this extension can be achieved. Perhaps it could be shown that there is a balanced decomposition tree which will not force nearby processors to be mapped too far from each other in the universal fat-pyramid.

Another open question is whether on-line simulation by a universal network can be performed with overhead better than $O(\log^2 A)$ bit times. Of the known on-line results, only the overhead in the word model for unit wire delay ($O(\log A)$) is known to be optimal (by an AT^2 lower bound of Bay and Bilardi [3, 4]).

Finally, it would be desirable to find networks that have good universality properties without using transmission lines.

Acknowledgements

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, MA, 1974.

[2] R. Aleliunas and A. L. Rosenberg. On embedding rectangular grids in square grids. *IEEE Trans. Computers*, C-31(9):907–913, Sept. 1982.

[3] P. Bay and G. Bilardi. Deterministic on-line routing on area-universal networks. In *31st Annual Symposium on Foundations of Computer Science*, pages 297–306. IEEE Computer Society Press, 1990.

[4] P. E. Bay. *Area-Universal Interconnection Networks for VLSI Parallel Computers.* PhD thesis, Department of Computer Science, Cornell University, May 1992.

[5] J. L. Bentley, D. Haken, and J. B. Saxe. A general method for solving divide-and-conquer recurrences. Technical Report CMU-CS-78-154, Department of Computer Science, Carnegie-Mellon University, Dec. 1978.

[6] S. N. Bhatt and F. T. Leighton. A framework for solving VLSI graph layout problems. *Journal of Computer and System Sciences*, 28(2):300–343, Apr. 1984.

[7] R. P. Brent and H. T. Kung. Fast algorithms for manipulating formal power series. *Journal of the ACM*, 25(4):581–595, Oct. 1978.

[8] A. M. Despain and D. A. Patterson. X-Tree: A tree structured multi-processor computer architecture. In *Proceedings of the 5th Annual International Symposium on Computer Architecture*, pages 144–151. ACM/IEEE, 1978.

[9] R. I. Greenberg. *Efficient Interconnection Schemes for VLSI and Parallel Computation.* PhD thesis, Department of Electrical Engineering & Computer Science, Massachusetts Institute of Technology, Aug. 1989. MIT/LCS/TR-456.

[10] R. I. Greenberg. The fat-pyramid: A robust network for parallel computation. In W. J. Dally, editor, *Advanced Research in VLSI: Proceedings of the Sixth MIT Conference*, pages 195–213. MIT Press, 1990.

[11] R. I. Greenberg and C. E. Leiserson. A compact layout for the three-dimensional tree of meshes. *Applied Mathematics Letters*, 1(2):171–176, 1988. Also see erratum in vol. 1, no. 3, p. 315.

[12] R. I. Greenberg and C. E. Leiserson. Randomized routing on fat-trees. In S. Micali, editor, *Randomness and Computation.* Volume 5 of *Advances in Computing Research*, pages 345–374. JAI Press, 1989.

[13] R. I. Greenberg and H.-C. Oh. Packet routing in networks with long wires. In *Proceedings of the 30th Annual Allerton Conference on Communication, Control, and Computing*, pages 664–673, 1992. Revised version in *Journal of Parallel and Distributed Computing*, 31(2):153–158, December 1995.

[14] F. T. Leighton, B. M. Maggs, A. G. Ranade, and S. B. Rao. Randomized routing and sorting on fixed-connection networks. *Journal of Algorithms*, 17(1):157–205, July 1994.

[15] F. T. Leighton, B. M. Maggs, and S. B. Rao. Packet routing and job-shop scheduling in O(congestion + dilation) steps. *Combinatorica*, 14(2):167–180, 1994.

[16] T. Leighton, B. Maggs, and S. Rao. Universal packet routing algorithms. In *29th Annual Symposium on Foundations of Computer Science*, pages 256–269. IEEE Computer Society Press, 1988.

[17] C. E. Leiserson. Area-efficient graph layouts (for VLSI). In *21st Annual Symposium on Foundations of Computer Science*, pages 270–281. IEEE Computer Society Press, 1980.

[18] C. E. Leiserson. Fat-trees: Universal networks for hardware-efficient supercomputing. *IEEE Trans. Computers*, C-34(10):892–901, Oct. 1985.

[19] C. E. Leiserson. VLSI theory and parallel supercomputing. In C. L. Seitz, editor, *Advanced Research in VLSI: Proceedings of the Decennial Caltech Conference on VLSI*, pages 5–16. MIT Press, 1989.

[20] C. E. Leiserson, Z. S. Abuhamdeh, D. C. Douglas, C. R. Feynman, M. N. Ganmukhi, J. V. Hill, W. D. Hillis, B. C. Kuszmaul, M. A. S. Pierre, D. S. Wells, M. C. Wong, S.-W. Yang, and R. Zak. The network architecture of the connection machine CM-5. In *Proceedings of the 4th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 272–285. Association for Computing Machinery, 1992.

[21] C. E. Leiserson and B. M. Maggs. Communication-efficient parallel algorithms for distributed random-access machines. *Algorithmica*, 3:53–77, 1988.

[22] S. L. Scott and J. R. Goodman. The impact of pipelined channels on k-ary n-cube networks. *IEEE Trans. Parallel and Distributed Systems*, 5(1):2–16, Jan. 1994.

[23] C. H. Séquin, A. M. Despain, and D. A. Patterson. Communication in X-TREE, a modular multiprocessor system. In *ACM 78: Proceedings 1978 Annual Conference*, pages 194–203, 1978.

[24] D. B. Shmoys, C. Stein, and J. Wein. Improved approximation algorithms for shop scheduling problems. In *Proceedings of the 2nd Annual SIAM Symposium on Discrete Algorithms*, pages 148–157, 1991.

[25] S. L. Tanimoto. Towards hierarchical cellular logic: Design considerations for pyramid machines. Technical Report 81-02-01, Department of Computer Science, University of Washington, Feb. 1981.

[26] C. D. Thompson. *A Complexity Theory for VLSI*. PhD thesis, Department of Computer Science, Carnegie-Mellon University, 1980.

Using Sparse Crossbars within LUT Clusters

Guy Lemieux
Dept. of Electrical and Computer Engineering
University of Toronto
Toronto, Ontario, Canada   M5S 3G4
lemieux@eecg.toronto.edu

David Lewis
Dept. of Electrical and Computer Engineering
University of Toronto
Toronto, Ontario, Canada   M5S 3G4
lewis@eecg.toronto.edu

ABSTRACT

In FPGAs, the internal connections in a cluster of lookup tables (LUTs) are often fully-connected like a full crossbar. Such a high degree of connectivity makes routing easier, but has significant area overhead. This paper explores the use of sparse crossbars as a switch matrix inside the clusters between the cluster inputs and the LUT inputs. We have reduced the switch densities inside these matrices by 50% or more and saved from 10 to 18% in area with no degradation to critical-path delay. To compensate for the loss of routability, increased compute time and spare cluster inputs are required. Further investigation may yield modest area and delay reductions.

1. INTRODUCTION

A recent trend in FPGA architectural design is to use a *clustered architecture*, where a number of lookup tables (LUTs) are grouped together to act as the configurable logic block. The motivation for using clusters is manifold: to reduce area, to reduce critical path delay, and to reduce CAD tool runtime [1, 2, 9, 10]. This trend is followed by FPGAs from Xilinx's Virtex and Spartan-II families, as well as Altera's APEX and ACEX products. All of these FPGAs are based on clusters of 4-input lookup tables.

In a clustered architecture, the LUT inputs can be chosen from two sources: 1) a set of shared *cluster inputs*, which are signals arriving from other clusters via the general purpose routing, or 2) from *feedback connections*, which are the outputs of LUTs in this cluster. It has been common to assume that these internal cluster connections are *fully populated* or *fully connected*, meaning every LUT input can choose any signal from all of the cluster inputs and feedback connections combined. This arrangement can also be viewed as a full crossbar, where a switch or crosspoint exists at the intersection point of every LUT input and every cluster input or feedback connection.

In this paper, it is assumed that the connections within the cluster are made by multiplexers driving the LUT inputs, called *LUT input multiplexers*. These multiplexers tend to have a large number of inputs and, after including the requisite input buffers and controlling SRAM bits, contribute significantly to FPGA area.

A clustered FPGA is composed of a number of *cluster tiles* which are repeated in a simple array pattern during layout. Each tile is complete in that it includes the cluster logic (the flip-flops, LUTs, and LUT input multiplexers) as well as the general routing to interconnect them. Based on an area model stated later in Section 2, the LUT input multiplexers alone can consume 24 to 33% of the transistor area in a cluster tile. A breakdown of the area estimates for a number of such tiles is provided in Table 1.

The significant amount of area required by the LUT input multiplexers motivated the idea of removing switches from the full crossbar, or *depopulating* it, to result in a sparse crossbar. Naturally, depopulating the cluster raises the following questions:

1. Will depopulation save area, require greater routing area, or create unroutable architectures?

2. Will depopulation reduce or increase routing delays?

3. What amount of depopulation is reasonable?

4. How much area or delay reduction can be attained, if any?

5. What are the other effects of depopulating the cluster?

This paper addresses these questions using an experimental process of mapping benchmark circuits to clustered FPGA architectures and measuring the resulting area and delay characteristics.

1.1 Comparison to Prior Work

The use of fully-connected clusters likely stems from previous work [12] which suggests that inputs of a 4-LUT be fully connected to the routing channel. This provides enough routing flexibility to obtain minimum channel widths in non-clustered architectures, the area metric in use at that time. Since then, clustered architectures have become prevalent, CAD tools have improved, and area metrics have become more detailed.

Reducing the amount of connectivity within the cluster was recently explored using a simple striped switch layout [11]. Rather than modify the router, the T-VPACK packing algorithm was altered in such a way that routability of the cluster was still guaranteed. Unfortunately, the area improvement obtained using this technique was limited to 5% and delays increased up to 30%.

In this work, the packing algorithm was left unchanged. Instead, improved switch patterns were used, spare cluster inputs were added to the cluster, and modifications to the router were made to support these architectural changes. Although these spare inputs contribute to additional area, they also improve routability and reduce channel width requirements. Overall, a net area reduction of up to 18% with *no degradation* to critical-path delay was obtained.

Architecture		Fully Populated Cluster Tile Area (Number of Minimum-Width Transistor Areas)						
LUT size	Cluster size	LUT+FF		Routing		LUT Input Mux		Total
4	6	990	(10.6%)	6050	(65.0%)	2267	(24.4%)	9307
5	6	1840	(16.4%)	6321	(56.2%)	3080	(27.4%)	11241
6	6	3496	(24.4%)	6713	(46.9%)	4109	(28.7%)	14318
7	6	6831	(34.8%)	7645	(39.0%)	5146	(26.2%)	19622
7	10	11358	(32.3%)	12022	(34.2%)	11765	(33.5%)	35145

Table 1: Breakdown of cluster tile area. The routing area is an arithmetic average required to route 20 MCNC circuits.
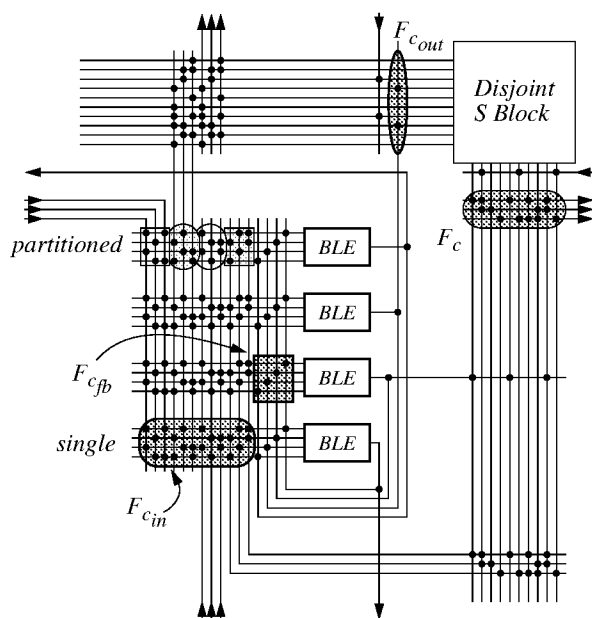
Figure 1: Details of the cluster tile architecture.

k	LUT size
N	cluster size
I	number of cluster inputs
I_{spare}	number of additional cluster inputs, used for routing only

Table 2: Cluster organization parameters.

$F_{c_{in}}$	cluster input to LUT input density
$F_{c_{fb}}$	LUT feedback to LUT input density
F_c	routing channel to cluster input density
$F_{c_{out}}$	cluster output to the routing channel density

Table 3: Switch density parameters.

1.2   Tradeoffs

Sparse clusters give the promise of reduced area, but one important tradeoff that must be made to realize this savings is increased routing time. In our experience, an approximate runtime increase of three to four times was observed. This increase may not be tolerable during early prototyping stages when design changes are frequent, but a less costly device could offset this inconvenience when an FPGA design shifts to volume production. Consequently, the premise of this paper is to evaluate the limits of area reduction that can be obtained using a high degree of CAD tool effort.

The remainder of this paper is organized as follows. In Section 2, the FPGA architecture is described along with the area and delay models. Section 3 discusses the experimental methodology and CAD tools used. Section 4 presents the results, and Section 5 concludes.

2.   FPGA ARCHITECTURE

This section describes assumptions made about the FPGA architecture and the area and delay models.

2.1   Architectural Model

The architecture used in this study is a symmetrical, island-style FPGA containing interconnected clusters. The basic FPGA tile

formed by a cluster and its routing channels is shown in Figure 1. This tile is drawn in a way to suggest a step-and-repeat layout that is possible, with wires on the left edge of one tile lining up with wires on the right edge of the adjacent tile.

One cluster contains N basic logic elements (BLEs), where one BLE contains a k-input LUT and a register. Each cluster has $I = \lfloor k(N+1)/2 \rfloor$ primary inputs which are used during packing [2]. As well, a cluster has I_{spare} additional cluster inputs which are reserved only for routing. These extra inputs are required to improve routability due to the restrictions imposed by sparse clusters. All of these cluster organization parameters are summarized in Table 2.

The cluster inputs are assumed to be logically equivalent, but they may connect to only some of the LUT inputs. The cluster input (and output) pins, which connect the cluster to the general routing, are evenly distributed on the four sides of the tile. Later in Section 4.3, we shall partition the cluster inputs into four groups based on which side they are placed.

2.2   Routing Architecture Details

Detailed routing architectural parameters were set to be the same as earlier studies [2, 4]. In the detailed routing architecture, 50% of the tracks are length-4 segments using tri-state buffers, the remaining tracks are length-4 segments using pass transistors, and clocks were assumed to be routed on a global resource. The *disjoint* switch (S) block was used, so signals entering the routing on track i must remain on that track number until the destination is reached. The number of i/o pads per cluster tile pitch was set to 5 for $N = 6$, and to 7 for $N = 10$.

The routing switch sizes (*i.e.*, buffer and pass transistor sizes) and wiring RC properties were computed assuming double minimum-spaced wiring and a fully-populated cluster tile size. For the $k = 4, N = 6$ architecture, the buffer was 6.1 times the minimum

size and the pass transistor was 12.2 times the minimum. The other architectures had larger tile sizes and used buffer sizes of 6.6, 7.6, 8.9, and 11.8. The pass transistor sizes were always chosen to be twice the corresponding buffer size.

Within a BLE, the LUT inputs are assumed to be logically equivalent and hence freely permutable. These inputs can select signals from two independent sources: either cluster inputs or feedback connections. The density of switches for these two regions, $F_{c_{in}}$ and $F_{c_{fb}}$, respectively, are independently controlled. These two parameters control the sparseness of switches inside the cluster.

For connections to outside the cluster, the inputs from and outputs to the general routing channels are selected using switch matrices with densities of F_c and $F_{c_{out}}$, respectively. The part of the general routing channel that connects to the cluster is commonly referred to as the connection block or C block.

The parameters controlling switch densities inside and outside of the cluster are summarized in Table 3.

Each BLE output directly drives a cluster output and a local feedback connection. The BLE outputs are assumed to be logically equivalent, allowing any function to be placed in any of the BLEs of the cluster. To achieve this output equivalence, every BLE is given the exact same input switch pattern.[1]

To improve routability, the routing tool takes advantage of the input and output equivalences just described. It may also replicate logic onto multiple BLEs in the same cluster, provided there are empty BLEs available.

2.3   Area Model

The area model used in this paper is the same buffer-sharing model used previously [2, 4], with a few minor changes described below. This model is based on the unit area of a minimum-width transistor (T), including the spacing to an adjacent transistor. As mentioned in [4], discussions with FPGA vendors have suggested that this, and not wiring, is the area-limiting factor.

All of the logic structures in the FPGA are modeled, including BLEs, the LUT input multiplexers, and the cluster routing, but not the padframe. For example, the area contribution of a pass transistor depends on the transistor width, and a buffer chain depends on the number of inverter stages as well as the required drive strength of each stage.

The drive strength requirement for a buffer is based on fan-out and is computed as follows. In general, it is assumed that a size B inverter in a buffer is sufficient to drive another inverter of size $4B$, or a total transistor gate width of $8B$. However, buffers driving the LUT input multiplexers, *i.e.*, the *cluster input buffers*, were sized differently. These buffers must drive a larger load created by the many levels of the LUT input multiplexer tree. This load is larger not only due to the depth of the tree, but also because diffusion is being driven. For these buffers, a size B was selected if the first level fan-out of the buffer [2] was loaded by a total diffusion width of $2B$, with the exception that drive strength was limited to be at least $7x$ and at most $25x$ minimum size. These approximations were made after examining HSPICE results [Ahmed and Wilton, *private communication*].

There were a few additional improvements made to the area

---

[1] An alternative architecture with different input switch patterns for each BLE can be built. Such an architecture would require a full permutation stage to reorder all of the BLE outputs to the cluster outputs and feedback connections. This could be done by fixing $F_{c_{fb}} = F_{c_{out}} = 1.0$, for example, or by using N additional $N-to-1$ multiplexers. We did not consider such an architecture here.

[2] Note that this fan-out can be significantly lower in a sparsely populated cluster, and this area savings is counted.

model described in [4]. The previous LUT area model was slightly pessimistic and used the largest buffer required for all LUT inputs. In addition, it was optimistic when estimating cluster input buffer load, and this produced slightly understated area results. In this paper, every LUT input buffer and every cluster input buffer was sized according to its unique load requirements, yielding a slightly smaller LUT area but a larger cluster tile area than previously reported.

One simplification made while employing this area model was that the routing switch sizes were chosen beforehand based on the tile size of a fully populated cluster. As a result, the switches are larger than required, since any area saved by employing a sparse cluster would surely shrink the tile size. If recomputed using the smaller sparse cluster tile size, the routing switch sizes, hence the overall tile size, would be reduced. However, this simplification merely implies that area savings reported in this paper are conservative.

2.4   Delay Model

The delay model used here is the same path-based, critical-path delay model used previously [2, 4]. Timing parameters for all delay results were obtained using $0.18\mu m$ TSMC process information and detailed HSPICE circuit models. The precise delays along each path are computed in one of two ways, as described below.

Routing delays from the cluster output buffer to the cluster input buffer are computed using the Elmore delay [6] of the RC-tree network. The delays inside a cluster, however, are modeled as constant worst-case delay times (either rise or fall) extracted from HSPICE simulation results of a fully populated cluster. For example, these constant delays measure propagation from a cluster input to a LUT input, or the delay through the LUT.

Delay results in this paper are very conservative and may be overstated for two principal reasons. First, the routing delay results are overestimated because they ignore the tile size shrink that was mentioned in Section 2.3. Consequently, the routing wirelength parasitics and switch sizes are larger than required. Second, due to reduced loading and smaller cluster input buffers, internal cluster delays might be reduced if this simulation was repeated for sparsely populated clusters.

For these two reasons, the delay model used tends to produce pessimistic results for both components of delay: internal cluster routing and general purpose routing. Since internal cluster routing alone accounts for 35% of the critical-path delay on average [14], any savings from either component would lead to a measurable overall delay reduction.

3.   METHODOLOGY

In general, the experimental methodology from [2, 4] was used. Benchmark circuits were optimized using SIS [13], mapped into LUTs using FlowMap and FlowPack [5], packed into clusters using T-VPACK [9], and placed using VPR [3, 4, 10] onto the smallest square FPGA that fits the circuit. In all experiments, the same packing and placement was used for each unique combination of circuit, LUT size and cluster size.

Below, the remainder of the CAD process is described, beginning with details about the routing stage, then a description of the router enhancements, and lastly a note on CAD tool parameter selection.

3.1   Routing Step

The last step of the CAD flow involves routing a placed netlist in the detailed routing architecture. The routing tool used here is based on a modified version of VPR 4.30 which was tailored specif-

ically for sparse clusters. This version of VPR includes the latest timing-driven packing and placement enhancements [9, 10].

During routing, the minimum channel width required to route, W_{min}, was found using a binary search. Afterwards, a final low-stress routing was done with $W = 1.3 \cdot W_{min}$ tracks to compute area and delay statistics. This procedure models the way FPGAs are actually used; designers are seldom comfortable working on the edge of capacity or routability.

The final low-stress routing actually failed in 34 out of 3980 (0.9%) circuit/architecture combinations, usually due to slow convergence or switch pattern interference.[3] To resolve this, one, two, then three additional tracks were added to the channel. This strategy was sufficient to route all but four of the troublesome cases — the three underlying architectures for these cases were deemed unroutable, so they were abandoned from further consideration in this paper.

Also, if the binary search was unable to find a reasonable minimum channel width ($W_{min} \leq 240$) for any of the circuits, the architecture was deemed unroutable and abandoned. Consequently, every architectural result presented in this paper was obtained by routing all of the benchmark circuits.

All area and delay results are averages obtained from placing and routing the 20 largest MCNC benchmark circuits [7]. Area is computed as the geometric average of the *active FPGA area*, which is defined below. The geometric average ensures that the circuits are all weighted equally, independent of the size of the circuit. Delay results are also the geometric average of the critical-path delay for each benchmark circuit.

Active FPGA area is the area, in units of minimum-width transistor areas, of one cluster tile (including its routing) times the number of clusters actually used by the benchmark circuit. This measurement was used in [1, 2] to better distinguish packing efficiency. We have chosen to use the active FPGA area metric here to be consistent with those results.[4]

3.2   CAD Tool Enhancements

Originally VPR routed only to cluster input pins because fully-connected clusters could guarantee the routability of cluster inputs and feedback connections. Extensive modifications to VPR were necessary to route sparsely populated clusters. For example, the routing graph, timing graph, and netlist structures had to be altered to accommodate the cluster feedback nets and the location of every BLE sink. As well, other changes were necessary to permit nets to enter a cluster more than once to improve routability.

The switch pattern generator from [8] was integrated into VPR to create the switch patterns for the LUT input multiplexers. This generator first distributes switches to balance the fan-in and fan-out of each wire, usually in a random pattern. A greedy improvement strategy is then followed which roughly maximizes the number of distinct output wires reached by every pair of input wires. To accomplish this, switches are randomly selected, first in pairs, then singly, and moved only if the fan-in/-out constraints are kept and the aforementioned cost improves. Using this technique, the switch patterns within a cluster are individually well-designed.

Other switch patterns in the routing fabric, namely the cluster input and output patterns, use the original VPR switch placement generators. Additionally, we *have not* attempted to optimize the cascading of the the cluster input multiplexers and LUT input multiplexers, except as noted below in Section 4.3. This extension to the work is nontrivial and left for future investigation.

Cluster feedback connections are also sparsely populated, and this may cause some problems during routing. In particular, there may be too few switches to satisfy all possible feedback connection patterns, so feedback signals are also permitted to leave the cluster and re-enter through the cluster inputs. This may cause speed degradation, or some netlists may become unroutable because all cluster inputs are used.

There is no immediate solution for the speed degradation problem, but we address the routability problem by assuming there are *spare cluster inputs* in the architecture. These spare inputs improve routability by providing the router with more choices [8]. The number of spare inputs given, I_{spare}, is specified prior to routing as a fixed part of the architecture. For convenience, the packing tool adds these as part of the netlist and the router automatically uses them.

The effectiveness of the modified VPR router was validated against the original version of VPR. Both routers obtained similar delays, channel widths and area results for fully populated clusters using a variety of cluster and LUT sizes.

3.3   Tool Parameters

In general, the packing, placement and routing tools were run in timing-driven mode using their default parameters. Some non-standard command line switches were used for routing — these are shown in Table 4 and described in further detail below.

The number of router iterations had to be increased beyond the default value of 30, partly because sparsely populated clusters require additional routing effort. As well, large variations were observed in delay results because the router parameter that increases the cost of nets sharing wires between iterations (pres_fac_mult) was too high.

The impact of reducing this parameter on runtime and average critical-path delay of the low-stress route can be seen in Figure 2. As well, the range of average delay values (across all benchmarks) for each architecture is shown using error bars. This wide range made it difficult to distinguish architectures with low delay from those with higher delay. Clearly, increased routing effort was required to reduce the delay variation, but we feel this was time well-spent. Without this effort, we would be unable to conclusively compare the delay results of the different architectures.

For this experiment, the maximum number of router iterations was set to 300. On average, however, the number of iterations used increases from 23 to 160 in a manner that very closely follows the increase in runtime. The router values shown in Table 4 were chosen in reference to these results.

4.   RESULTS

This section gives the area and delay results from placing and routing 20 MCNC benchmark circuits. In all cases, only the geometric average is used for FPGA area and critical-path delay. Initial experiments determined the best routing parameters, then these parameters were used to evaluate the area and delay of sparse cluster architectures.

4.1   Key to Curve Labels

In the following graphs, each curve represents a family of architectures parameterized along the x-axis. Each curve label describes the specific architecture parameters in the following order:

$$k \quad N \quad I_{spare} \quad F_{c_{in}} \quad F_{c_{fb}}$$

These parameters are fully described in Tables 2 and 3. Where the value of a parameter is given as 'X', that simply means the parameter is being varied along the x-axis.

---

[3]Of the failed combinations, 20 of them had $I_{spare} = 0$ and the remainder had $F_{c_{in}} \leq 0.25$.

[4]Note that the results in [2] use a different process technology.

Tool	Additional Parameters
T-VPACK	*default*
VPR placement	*default*
VPR binary search	`-pres_fac_mult 1.3 -max_router_iterations 250`
VPR final route	`-pres_fac_mult 1.05 -max_router_iterations 300`

Table 4: CAD tools and non-default parameters used.
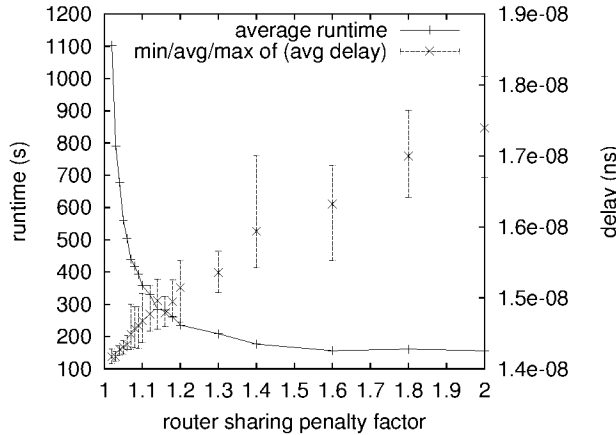
Figure 2: Variation in average critical path delay is shown as a function of the router sharing penalty factor. To reduce the variation (and the delay), longer runtime is required.

Figure 3: F_c impact on channel width.

4.2   Routing Architecture Selection

To explore the sparse population of switches inside the cluster, it is first necessary to establish a good routing architecture outside of the cluster. Hence, the best values for F_c and $F_{c_{out}}$ need to be selected beforehand. We chose to find the best switch density that would give minimum area rather than delay. To generate the results for this expediently, the number of router iterations was limited to 75, but all other parameters were left at their default values.

4.2.1   Selecting F_c for minimum area

The density of switches connecting channel wires to cluster inputs in the C blocks is called F_c. We wish to determine the value of F_c that would result in a minimum-area FPGA.

The choice of F_c depends on the effectiveness of the CAD tools and the size of the C block, determined by the channel width, W, and the number of cluster inputs, I. It has been our experience that I is the most important factor influencing the choice of F_c.

Routing experiments were done for $k = 7$ architectures, varying N from 2 to 9. This large LUT size was chosen because we are mostly interested in the effects of having a large number of cluster inputs. Both full (100%) and sparse (50%) population levels inside the cluster were tried. The 50% density was chosen because this was almost always routable without adding spare inputs, hence $I_{spare} = 0$ here.

The average low-stress channel width required to route the benchmark circuits, W, is presented in Figure 3 for a variety of F_c values. Only three cluster sizes are illustrated, but the other results are similar. From these results, it can be seen that choosing $F_c > 0.4$ has little impact on channel width. Although not shown, this is particularly true for $N > 3$.

Interestingly, the channel width results are very similar for both sparse and fully-populated clusters. Sparse clusters typically re-

quired only 2 to 4 more tracks than the corresponding fully populated ones. Hence, the sparse architecture is still quite routable at the 50% population level.
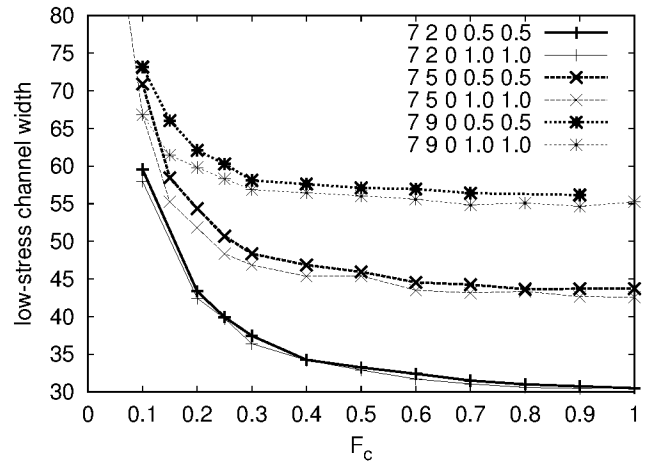
Although channel width is not hindered by a large value of F_c, having more switches than necessary will contribute to an area increase. Figure 4 also shows the active FPGA area versus F_c for cluster sizes 2 and 9.[5] Again, similar results were obtained for cluster sizes 3 through 8.

One unexpected area result is that the 50% sparse cluster near $F_c = 1.0$ always uses *fewer* transistors than the minimum-area fully-populated cluster. This can be seen in Figure 4 where point B is lower than A, and D is lower than C. This trend holds for the other cluster sizes as well. Hence, it is better to sparsely populate the clusters than the general routing, a non-intuitive result. One reasonable explanation for this is there are about twice as many LUT input multiplexers as cluster input multiplexers, even though the cluster input multiplexers can easily have twice as many inputs (based on the channel width).

Another result shown in Figure 4 is a significantly larger area reduction for $N = 9$ than $N = 2$. The reduction is so large that the $N = 9$ architecture goes from using more area (curve C) than the corresponding $N = 2$ architecture (curve A) to using less area (curves D and B). This result shows how sparse clusters can shift the optimum design point towards larger clusters. For example, in this $k = 7$ architecture, the fully-populated cluster should contain between 4 and 6 LUTs to be area-efficient. However, the 50% sparsely-populated cluster should contain between 4 and 9 LUTs. Further investigation of different cluster sizes is left as future work.

The values of F_c producing the lowest area for each cluster size, *i.e.*, for each value of I, are shown in Figure 5. It is remarkable that

---

[5]Notice that the sparse $F_c = 1.0$ result is missing for $N = 9$ in Figures 3 and 4 because VPR was unable to route the *clma* circuit under low-stress conditions due to slow convergence.
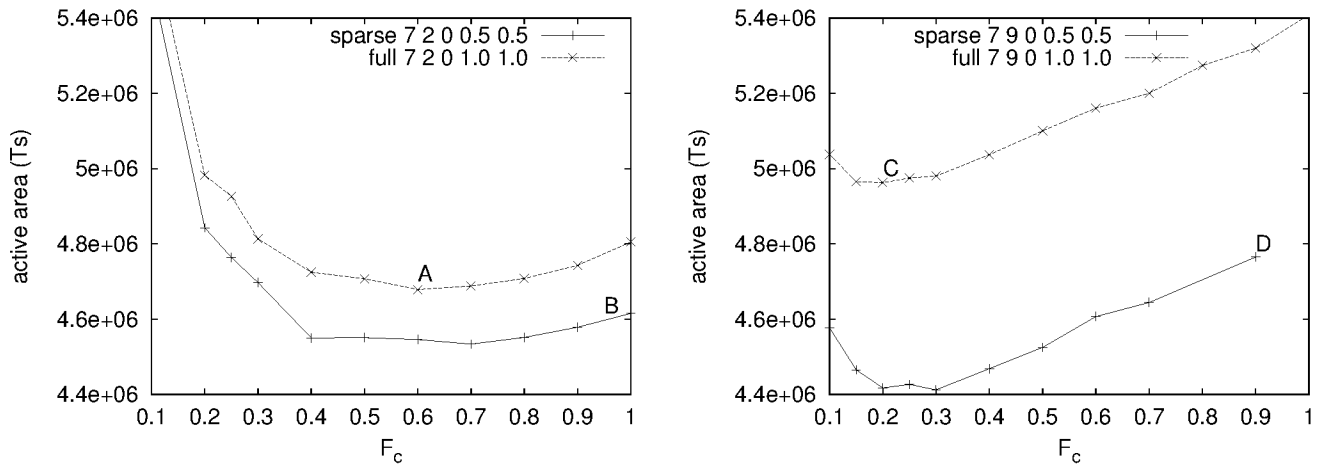
Figure 4: F_c **impact on area for cluster sizes of 2 and 9. Intermediate cluster sizes gave similar results.**

the sparse and fully populated cluster results are so similar. This can be partly attributed to the relative flatness near the minimum area. For $N = 9$, varying F_c from 0.1 to 0.5 causes less than 5% change in area. Hence, precise F_c selection is not critical, provided it is large enough to be routable, yet not wastefully large.

For the remainder of the results in this paper, it was determined that a fixed value of F_c would not significantly hinder area results. Rather than using the minimum-area F_c values from Figure 5, we felt that having a few more switches in the routing (by having a slightly larger F_c) would be helpful as clusters were made even more sparse (internally). This is especially important because no effort was made to tune the two switch patterns together and we wished to avoid possible interference patterns. Hence, we chose to set $F_c = 0.5$ for the $N = 6$ architectures and $F_c = 0.366$ for the $k = 7, N = 10$ architecture. These particular values were chosen because they were used in previous work [2, 4] and this gives us the most comparable results.

4.2.2   Selecting $F_{c_{out}}$

Previous experiments have shown that $F_{c_{out}} = 1/N$ is adequate for routing in fully populated architectures [4]. Considering the similarity of the F_c area results between sparse and fully populated architectures, it was decided that modifying $F_{c_{out}}$ would have insignificant impact in a sparsely connected architecture. Hence, $F_{c_{out}} = 1/N$ was used for all results.

4.3   Partitioning of Cluster Inputs

Additional net delay can be caused by sparsely populated clusters because some LUT inputs may not be reachable from particular sides of the cluster. For example, consider the case when some LUT input connections have already been formed, and the last remaining input signal is being made. A lack of switches inside the cluster may cause that net to enter the cluster from a more distant side. The result is increased delay.

We investigated this problem by trying a *single* switch matrix for all cluster inputs, and one which was *partitioned* into four smaller switch matrices, one for each input side. The partitioned matrix addresses the above problem by ensuring that all of the cluster inputs from any particular side can reach all of the LUT inputs. It also has a weakness though: these smaller switch matrices are not carefully designed to couple together well. Each partitioned matrix is derived from the same basic switch pattern, but each has its own
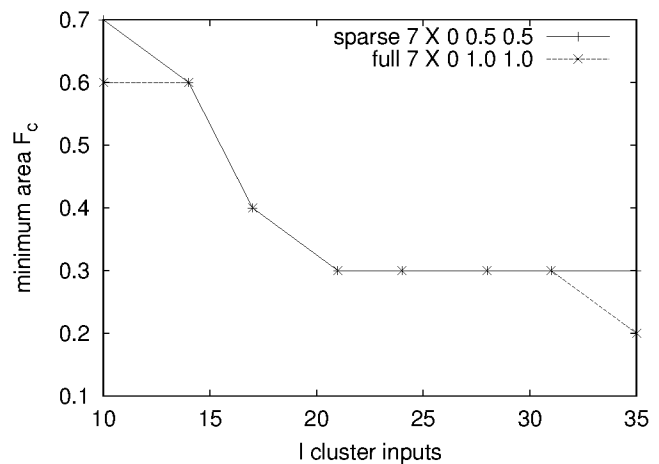
Figure 5: Best F_c **corresponding to minimum area as a function of** I **cluster inputs.**

permutation of the rows (or outputs) to balance the fan-in of the LUT inputs. These matrices, but not the permutation pattern, are illustrated in Figure 1.

Both switch designs were routed in a $k = 7, N = 10, F_{c_{in}} = F_{c_{fb}} = 0.43$ architecture. Both designs required identical transistor area, and the *partitioned* matrix was only about 1% faster. Although this is not significantly faster, it was used for subsequent results in this paper since it may help with some pathological cases.

4.4   Sparse Cluster Area Results

The primary motivation for depopulating clusters is to reduce the area, and subsequently the cost, of an FPGA. In Section 4.2, it was determined that simply depopulating the cluster to 50% is more effective at reducing area than choosing the proper value of F_c. In this section, further depopulation of the cluster is explored.

To reduce the number of routing experiments, it was decided to fix the cluster size to $N = 6$ and vary the LUT sizes from 4 through 7. That particular cluster size was selected because it generated near-minimum area and area-delay results for fully populated clusters with all of these LUT sizes. The larger LUT sizes are especially interesting because they require larger input switch matrices, hence
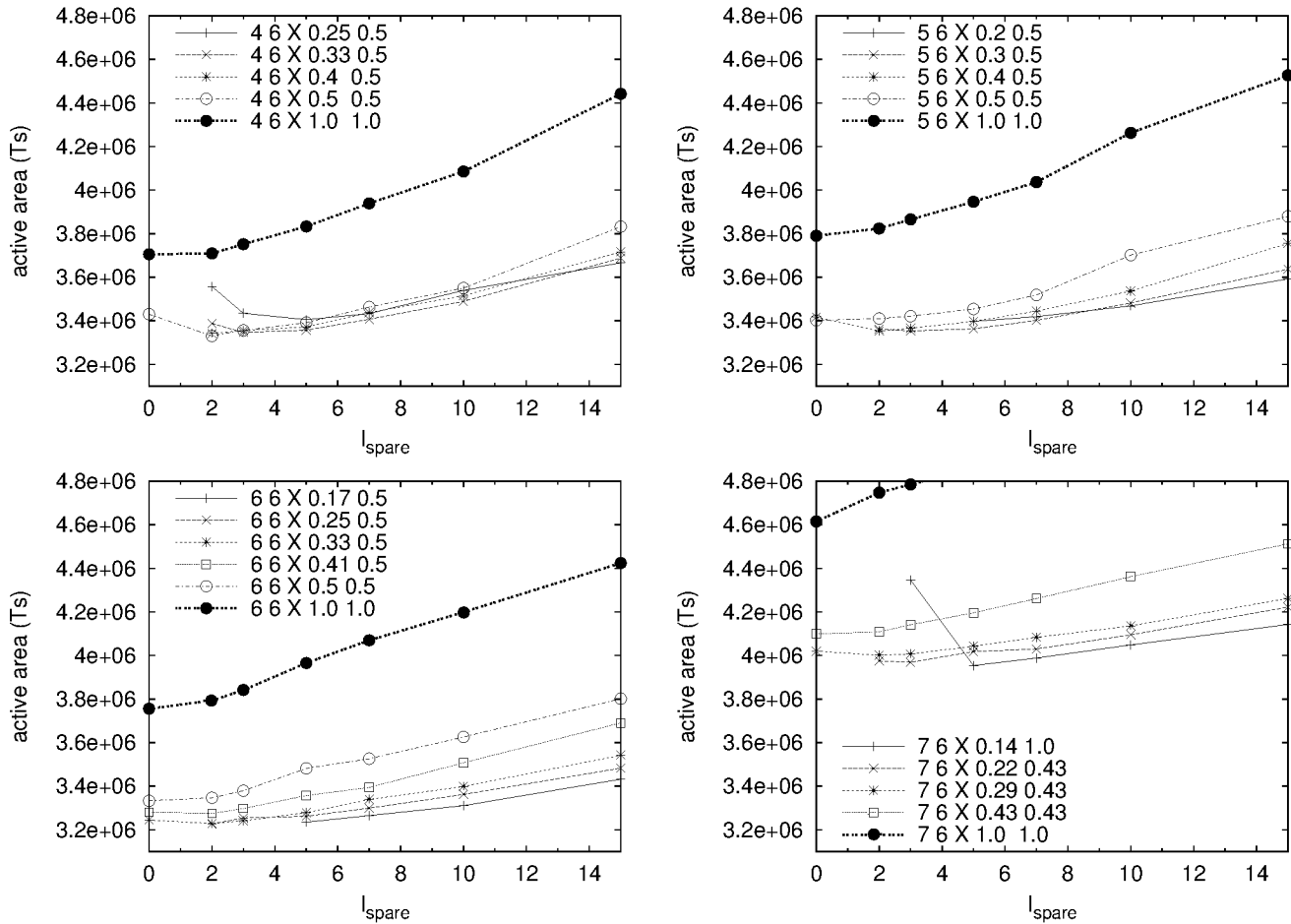
Figure 6: Active FPGA area of fully and sparsely populated clusters.

offering more potential for depopulation. One additional architecture with $k = 7, N = 10$ was chosen to study an even larger number of inputs entering the cluster.

A number of preliminary routing experiments were run with a wide range of values for $F_{c_{in}}$ and $F_{c_{fb}}$. From these results, which are not shown here, it was confirmed that $F_{c_{fb}}$ has less influence on area. As $F_{c_{fb}}$ was reduced below 50%, a number of circuits would no longer route. It was determined that $F_{c_{fb}}$ of 50% (or $3/7 = 43\%$ for $k = 7$) was as low a value as could be tolerated. Similar preliminary sweeps indicated that $F_{c_{in}} \geq 0.5$ was nearly always routable, so area reduction should concentrate on more sparse values.

The area results from routing the four LUT sizes are shown in Figure 6. In these graphs, each curve represents the geometric average of active FPGA area for a fixed value of $F_{c_{in}}$. The number of spare inputs is varied along the x-axis. The sparse cluster results should be compared against the bold curve representing the fully-populated cluster area.

The most apparent trend in these curves is a gentle dip, then a general upward climb in area as I_{spare} is increased. The upward trend is an expected result, since the spare inputs will require additional cluster input multiplexers. The dip is caused by a rapid initial decline in average channel width, which then gradually reaches a 5% to 20% reduction (10% is typical).

A number of data points are missing in Figure 6, specifically for small I_{spare} values. This is because one or more benchmark circuits

could not be routed on the architecture. Hence, although they contribute to area reduction in only a few cases, it is *essential* to have these spare inputs to make sparse clusters routable. Typically, between two to five spare inputs are required to make the architecture routable and attain the lowest area.

The lowest-area architectures from Figure 6 are summarized in Table 5. As well, the large $N = 10$ cluster architecture is included. With these architectures, a 10 to 18% area savings is achieved. As mentioned earlier, between two and five spare inputs is sufficient to achieve most of this savings, which is surprising since this only about one spare input per side.

A breakdown of the cluster tile area is given Table 6. For 4-input LUTs, there was a slight decrease in routing area because the spare inputs helped reduce average channel width. The 5- and 6-input LUTs cases did not achieve the same benefit because the spare inputs contributed more to area than the amount saved by the slight channel width reduction. The two 7-LUT architectures had an increase in routing area due to the spare inputs and a channel width increase. However, the sparse switch populations produced a net area savings of 14% and 18%, with the larger cluster benefitting more. With respect to the entire tile, depopulating the clusters was very effective at reducing the relative LUT input multiplexer size from the 24–33% range down to 12–18%.

One very interesting result from this data is that a sparse cluster of six 6-input LUTs is slightly more area-efficient (3%) than six

Architecture				Best Sparse Parameters			Channel Width (arith. avg.)		Active FPGA Area ($\times 10^6$ Ts)		
k	N	I	F_c	I_{spare}	$F_{c_{in}}$	$F_{c_{fb}}$	Fully Populated	Best Sparse	Fully Populated	Best Sparse	Savings
4	6	14	0.5	2	0.5	0.5	47.9	45.9	3.71	3.33	10.1%
5	6	17	0.5	2	0.4	0.5	46.4	45.6	3.79	3.35	11.5%
6	6	21	0.5	2	0.33	0.5	44.3	43.5	3.76	3.23	14.0%
7	6	24	0.5	5	0.143	0.43	43.8	44.6	4.62	3.95	14.3%
7	10	38	0.366	10	0.143	0.43	53.7	55.1	4.96	4.03	18.8%

Table 5: Active FPGA area savings obtained by depopulating switches inside the cluster.

Architecture		Tile Area (Number of Minimum-Width Transistor Areas)								
		Fully Populated Cluster					Best-Area Sparse Cluster			
k	N	Total	LUT+FF	Routing	LUT Input Mux		Total	LUT+FF	Routing	LUT Input Mux
4	6	9307	990	6050	2267	(24.4%)	8380	990	5960	1430 (17.1%)
5	6	11241	1840	6321	3080	(27.4%)	9964	1840	6371	1753 (17.6%)
6	6	14318	3496	6713	4109	(28.7%)	12343	3496	6732	2115 (17.1%)
7	6	19622	6831	7645	5146	(26.2%)	16879	6831	8120	1928 (11.4%)
7	10	35145	11358	12022	11765	(33.5%)	28646	11358	12990	4298 (15.0%)

Table 6: Breakdown of cluster tile area. The routing area is an arithmetic average for all circuits.
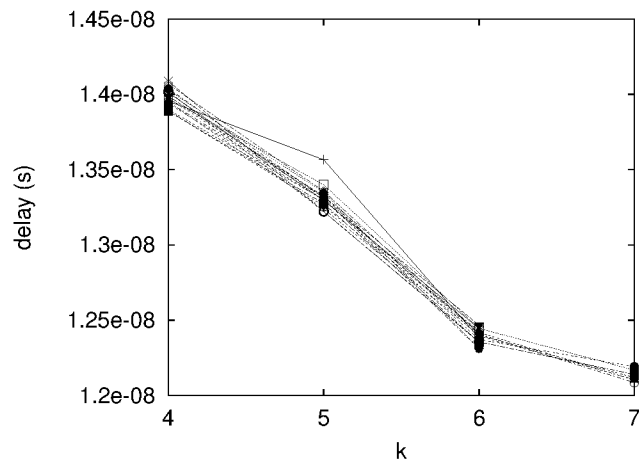
Figure 7: Delay decreases with LUT size.

Figure 8: Delay is not influenced by $F_{c_{in}}$. Similar results indicate it is not influenced by I_{spare} or $F_{c_{fb}}$.

4-LUTs in a sparse cluster. This is a departure from previous work which has consistently shown that 4-LUTs achieve lower area, albeit in fully populated clusters. The reason for this difference is simple: larger LUTs provide more opportunity for depopulation. This concept is supported by previous work which has shown that sparse crossbars with more outputs require fewer switches for the same level of routability [8].

4.5   Sparse Cluster Delay Results

As mentioned earlier, reduced switch densities may cause an increase in delay due to an increase in bends or wire use to achieve routability. Although delay may decrease for other reasons such as reduced loading, we chose to be conservative and ignore these possible benefits.

The curves in Figure 7 show the impact that varying the LUT size has on delay for a few of the $N = 6$ architectures. The curve labels identifying the architectures have been omitted for clarity, since only trends need to be observed. The important thing to notice is that, for all architectures, delay goes down as k increases.

Similarly, Figure 8 shows the change in delay as the switch density $F_{c_{in}}$ is varied. It is apparent in the graph that curves of the same LUT size are all grouped together. In particular, the 4- and 5-LUT data is easily distinguished from the 6- and 7-LUT data. The flatness of all of these curves illustrates how little impact $F_{c_{in}}$ has on delay.

Analysis of delay while varying I_{spare} or $F_{c_{fb}}$ shows the same result: delay is independent of these parameters. Even though sparse clusters present a challenge to the router and remove many choices, and even though some feedback connections must leave the cluster and re-enter through the general-purpose routing, the router still has enough freedom to ensure that nets on the critical path remain on the fastest paths to the critical sinks.

4.6   Sparse Cluster Area-Delay Product

The previous two sections presented results indicating the 6-LUT had the lowest area and the 7-LUT had the lowest delay. When the
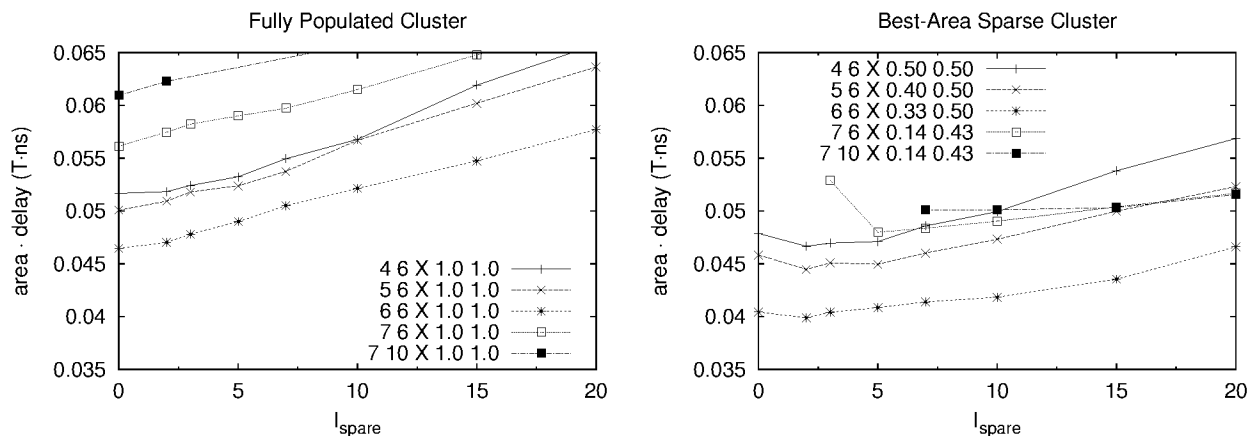
Figure 9: Area-delay product results for fully-populated and best-area sparse architectures.

		Average Runtime (seconds)		Average # Routing Iterations		
		VPR 4.30	Modified VPR	VPR 4.30	Modified VPR	
k	N	Fully Populated	Best Sparse	Fully Populated	Best Sparse	
4	6	70	153	150	84	86	86
5	6	72	183	205	93	91	93
6	6	57	177	178	84	86	88
7	6	53	188	350	88	83	109
7	10	43	177	275	96	94	116

Table 7: Average runtime and number of routing iterations for the final low-stress route (arithmetic averages of 20 benchmarks). Runtimes were collected on an 866MHz Pentium III computer with 512MB of SDRAM.

area and delay results are combined in the form of an area-delay product, the 6-LUT emerges as the superior logic block choice. This metric is important because it indicates when the best trade-off is being made between using an additional amount of area for a similar relative gain in clock rate (or vice versa). For example, it is directly useful in FPGA-based computation because the computation rate is a product of both the clock rate and parallelism.

The best sparse area-delay product organizations are compared to their fully-populated versions in Figure 9. The area-delay product improves for every LUT size due to the area reduction. The overall best sparse architecture containing 6-LUTs is about 14% more efficient than one containing 4-LUTs, and about 22% more efficient than the traditional fully-populated 4-LUT cluster.

4.7   Routing Runtime with Sparse Clusters

The removal of switches inside the cluster also removes the routability guarantee of the cluster. Consequently, the router must pay attention to all of the wires and switches within the cluster, so it is expected that additional runtime effort is required to complete the route.

The average runtime and average number of iterations required for routing the different architectures are shown in Table 7. Results are presented for fully populated clusters to compare the original VPR 4.30 to the modified one. As well, the modified VPR can be compared against itself to study the additional impact of routing the best-area sparse clusters.

Generally, the modified VPR currently runs about three to four times slower than the original version when fully populated clusters

are used. Even though runtime has increased, the number of router iterations used is practically unchanged. The main reason for the slowdown comes from the increased number of wires and switches in the architecture that must be examined with each iteration: all cluster inputs now have connections to many LUT inputs, and nets are allowed to enter a cluster more than once. This causes the router to evaluate many more routing paths before making a decision.

It is worthwhile to note that having larger LUT sizes and cluster sizes reduces the amount of work that VPR 4.30 must do, so runtime decreases. This benefit was not realized in the modified VPR because the amount of wiring inside the cluster also increases, keeping runtime relatively flat.

The additional runtime needed to route the best-area sparse architectures is also shown in Table 7. For $k = 4, 5, 6$ the runtime and the number of iterations is similar, for $k = 7$ runtime nearly doubled and the number of iterations increased by 25–30%.[6] This increase in the average is caused by a large increase in four of the normally difficult-to-route circuits. The need for more router iterations indicates these architectures are barely routable, probably because $F_{c_{in}}$ is so low, even though these circuits are being routed using the low-stress channel width.

Increasing routability by increasing I_{spare} to 15 for the $k = 7, N = 10$ architecture reduced runtime to 210 seconds and 97 iterations. Hence, the amount of area savings can also be balanced against the runtime effort.

---

[6]The amount of searching done in each iteration may increase as the search space expands, so each iteration's runtime may increase.

5. CONCLUSIONS

This work has studied the area and delay impact of sparsely populating the internal cluster connections in a clustered architecture. At the expense of three to four times the compute time, an area savings of 10 to over 14% was realized by sparsely populating the cluster internals of 4-, 5-, 6-, and 7-input LUT architectures containing 6 LUTs per cluster. A larger cluster size of ten 7-LUTs obtained an 18% area savings. It was also observed that the additional router effort and reduced routing flexibility did not degrade critical-path delay.

A fixed number of spare inputs were added to each cluster. These inputs are used only by routing, and are not used or required for packing. By adding up to 15 spare inputs, the channel width decreased by about 10% in most architectures, whether full or sparsely populated. Although sparse clusters on their own impose a small increase in channel width, the spare inputs reduce the channel width, resulting in a small, net savings.

The channel width reduction typically produced a net savings in routing area alone when up to seven spare inputs were added, but resulted in a net increase thereafter. Of course, the cluster area (excluding the routing) always increased with the addition of spare inputs. However, this area increased at a slower rate in more sparsely populated clusters, as expected. When added to the routing area, most architectures became less efficient after more than five spare inputs were employed.

The increase in routability and decreases in channel width and area indicate that it is best to force the packing algorithm to leave a few spare inputs (two or three) for the router.

One interesting outcome of this work is that, contrary to popular belief, it is more area-efficient to depopulate only the LUT input multiplexers than it is to depopulate only the cluster input multiplexers (*i.e.*, the C blocks) in the general routing. The reason for this is that, due to input sharing in a cluster, there are about twice as many LUT input multiplexers than cluster input multiplexers. Of course, depopulating both regions provides even more savings.

Another interesting observation is that 6-LUTs become more area efficient than 4-LUTs when sparse clusters are employed. This was entirely attributable to the more sparse pattern that could be used in the 6-LUT case.

The area and delay results in this paper used conservative estimates and ignored secondary effects which would improve results further. In particular, the tile size and the subsequent routing switch size reduction from sparse cluster use should lead to additional area and delay reduction. Delay improvement may also come from reduced loading inside the cluster and by generally using larger cluster sizes, which are more area-efficient when using sparse clusters.

It is reasonable to expect that larger cluster sizes may produce an even larger area savings due to the large amount of area concentrated in the LUT input multiplexers.

Future work in this area will include effort to jointly design the LUT input switch matrices with the cluster input multiplexers to avoid switch pattern interference. Additional constraints such as carry chains or other local routing may impact sparse cluster design and should be evaluated. A wider variety of cluster sizes, particularly the effectiveness of large clusters, should also be explored. The area savings from sparely populated clusters will reduce tile size, but the subsequent area and delay reduction from using smaller routing switches should also be quantified. The delay improvements arising from reduced loading and larger cluster sizes should be investigated. Also, efforts should be made to improve the runtime of the router while still retaining the area savings.

An interesting extension of this work would involve tighter coupling with the packing stage. For example, under special circumstances, it may be reasonable to have the packing tool use the spare inputs reserved for routing. Before doing this, it could first do a routability test to verify whether the potential cluster of logic blocks is routable. Since this shouldn't be a common case, it can be done with reasonable CPU effort. This may increase the usefulness of the FPGA architecture for subcircuits which have wide fan-in (or poor input sharing), such as finite state machines.

6. ACKNOWLEDGEMENTS

7. REFERENCES

[1] E. Ahmed. The effect of logic block granularity on deep-submicron FPGA performance and density. Master's thesis, Department of Electrical and Computer Engineering, University of Toronto, 2001.

[2] E. Ahmed and J. Rose. The effect of LUT and cluster size on deep-submicron FPGA performance and density. In *ACM/SIGDA Int. Symp. on FPGAs*, pages 3–12, 2000.

[3] V. Betz and J. Rose. VPR: A new packing, placement and routing tool for FPGA research. In *Field-Programmable Logic*, pages 213–222, 1997.

[4] V. Betz, J. Rose, and A. Marquardt. *Architecture and CAD for Deep-Submicron FPGAs*. Kluwer Academic Publishers, Boston, 1999.

[5] J. Cong and Y. Ding. FlowMap: An optimal technology mapping algorithm for delay optimization in lookup-table based FPGA designs. *IEEE Transactions on Computer-Aided Design*, pages 1–12, January 1994.

[6] W. Elmore. The transient response of damped linear networks with particular regard to wideband amplifiers. *Journal of Applied Physics*, pages 55–63, January 1948.

[7] C. B. Laboratory. *LGSynth93 suite*. http://www.cbl.ncsu.edu/www/.

[8] G. Lemieux, P. Leventis, and D. Lewis. Generating highly-routable sparse crossbars for PLDs. In *ACM/SIGDA Int. Symp. on FPGAs*, pages 155–164, Monterey, CA, February 2000.

[9] A. Marquardt, V. Betz, and J. Rose. Using cluster-based logic blocks and timing-driven packing to improve FPGA speed and density. In *ACM/SIGDA Int. Symp. on FPGAs*, pages 37–46, 1999.

[10] A. Marquardt, V. Betz, and J. Rose. Timing-driven placement for FPGAs. In *ACM/SIGDA Int. Symp. on FPGAs*, pages 203–213, 2000.

[11] M. I. Masud. FPGA routing structures: A novel switch block and depopulated interconnect matrix architectures. Master's thesis, Department of Electrical and Computer Engineering, University of British Columbia, December 1999.

[12] J. Rose and S. Brown. Flexibility of interconnection structures in field-programmable gate arrays. *IEEE Journal of Solid State Circuits*, 26(3):277–282, March 1991.

[13] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. Sangiovanni-Vincentelli. SIS: A system for sequential circuit analysis. Technical Report UCB/ERL M92/41, University of California, Berkeley, May 1992.

[14] M. Sheng and J. Rose. Mixing buffers and pass transistors in FPGA routing architectures. In *ACM/SIGDA Int. Symp. on FPGAs*, 2001.

VLSI Layout of Benes Networks

Paul Manuel
Department of Information Science
Kuwait University, Kuwait

Kalim Qureshi
Department of Mathematics and Computer
Science, Kuwait University, Kuwait

Albert William
Department of Mathematics, Loyola College,
Chennai 600 034

Albert Muthumalai
Department of Mathematics, Loyola College,
Chennai 600 034

Abstract − The Benes network consists of back-to-back butterflies. There exist a number of topological representations that are used to describe butterfly - like architectures. We identify a new topological representation of Benes network. The significance of this representation is demonstrated by solving two problems, one related to VLSI layout and the other related to robotics. An important VLSI layout network problem is to produce the smallest possible grid area for realizing a given network. We propose an elegant VLSI layout of r-dimensional Benes networks using this representation. The area of this layout is $O(2^{2r})$ whereas the lower bound for the area of the VLSI layout of Benes networks is $O(2^{2r})$. This lower bound is estimated by applying Thompson result.

Keywords: Butterfly network, Benes network, VLSI layout problem

1.0 Introduction and Background

A multistage network consists of a series of switch stages and interconnection patterns, which allows N inputs to be connected to N outputs. A multistage network uses dynamic interconnection to allow communication paths to be established as needed for the transfer of information between I/O nodes. In massively parallel computing, interconnection networks remain to be one of the most critical components. Multistage interconnection networks (MINs) have long been used as the communication network for parallel computers. The main advantages of MINs are their high bandwidth O(N), low diameter O(log N), and constant degree switches. Multistage networks have been used in commercial machines, such as the BBN, CM 5 and Meiko [8]. The butterfly and Benes networks are important multistage interconnection networks, which possess attractive topologies for communication networks [10]. They have been used in parallel computing systems such as IBM SP1/SP2, MIT Transit Project, and NEC Cenju-3, and used as well in the internal structures of optical couplers, e.g., star couplers [10, 13].

We represent networks as undirected graphs whose nodes represent processors and whose edges represent inter-processor communication links. An *embedding* of undirected graph, G, in another, H, comprises a one-to-one *assignment* of the nodes in G to nodes in *H*, plus a *routing* of each edge of G within H that is, an assignment of a path in H connecting the images of the endpoints of each edge in G.

The set of nodes V of an r-dimensional butterfly correspond to pairs $[w, i]$, where i is the dimension or level of a node ($0 \leq i \leq r$) and w is an r-bit binary number that denotes the row of the node. Two nodes $< w, i >$ and $< w', i' >$ are linked by an edge if and only if $i' = i + 1$ and either:

1. w and w' are identical, or
2. w and w' differ in precisely the ith bit.

The edges in the network are undirected. An r-dimensional butterfly is denoted by BF(r).

An *r-dimensional Benes network* has 2r+1 levels, each level with 2^r nodes. The level zero to level r vertices in the network form an r-dimensional butterfly. The middle level of the Benes network is shared by these butterflies [9]. As butterfly is known for FFT, Benes is known for permutation routing (*rearrangeable* network). An r-dimensional Benes is denoted by B(r). Figure 1 shows a B(4) network. All the results in this paper are discussed only for Benes since the results corresponding to butterfly can be derived as a particular case.

One of the important parameters used to evaluate parallel architectures is symmetry. Other considerations are: bisection width, wire length (layout aspects), existence of optimal communication techniques such as routing and broadcasting, node disjoint paths (fault tolerance), and recursive decomposition (or scalability). Symmetry is helpful for solving issues related to VLSI design. A VLSI layout for a network is characterized as an embedding within a two-dimensional grid, which assigns nodes of a graph G to points in the grid together with an (incidence preserving) assignment of the edges of G to paths in the grid. The paths of the layout are restricted to follow along grid tracks and are not allowed to overlap for any distance (although a vertical path segment may cross a horizontal path segment). If they change direction at this point, it is called a *knock-knee* [5]. In addition, the paths may not cross nodes to which they are not adjacent [2].

1.1   Mathematical Definition of a VLSI Layout

Following [1], a VLSI layout Ł(α, ρ) of an N-node graph G(V, E) in an *m* x *n* grid M, where $N \leq mn$, is an embedding (α, ρ) of G into M[*m, n*] where α is a one – one mapping from V(G) into V(M) and ρ = {P(u, v) / (u, v) is an edge of E(G) and P(u, v) is a path of M joining α(u) and α(v)}. The routing paths of ρ collectively satisfy the following conditions:

- Distinct routing paths in the grid are edge-disjoint, so that the embedding that embodies a layout has unit congestion in the grid.

- A routing path P(u, v) traverses over no image node α(w) where w is in V(G) and w $\neq$ u, v.

A *VLSI layout problem* of a graph G is to produce an area-efficient layout for G. It is shown that a VLSI layout problem of a forest of trees is NP - Complete [2]. The butterfly graphs have different representations including Omega network, the flip network, the baseline, and the reverse baseline networks. Each representation exhibits different characteristics [1, 5, 9, 13]. In other words, the butterfly network is drawn in different ways to exhibit different properties. In this paper, we introduce a new representation of Benes network, which helps solving the VLSI layout problem in Benes networks. We focus on laying out the Benes network on a square grid. Our aim is to produce an efficient VLSI layout of Benes without knock-knees. Our VLSI layout of B(r) is a square area. The area of this layout is $(2^{(r+2)} – 1)$ $(2^{(r+2)} – 1)$ which is $O(2^{2r})$. This is satisfactory since the lower bound of the area of a VLSI layout of B(r) is $O(2^{2r})$.

2.0 Proposed VLSI Representation of Benes Network

In this section, we discuss a representation of Benes network, which is suitable for VLSI layout. The proposed representation is shown in Figure 2. To avoid confusion between the two representations of Figure 1 and Figure 2, the representation in Figure 1 will be called *Rearrangeable representation of Benes* and the representation in Figure 2 will be called *VLSI representation of Benes*. A similar representation for butterfly is studied by Dinitz et. al. [5, 15].

The proposed VLSI representation of Benes network is constructed recursively as follows: Two (r-1)-dimensional Benes networks B(r – 1) form mirror images with respect to an array of level 0 and level 2r nodes. Each 4-cycle is drawn as a diamond. Particularly, the level 0 and level 2r nodes are the

vertices belonging to chordless 4-cycles in the diamond formation bridging the two (r-1)-dimensional Benes networks B(r – 1). See Figure 2. This representation provides a structural visualization, an in-depth understanding about the cyclic properties and the organization of spanning trees of Benes networks.

The description of cyclic structures is an important problem in graph theory. The following observations of Benes network, which are similar to butterfly, are straightforward from the VLSI representation given in Figure 2.

Lemma 2.1 [10]:

1. A Benes network is bipartite.
2. No two 4-cycles of B(r) have a common edge.
3. The edge set E of B(r) is disjoint union of 4-cycles, that is, there are $2r \times 2^{(r-2)}$ number of 4-cycles. □
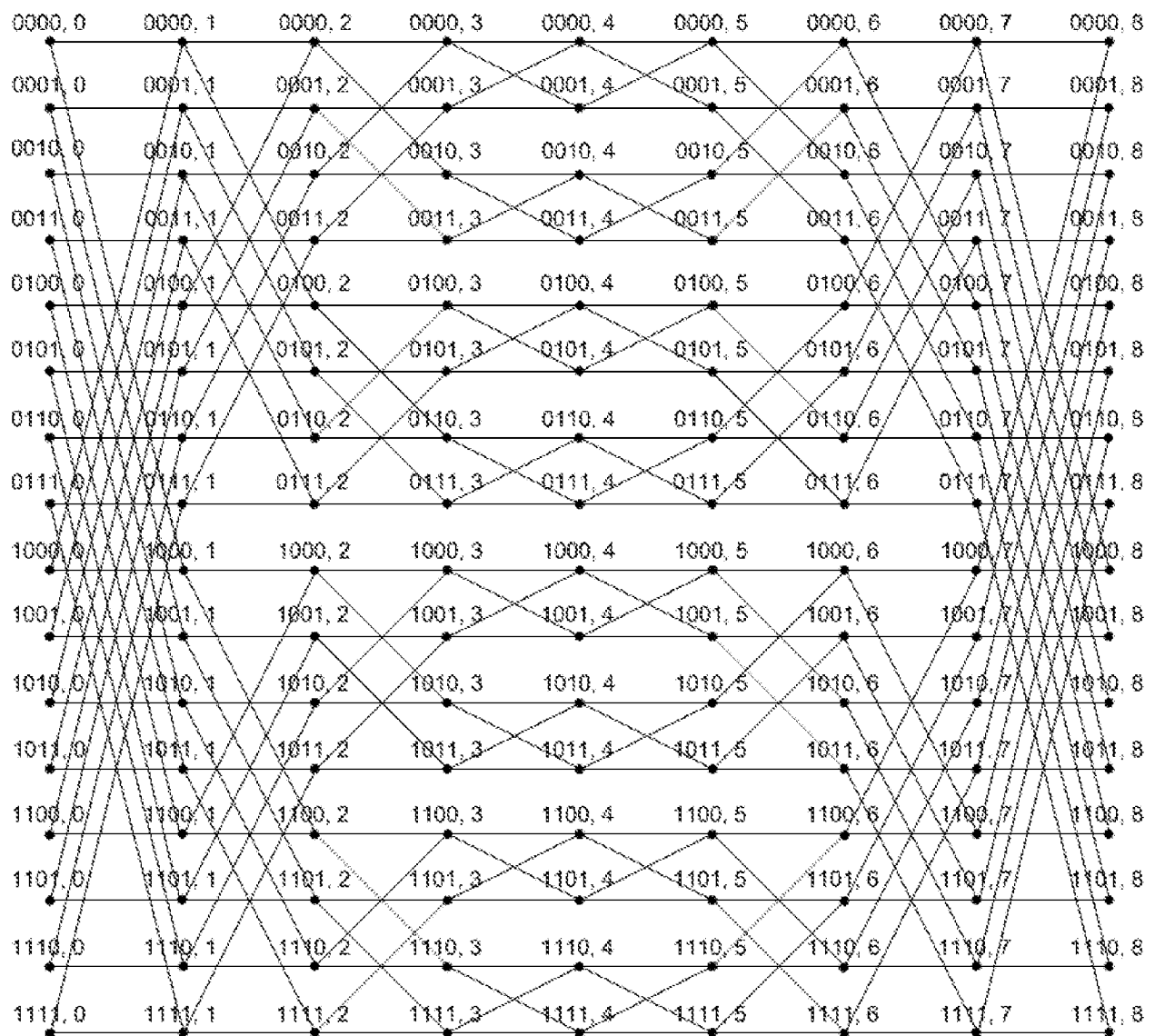
Figure 1: Rearrangeable representation of Benes B(4)

Even though the Benes network consists of back-to-back butterflies, there is a subtle structural difference between Benes and butterfly. See Figure 7. The removal of level 0 nodes of BF(r) leaves two disjoint copies of BF(r − 1). In the same way, the removal of level r nodes of BF(r) leaves two disjoint copies of BF(r − 1). This recursive structure can be viewed in another way. The removal of level 0 nodes and level r nodes (removal of all nodes of degree 2) of BF(r) leaves 4 disjoint copies of a BF(r − 2). However the removal of level 0 nodes and level 2r nodes (removal of all nodes of degree 2) of B(r) leaves 2 disjoint copies of a B(r − 1). In other words, the butterfly has dual symmetry, which the Benes does not have.
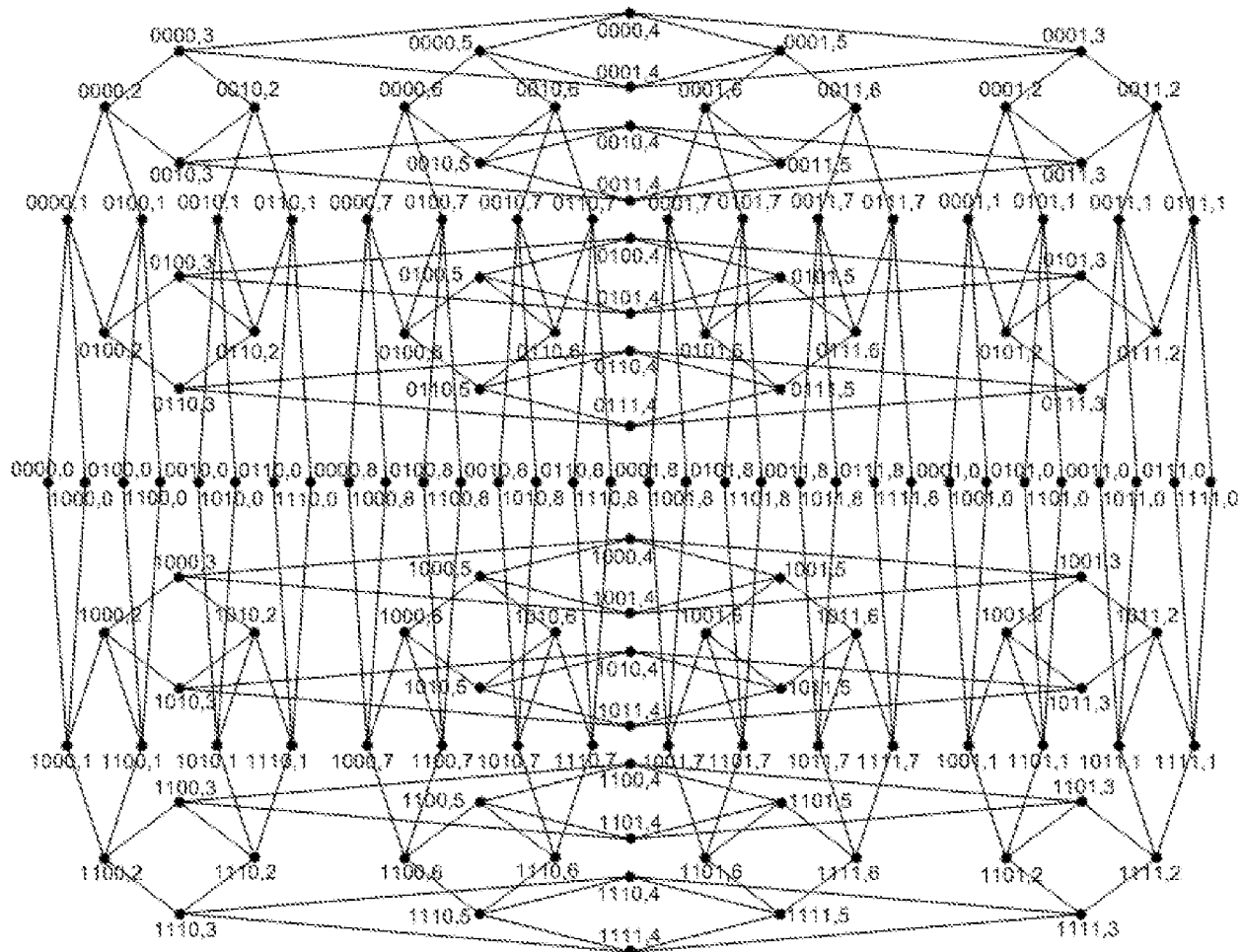
Figure 2: VLSI representation of Benes B(4).

Lemma 2.2: The Rearrangeable and VLSI representations of Benes are isomorphic.

Proof: We apply induction on the dimension of Benes B(r). Both 1-dimesional Rearrangeable representation and 1-dimesional VLSI representation are cycles of length 4. Let us assume that (k-1)-dimensional Rearrangeable representation and (k-1)-dimensional VLSI representation of Benes are isomorphic.

Now let us show that the k-dimensional Rearrangeable representation and the k-dimensional VLSI representation of Benes are isomorphic. Remove nodes of degree 2 (level 0 nodes and level 2k nodes) from both *k*-dimensional Rearrangeable representation and k-dimensional VLSI representation. By

induction hypothesis, the resultant Benes networks are isomorphic. The nodes of degree 2 (level 0 nodes and level 2k nodes) of both k-dimensional Rearrangeable representation and k-dimensional VLSI representation are organized in the same way as follows: The level 0 nodes $[0u_2\cdots u_k,0]$ and $[1u_2\cdots u_k,0]$ are adjacent to level 1 nodes $[0u_2\cdots u_k,1]$ and $[1u_2\cdots u_k,1]$ respectively. In the same way, level 2k nodes $[0u_2\cdots u_k,2k]$ and $[1u_2\cdots u_k,2k]$ are adjacent to level (2k – 1) nodes $[0u_2\cdots u_k,2k – 1]$ and $[1u_2\cdots u_k,2k – 1]$ respectively. Moreover, these nodes form a chordless cycle of length 4. These 4-cycles are edge disjoint with the rest of the graph. Hence both Rearrangeable representation and VLSI representation are isomorphic. □

3.0 A Simple VLSI Layout of Benes Network

Avior et. al. [1] have estimated that the r-dimensional butterfly network can be laid out in area $(1 + o(1))2^{2r}$ while no layout of the network can have area smaller than $(1 – o(1))\ 2^{2r}$. Dinitz et. al. [5, 15] have presented a layout whose encompassing rectangle is of area $(1/2)\ 2^{2r}+o(2^{2r})$, but this rectangle is $45°$ slanted w. r. t. the grid axes. There are different models of VLSI layouts for butterfly-like architectures [14, 15, 16]. Even though Benes network consists of back-back butterflies, these models of VLSI layouts of butterfly are not extendable to Benes. In this paper we provide a simple square VLSI layout without knock-knees for Benes which is of course applicable to butterfly networks too.

In the VLSI representation of Benes network, each 4-cycle is represented as a diamond. The 4-cycles with level r nodes of B(r) are in pairs. We call them *"Nested diamonds"*. Other 4-cycles are called *"Normal diamonds"*. In Figure 2, the six vertices [0000,3], [0000,4], [0000,5], [0001,3], [0001,4] and [0001,5] form a pair of nested diamonds. Notice that a pair of nested diamonds contains 6 nodes inducing two 4-cycles. Similarly, the four vertices [0000,2], [0000,3], [0010,2] and [0010,3] form a normal diamond.

VLSI Layout Algorithm and its Proof of Correctness

Our construction of VLSI layout of Benes is done in 3 steps:

Step 1: Draw the VLSI representation of Benes as in Figure 3.

Step 2: Each normal diamond is stretched to a rectangle as in Figure 4.

Step 3: Each pair of Nested diamonds is stretched along the grid lines as in Figure 5.
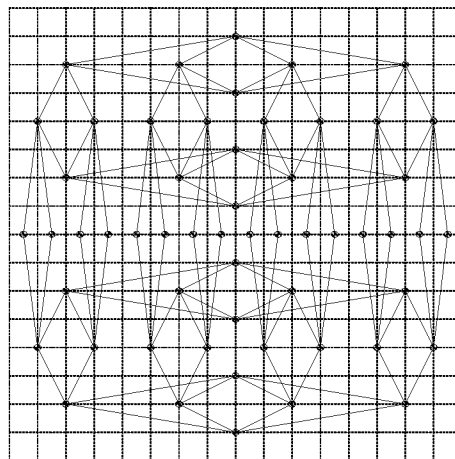
Figure 3: Step 1 – Drawing Benes B(3) on a Grid
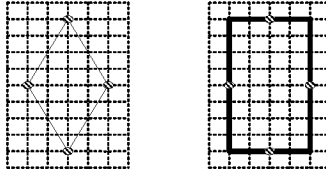
Figure 4: Step 2 - A Normal Diamond is stretched to a rectangle of the grid.
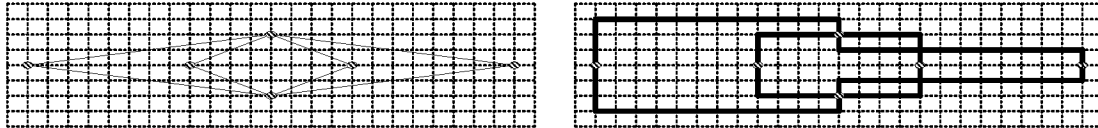
Figure 5: Step 3 - A pair of Nested Diamonds is stretched on grid axis.

The resultant layout is given in Figure 6. Now we claim that this grid embedding is indeed a VLSI layout. The proof of correctness is straightforward using induction hypothesis. Suppose it is true for $(k - 1)$-dimensional Benes. For a k-dimensional Benes, it is enough to consider the 4-cycles (normal diamonds) at level 0 and level 2k nodes. By the very structure of the VLSI representation of Benes, it is easy to see that the 4-cycles (normal diamonds) at level 0 and level 2k nodes can be drawn as a rectangle without violating VLSI requirements.
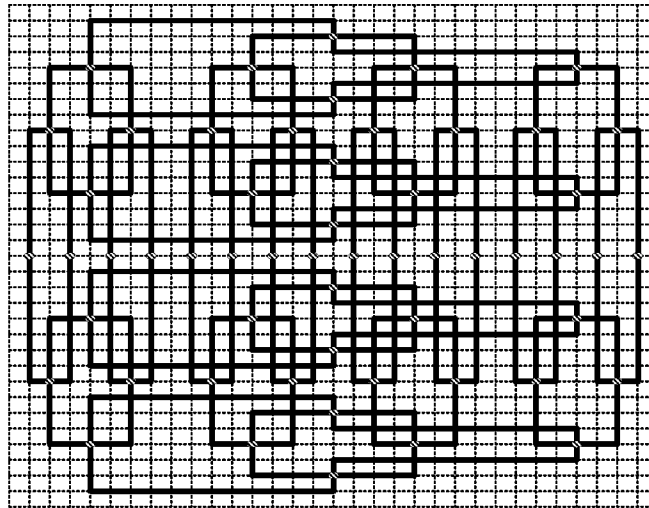
Figure 6: A VLSI layout of Benes B(3)

Area of this layout and Lower Bound for the Area of the VLSI Layout of B(r)

Let us now estimate the area of this VLSI layout of B(r). All the level 0 and level 2r nodes are placed in one horizontal grid axis with exactly one vertical grid axis between any two of these nodes. In the same way, all the level r nodes are placed in a vertical grid axis with exactly 3 horizontal grid axes between any two of these nodes. This observation is straightforward and it can be easily proved by induction. Thus, the area of this layout of B(r) is $(2^{(r+2)} - 1)$ $(2^{(r+2)} - 1)$ which is $O(2^{2r})$. Thompson [12] showed that, up to a constant factor, the layout area can be no less than the square of the bisection width. Since the bisection width of r-dimensional Benes network is $O(2^r)$ [8], the lower bound for the area of VLSI layout of Benes networks is $O(2^{2r})$.

4.0 Conclusion

Even though this paper focuses on Benes networks, all the results are applicable to butterfly too. We provide a simple VLSI layout without knock-knees for Benes network. This VLSI layout of B(r) is laid in a square area. The area matches with the lower bound up to a constant factor.

   Though wrapped butterfly is a butterfly-like architecture, it is not straightforward to extend these results to wrapped butterfly. The NP complete problems such as achromatic number problem and minimum crossing number problem [2,6] are open for Benes and butterfly networks. It is interesting to see whether these problems can be solved using this representation. □

5.0 References

[1]   A. Avior, T. Calamoneri, S. Even, A. Litman, and A. L. Rosenberg, *A tight layout of the butterfly network*, Theory of Computing Systems, Vol 31, No. 4, pp. 474-488, 1998.

[2]   S. N. Bhatt and F. T. Leighton, *A Framework For Solving VLSI Graph Layout Problems*, MIT/LCS/TR-305 44, 1983.

[3]   H. L. Bodlaender, *Achromatic number is NP-complete for co-graphs and interval graphs*, Information Processing Letters, Vol 32, No 3, pp. 135-138, 1989.

[4]   C. Bornstein, A. Litman, B. M. Maggs, R. K. Sitaraman, and T. Yatzkary *On the Bisection Width and Expansion of Butterfly Networks*, Theory of Computing Systems, Vol. 34, No. 6, pp. 491-518, 2001.

[5]   Y. Dinitz, S. Even, R. Kupershtok, and M. Zapolotsky, *Some compact layouts of the butterfly*, Proceedings of the eleventh annual ACM symposium on Parallel algorithms and architectures, Saint Malo, France , pp. 54-63, June 27-30, 1999.

[6]   M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP – Completeness*, W. H. Freeman and Company, 1979.

[7]   L. T. Q. Hung, M. M. Syslo, M. L. Weaver and D. B. West, *Bandwidth and density for block graphs*, Discrete Mathematics, vol. 189, pp. 163 - 176, 1998.

[8]   S. Konstantinidou, *The Selective Extra Stage Butterfly*, IEEE Transactions on very Large Scale Integration Systems, Vol. 1, No. 2, June 1993.

[9]   T. F. Leighton, *Introduction to parallel algorithms and architecture: Arrays, Trees, Hypercubes*, Morgan Kaufmann Publishers, ISBN 1-55860-117-1, 1992.

[10]   X. Liu and Q. P. Gu, *Multicasts on WDM All-Optical Butterfly Networks*, Journal of Information Science and Engineering, vol. 18, pp. 1049-1058, 2002.

[11]   D. Manlove and C. McDiarmid, *The complexity of harmonious coloring for trees*, Discrete Applied Mathematics, vol. 57, pp. 133–144, 1995.

[12]   C. D. Thompson, *Area-time complexity for VLSI*, Eleventh Annual ACM Symposium on Theory of Computing, (1979).

[13]   J. Xu, *Topological Structure and Analysis of Interconnection Networks*, Kluwer Academic Publishers, ISBN 1-4020-0020-0, 2001.

[14]   Chi-Hsiang Yeh , Behrooz Parhami , E. A. Varvarigos , H. Lee, *VLSI layout and packaging of butterfly networks*, Proceedings of the twelfth annual ACM symposium on Parallel algorithms and architectures, p.196-205, July 09-13, 2000, Bar Harbor, Maine, United States

[15]   Yefim Dinitz, Shimon Even, Maria Zapolotsky: *A Compact Layout of the Butterfly*, Journal of Interconnection Networks 4(1): 53-75 (2003)

[16]   S. Muthukrishnan, Michael S. Paterson, Suleyman Cenk Sahinalp, Torsten Suel, *Compact Grid Layouts of Some Multi-Level Networks*, 31st Symposium of Theory of Computing, Georgia, Atlanta, 455-463, May 1999.

DATA MOVEMENT TECHNIQUES FOR THE PYRAMID COMPUTER*

RUSS MILLER†‡ AND QUENTIN F. STOUT†§

Abstract. The pyramid computer was initially proposed for performing high-speed low-level image processing. However, its regular geometry can be adapted naturally to many other problems, providing effective solutions to problems more complex than those previously considered. We illustrate this by presenting pyramid computer solutions to problems involving component labeling, minimal spanning forests, nearest neighbors, transitive closure, articulation points, bridge edges, etc. Central to these algorithms is our collection of data movement techniques which exploit the pyramid's mix of tree and mesh connections. Our pyramid algorithms are significantly faster than their mesh-connected computer counterparts. For example, given a black/white square picture with n pixels, we can label the connected components in $\theta(n^{1/4})$ time, as compared with the $\Omega(n^{1/2})$ time required on the mesh-connected computer.

Key words. pyramid computer, graph-theoretic algorithms, image processing, component labeling, mesh-connected computer

AMS(MOS) subject classifications. 68Q25, 68Q20, 68U10

1. Introduction. Pyramid-like parallel computers have long been proposed for performing high-speed low-level image processing [4], [17], [24], [32], [34]. The pyramid has a simple geometry which adapts naturally to many types of problems, and which may have ties to human vision processing. The pyramid can be projected into a regular pattern in the plane, which makes it ideal for VLSI implementation, providing thousands or millions of processing elements. At least three pyramid computers for image processing are currently being constructed [12], [23], [30].

There is no reason to limit pyramid computers to low-level image processing. They can be adapted to many other problems, and should be considered as alternatives to machines such as the mesh-connected computer. To show this, we present several new fundamental pyramid computer algorithms which are significantly faster than their mesh-connected computer counterparts. These algorithms solve problems in graph theory, image processing, and digital geometry.

The pyramid computer we consider is a combination of tree and mesh structures. Complete definitions appear in § 2, with the essentials being that a pyramid of size n has an $n^{1/2} \times n^{1/2}$ mesh-connected computer as its base, and $\log_4(n)$ levels of mesh-connected computers above. A generic processing element (PE) at level k is connected to 4 siblings at level k, 4 children at level $k-1$, and a parent at level $k+1$. (See Fig. 1.)

To date, the literature on pyramids primarily consists of two classes of algorithms. The first concentrates on the tree structure, using child–parent links. Examples of this are the component labeling in [6], [29], the feature extraction in [20], the median filtering in [31], the selection in [25], the single-figure convexity in [15], and the polygon construction in [21]. These algorithms work efficiently only when the amount of data

---

TECHNIQUES FOR THE PYRAMID COMPUTER   39
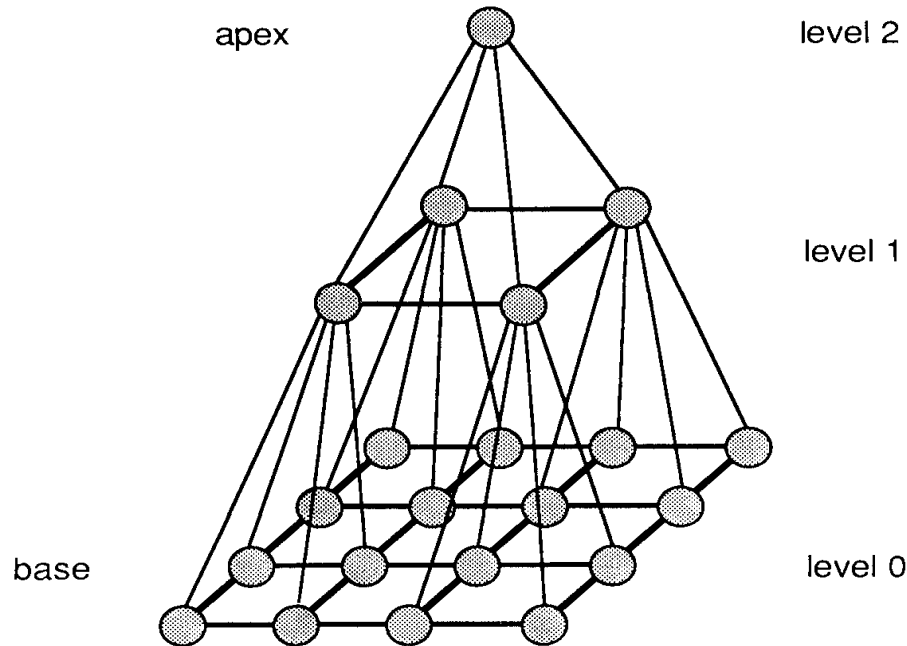
FIG. 1. *A pyramid computer of size* 16.

can be drastically reduced; otherwise too much data must pass through the apex, creating a bottleneck. The second class of algorithms concentrates on the mesh, essentially ignoring everything above the base. Examples of this are the sorting and median filtering in [25], matrix multiplication, and the multiple-label convexity in [15]. Reference [25] shows that the excessive data movement requirements of sorting force any pyramid algorithm to take $\Omega(n^{1/2})$ time. Since the base mesh can sort in $\theta(n^{1/2})$ time, the mesh oriented approach to sorting is within a multiplicative constant of being optimal.

In this paper, we consider a third class of algorithms which utilizes both types of connections. The basic approach is to reduce $O(n)$ pieces of intial data, stored one piece per base PE, down to $O(n^{1/2})$ pieces of data from which the desired result can be obtained. As has been noted for other models [10], [16], [18], this final information should be quickly moved to a region where interprocessor communication is as fast as possible, and once the answer has been obtained the results should be quickly moved to their final locations. For the pyramid this suggests moving the $O(n^{1/2})$ pieces to the middle level, which is an $n^{1/4} \times n^{1/4}$ mesh. The movement to and from the middle level is often the most time-consuming part of the algorithm, so we have developed a collection of efficient operations for performing these data movements, as well as techniques for reducing the amount of movement required.

These new data movement operations are presented for several algorithmic strategies, such as divide-and-conquer, and for various formats of the input data. They are used in several different algorithms, some of which solve various versions of the connected component labeling problem defined in § 2. In § 3, we use the pyramid read and pyramid write operations in an algorithm which labels the components of a graph of $\theta(n^{1/2})$ vertices in $\theta(n^{1/4} \log(n))$ time, where the graph is given as unsorted edges stored one per base PE. In § 4, we show that if the input is organized as an adjacency matrix, then the faster pyramid matrix read and pyramid matrix write operations reduce the time to $\theta(n^{1/4})$. In § 5, we consider input which is a digitized black/white picture,

for which we wish to label the black figures. Since each black pixel is a vertex, there may be $\theta(n)$ vertices, but the geometry of the situation allows us to use the funneled read operation to complete the labeling in $\theta(n^{1/4})$ time. These times are far better than the $\Omega(n^{1/2})$ required on a mesh-connected computer of size n [2], [18], [37].

Section 5 also introduces the operation of reducing a function over a cross-product. This is used to solve a nearest neighbor problem in which for each black component we wish to determine the label of and distance to the nearest black component. This operation is somewhat unusual in that once the relevant data has been collected at the proper level of the pyramid, it is then spread downward to finish the calculations.

In § 6, we give the detailed implementations of the data movement operations and also consider the optimality of our algorithms. In § 7, we extend the operations to pyramids of other dimensions. Throughout the paper we also solve related problems, such as marking minimal weight spanning forests, finding the transitive closure of a symmetric boolean matrix, marking articulation points, and deciding if a graph is bipartite.

2. Definitions. The *mesh-connected computer* (MCC) *of size n* is a single instruction stream-multiple data stream (SIMD) machine in which n processing elements (PEs) are arranged in a square lattice. (We assume that n is a perfect square.) PE (i, j), $1 \le i, j \le n^{1/2}$, is connected via unit-time communication links to PEs $(i \pm 1, j)$ and PEs $(i, j \pm 1)$, assuming they exist. See [7], [14], [16], [18], [33] for an overview of the MCC.

A *pyramid computer* (PC) *of size n* is an SIMD machine that can be viewed as a full, rooted, 4-ary tree of height $\log_4 (n)$, with additional horizontal links so that each horizontal *level* is an MCC. A PC of size n has at its base an MCC of size n, and a total of $(4/3)n - (1/3)$ PEs. The levels are numbered so that the base is level 0 and the apex is level $\log_4 (n)$. A PE at level i is connected via bidirectional unit-time communication links to its 9 neighbors (assuming they exist): 4 siblings at level i, 4 children at level $i - 1$, and a parent at level $i + 1$. (See Fig. 1.) We make the standard assumptions that each PE has a fixed number of words (registers), each of length $\theta(\log (n))$, and that all operations take unit time. Each PE contains registers with its row, column and level coordinates, the concatenation of which provides a unique label for the PE. (These registers can be initialized in $\theta(\log (n))$ time if necessary.)

We will illustrate the use of our data movement techniques by giving solutions to a variety of problems. Each problem involves a graph $G = (V, E)$ where V is the set of vertices and E is the set of edges. The graph can be expressed in various forms:

(a) *Unordered edge input.* The edges of the graph are initially distributed in a random fashion throughout the base of the PC, no more than one edge per PE.

(b) *Adjacency matrix input.* PE(i, j) at the base of the PC contains entry $A(i, j)$ of the adjacency matrix A of the graph.

(c) *Digitized picture input.* A digitized black/white picture is initially stored one pixel per PE in the base of the PC. The vertex set consists of the black pixels, where neighboring black pixels have an edge between them.

The problems we solve are the following:

(1) *Component labeling.* The input to the problem is an undirected graph $G = (V, E)$, given in any of the three input formats. We assume that the elements of V have a linear order. The component labeling algorithm assigns to each vertex a component label, where two vertices receive the same component label if and only if there is a connected path between them. In this paper, the component label will be chosen to be the minimum label of any vertex in the component.

(2) *Minimal spanning forest.* Given a weighted undirected graph, mark the edges of a minimal weight spanning tree for each component of the graph. The input format can be unordered edges or a weight matrix.

(3) *Nearest neighbor.* Given a digitized picture at the base of the PC with its components already labeled, calculate for each black component the label of the component nearest to it and its corresponding distance. Any l_p metric can be used to measure the distance.

(4) *Transitive closure.* Compute the transitive closure of a symmetric boolean matrix initially given at the base of the PC.

(5) *Bipartite graphs.* Given an undirected graph $G = (V, E)$, decide if G is bipartite. That is, decide if V can be partitioned into sets $V1$ and $V2$ so that each edge of G joins a member of $V1$ to a member of $V2$.

(6) *Cyclic index.* Compute the cyclic index of an undirected graph $G = (V, E)$, where the cyclic index of G is the largest number s so that V can be partitioned into sets $V(0), \ldots, V(s-1)$, such that for any edge (x, y), if x is in $V(i)$ then y is in $V((i \pm 1) \bmod s)$.

(7) *Bridge edges.* Given an undirected graph, decide which edges are bridge edges, where a bridge edge is an edge whose removal increases the number of components.

(8) *Articulation points.* Given an undirected graph, decide which vertices are articulation points, where a vertex is called an articulation point if its removal (along with its incident edges) increases the number of components.

(9) *Biconnectivity.* Given an undirected graph, decide if all components are biconnected, where a component is said to be biconnected if, for any two points in the component, there are two disjoint paths between them.

3. Graphs as unsorted edges. In this section, we assume that the graphs are given as unsorted edges stored one per PE at the base of the pyramid, where edges may be represented more than once. This format is the most general, including the others as special cases.

3.1. Data movement operations. There are several well-known data movement operations for the MCC. Two of these, the random access read (RAR) and random access write (RAW), will be defined here for the MCC and then extended to the PC. These operations involve two sets of PEs, the *sources* and the *destinations.* Source PEs send zero, one or two records, each consisting of a key and one or more data parts. (The upper limit of two records simplifies our algorithms. Most authors allow only one, in which case the operation needs to be repeated.) Destination PEs receive zero, one or two records sent by the source PEs. We allow the possibility that a PE is both a source and a destination.

MCC *Random Access Write* (RAW): In a RAW the destination PEs specify the number of records they wish to receive. At the end of the RAW, the number of records received by a destination PE is between zero and the number requested. Each record sent by a source PE is received by a destination PE, with the exception that if two or more source PEs send records with the same key then only the minimum such record is received. (In other circumstances the minimum could be replaced by any other commutative, associative, binary operation computable in $\theta(1)$ time.)

MCC *Random Access Read* (RAR): In a RAR no two records sent by source PEs can have the same key. Each destination PE specifies the keys of the records it wishes to receive, or it specifies a null key, in which case it receives nothing. Several destination PEs can request the same key. A destination PE may specify a key for which there is no source record, in which case it receives a null message.

42   RUSS MILLER AND QUENTIN F. STOUT

Both the RAW and the RAR can be completed in $\theta(n^{1/2})$ time on an MCC of size n [19]. On the PC, the RAW and RAR extend to the pyramid write and pyramid read, respectively. We now describe the actions of these operations, deferring the details of their implementations to § 6.1.

A *pyramid write* has the source and destination PEs playing the same roles as in the MCC's RAW. Furthermore, all destination PEs must lie on one level and all source PEs must lie on the same level or some level below. (If both levels are the same then a given PE might be both a source and a destination.) As an example, consider the following "sample" call:

> Pyramid write from level L up to level M,
> > For all PEs on level L,
> > > if *test1* then send($A1,B1,C1$), send($A2,B2,C2$);
> > For all PEs on level M,
> > > if *test2* then receive(D,E,F);

Since source PEs are descendents of destination PEs it must be that $L \leq M$ in this example. *test1* and *test2* are arbitrary boolean tests. For a PE on level L, if *test1* is true then the PE creates and sends two records, one whose key is the value of $A1$, with the values of $B1$ and $C1$ as data, and a second with key $A2$, with data $B2$ and $C2$. (The key is always the first component.) If *test1* is false then the PE does not send any records. A PE on level M will not try to receive any record if *test2* is false. If *test2* is true it will try to receive a single record, where the key goes into D and data parts go into E and F. If no record is received then the values of D, E, and F become ∞.

A *pyramid read* parallels the RAR in the same way that the pyramid write parallels the RAW, except now the destination PEs are descendents of the source PEs. As an example, consider

> Pyramid read at level L from level M,
> > For all PEs on level M,
> > > if *test1* then send(A,B)
> > For all PEs on level L,
> > > if *test2* then receive(C,D);

If a PE on level L requests a key C which has not been sent then D is set equal to ∞.

In § 6.1, we show how to implement the pyramid write and pyramid read. If the top level is an MCC of size m and the bottom level is $i-1$ levels below, then the time for both operations is $\theta(i+(m*i)^{1/2})$.

3.2. Component labeling. Except for obvious differences in the computer model and the data movements operations, our component labeling algorithm is similar to those in [9], [11], [13], [18], to which the reader is referred for proofs and further details. The algorithm proceeds through a series of stages, where at each stage the vertices are partitioned into disjoint *clubs*. Vertices are in the same club only if they are in the same component of the graph. We say that a club is *unstable* if it is not an entire component.

Initially each vertex is its own club. During a single *stage* of the algorithm, unstable clubs are merged together to form larger clubs, and the number of unstable clubs decreases by at least half. This is repeated until no unstable clubs remain. Since each stage reduces the number of unstable clubs by at least half, at most $\lfloor \log_2(v) \rfloor$ stages are needed for a graph with v vertices.

TECHNIQUES FOR THE PYRAMID COMPUTER   43

Each club has a unique label, which is the minimum label of any vertex in the club. During the algorithm, let $L(x)$ denote the current label of the club containing vertex x. Initially $L(x) = x$. During a stage, clubs are merged as follows: let u be a label of an unstable club. Compute $M(u) = \min\{L(y): (x, y) \in E, L(x) = u\}$. The graph whose vertices are the labels of unstable clubs and whose edges are of the form $(u, M(u))$, u an unstable club, is called a *min-tree forest*. In this forest each unstable club is connected to at least one other, which guarantees that the number of unstable clubs is at least halved after each stage. For each tree in the min-tree forest, we form a new club which is the union of all the clubs in the tree. For each unstable club u, let $N(u)$ denote the resulting label of the tree containing u. Since $N(u)$ is the minimum label of any club in u's tree, it is the minimum label of any vertex in the new larger club which contains all the vertices originally in u. For each vertex x, the new value of $L(x)$ is $N(L(x))$, and the stage is completed.

Our component labeling algorithm for a PC is given in Fig. 2. It incorporates an integer function count_keys which counts the number of distinct keys in the base. The operation of count_keys is similar to that of the pyramid write, and is given in detail in § 6.1.

An important point of the implementation is moving the data to a place where the min-tree forest relabeling can be done quickly. The forest is essentially upward directed, in that $M(u) \leq u$, and this makes it easier to label. Reference [18] showed that if the forest has f vertices then the relabeling can be done in $\theta(f^{1/2})$ time on an MCC of size $\theta(f)$. If the forest data remained at the base, then the relabeling would take $\theta(n^{1/2})$ time. However, by first moving the data up the pyramid and then relabeling it, this step will take only $\theta(v^{1/2})$ time. We use forest_level to indicate the level of the PC on which the forest is formed. Initially, this level must have at least v PEs. Each stage reduces the size of the forest by at least half, so after 2 stages forest_level can be increased by 1. Without this upward movement, the time would increase by a factor of $\log(n)$.

THEOREM 1. *Given a pyramid computer of size n, if the base contains the unsorted edges of an undirected graph with v vertices, then the above algorithm labels the components in $\theta\ (\log(n) + v^{1/2}[1 + \log(n/v)]^{1/2})$ time.*

Proof. Proposition 3 of § 6.1 shows that count_keys finishes in the time claimed. Within the loop, at the start of an iteration, let k be the number of PEs at level forest_level. The pyramid read and write take $\theta(\text{forest\_level} + k^{1/2}[1 + \text{forest\_level}]^{1/2})$ time and the min-tree forest relabeling takes $\theta(k^{1/2})$ time. Since $k = n/4^{\text{forest\_level}}$, the time for this iteration of the loop is

$$\theta(\text{forest\_level} + n^{1/2}[1 + \text{forest\_level}]^{1/2}/2^{\text{forest\_level}}).$$

The initial value of forest_level is $\lfloor\log_4(n/v)\rfloor$ and forest_level increases every 2 iterations, so the total time of the algorithm is

$$\theta\left(\sum_{i=\lfloor\log_4(n/v)\rfloor}^{\log_4(n)} \frac{i + n^{1/2}[1+i]^{1/2}}{2^i}\right) = \theta\left(\log(n) + v^{1/2}\left[1+\log\left(\frac{n}{v}\right)\right]^{1/2}\right). \qquad \square$$

In the worst case, when $v = \theta(n)$, our PC algorithm takes $\theta(n^{1/2})$ time, which is better than the $\theta(n^{1/2}\log(n))$ MCC algorithm of [18]. This situation arises, for example, when considering planar graphs, for which $v \leq 3e - 6$ edges. For smaller values of v our improvement over MCC algorithms becomes even more pronounced. All MCC algorithms must take $\Omega(n^{1/2})$ time, but for a dense graph with $v = \theta(n^{1/2})$ our pyramid algorithm requires only $\theta(n^{1/4}\log^{1/2}(n))$ time.

44   RUSS MILLER AND QUENTIN F. STOUT

```
For all base PEs,
      Label1:=Vertex1;  Label2:=Vertex2;
      If Vertex1<∞  then  create a record with key Vertex1,
                          create a record with key Vertex2;


v:=count_keys;    {v is the number of vertices}


forest_level:=⌊log₄(n/v)⌋;


for stage:=1 to ⌊log₂(v)⌋ do


      Pyramid write from the base upto level forest_level,
            For all base PEs,
                  send(Label1, Label2), send(Label2, Label1);
            For all PEs at level forest_level,
                  receive(Vertex1, Vertex2);


      Relabel the min-tree forest, so that each PE on level forest_level has
            Label1:=N(Vertex1).


      Pyramid read at the base from level forest_level,
            For all PEs on level forest_level,
                  if Vertex1<∞ then send(Vertex1, Label1);
            For all base PEs,
                  receive(Label1, temp_label1), receive(Label2, temp_label2);


      For all base PEs,
            if temp_label1<∞ then Label1:=temp_label1,
            if temp_label2<∞ then Label2:=temp_Label2;


      If (stage mod 2)=0  then forest_level:=forest_level + 1;


end {for};
```

FIG. 2. *Component labeling algorithm.*

3.3. Minimal spanning forests. The strong similarities between component labeling algorithms and minimal spanning forest algorithms are well known. In particular, others have noted that small changes to a component labeling algorithm for a parallel computer can give a minimal spanning forest algorithm for the same computer [3], [10], [22]. There are two changes that must be made to our component labeling algorithm. First, one must keep track of which edges are used. Second, when clubs are being merged each club must use an edge of minimal weight, rather than an edge to a club of minimal index. Furthermore, a club may have more than one minimal edge, which may introduce cycles. We prevent this by ordering the edges. We say that weighted edge $(w1, x1, y1)$ is less than weighted edge $(w2, x2, y2)$ if $w1 < w2$, or if

$w1 = w2$ and min $(x1, y1) < $ min $(x2, y2)$, or if $w1 = w2$, min $(x1, y1) =$ min $(x2, y2)$, and max $(x1, y1) < $ max $(x2, y2)$.

Incorporating these changes is quite straightforward, giving the following result:

THEOREM 2. *Given a pyramid computer of size n, if the base contains the unsorted weighted edges of an undirected graph with v vertices, then the above algorithm finds a minimal spanning forest in* $\theta(\log (n) + v^{1/2}[1 + \log (n/v)]^{1/2})$ *time.*

Even if the edges are unweighted, spanning forests can be quite useful. We illustrate this with an example. To decide if an undirected graph $G = (V, E)$ is bipartite, let each edge have weight 1 and use Theorem 2 to select a spanning forest. Using a pyramid write, write the edges of the forest to level $\lfloor \log_4 (n/v) \rfloor$. In each tree of the forest, select the vertex of minimum label as the root, and use the MCC algorithm in [27] to determine the depth of each vertex in its rooted tree. (This algorithm takes $\theta(v^{1/2})$ time.) Say that a node is in $V1$ if its depth is even, and is in $V2$ if its depth is odd. It is easy to show that G is bipartite if and only if this particlar choice of $V1$ and $V2$ is such that every edge of E joins a member of $V1$ and a member of $V2$. To check whether this property is true, have the base PEs use pyramid reads to determine the depths of the vertices of the edges they contain. Finally, pass these results to the apex, combining them along the way.

The above algorithm takes $\theta(\log (n) + v^{1/2}[1 + \log (n/v)]^{1/2})$ time. Furthermore, we can solve several graph-theoretic problems by using Theorem 2 to pick a spanning forest, moving the forest to level $\lfloor \log_4 (n/v) \rfloor$, using an MCC algorithm at that level, and using pyramid reads and writes to move data up and down. MCC algorithms for several graph-theoretic problems are given in [27], and these can be incorporated in a PC algorithm as described, yielding:

COROLLARY 1. *Given a pyramid computer of size n, if the base contains the unsorted edges of an undirected graph G with v vertices, then in* $\theta(\log (n) + v^{1/2}[1 + \log (n/v)]^{1/2})$ *time one can*

    (a) *decide if G is bipartite,*

    (b) *determine the cyclic index of G,*

    (c) *find all bridge edges of G,*

    (d) *find all articulation points of G,*

    (e) *decide if G is biconnected.*

We should mention that some of the MCC algorithms of [27] are patterned after MCC algorithms in [2], with the difference that the algorithms in [2] require matrix input while those in [27] use only edge input. The algorithms of [2] are unsuitable because there may not be v^2 PEs to hold the adjacency matrix. More important, the algorithms of [2] are too slow because they use matrix calculations that take $\theta(v)$ time on a PC.

4. Graphs as adjacency matrices. In this section, we consider undirected graphs with $n^{1/2}$ vertices, where the graph is given as an adjacency matrix or weight matrix. We assume that the (i, j) entry of the matrix is stored in base PE (i, j). Because the input is now more structured, we are able to give algorithms which are slightly faster than those of § 3.

4.1. Data movement operations. The algorithms of this section require two new data movement operations, *pyramid matrix write* and *pyramid matrix read*. A pyramid matrix write performs the same basic action as a pyramid write and comes in two versions, one for rows and one for columns. In the row (column) version source PEs lie in the base, and those in the same row (column) send the same key. The pyramid matrix read performs the same basic action as a pyramid read, and also comes in two

versions. For the row (column) version, all destination PEs lie in the base, and those in the same row (column) request the same key.

Detailed implementations of these operations appear in § 6.2, where it is shown that if a pyramid matrix write (read) has its destination (source) PEs at a level which is an MCC of size k, $k \leq 2n^{1/2}$, then the time used is $\theta(\log(n) + k^{1/2}[1 + \log(n/k^2)]^{1/2})$. (Though we never have more than $n^{1/2}$ keys, we must allow $k = 2n^{1/2}$ since the highest level holding $n^{1/2}$ PEs actually has $2n^{1/2}$ PEs when $n > 256$ is an odd power of 4.)

4.2. Matrix algorithms. Our algorithms for graphs given as adjacency or weight matrices are simple adaptations of our algorithms for unsorted edges. We merely start with the previous algorithms, remove the call to count_keys, replace pyramid read with pyramid matrix read, and replace pyramid write with pyramid matrix write. The resulting algorithms are faster than those of § 3 by a factor of $\log^{1/2}(n)$.

THEOREM 3. *Suppose the adjacency matrix of an undirected graph with $n^{1/2}$ vertices is stored in the base of a pyramid computer of size n. Then the above algorithm labels the connected components in $\theta(n^{1/4})$ time.*

THEOREM 4. *Suppose the weight matrix of a weighted undirected graph with $n^{1/2}$ vertices is stored in the base of a pyramid computer of size n. Then the above algorithm marks a minimal spanning forest in $\theta(n^{1/4})$ time.*

COROLLARY 2. *Suppose the adjacency matrix of an undirected graph G with $n^{1/2}$ vertices is stored in the base of a pyramid computer of size n. Then in $\theta(n^{1/4})$ time one can*

  (a) *decide if G is bipartite,*
  (b) *determine the cyclic index of G,*
  (c) *find all bridge edges of G,*
  (d) *find all articulation points of G,*
  (e) *decide if G is biconnected.*

Determining the transitive closure of a symmetric boolean matrix stored in the base of a PC is a simple adaptation of component labeling. First perform component labeling for matrix input. For PEs which are storing off-diagonal entries (i.e., for which the row and column are different), the new entry is 1 if the row label equals the column label, while otherwise it remains 0. For PEs on the diagonal, if the original entry was 1 it remains so, while if it was 0 then it becomes 1 only if some other entry in the row is 1. Pyramid matrix reads and writes can be used to determine the proper diagonal entries, giving the following result:

COROLLARY 3. *Suppose an $n^{1/2} \times n^{1/2}$ symmetric boolean matrix is stored in the base of a pyramid computer of size n. Then the transitive closure can be determined in $\theta(n^{1/4})$ time.*

5. Divide-and-conquer algorithms. In this section, we use a divide-and-conquer approach to solve a variety of geometric problems involving black/white pictures stored one pixel per PE at the base of the PC. The use of divide-and-conquer for geometric problems is well known, but a naive use of this strategy on the pyramid computer does not necessarily produce good results. We demonstrate some efficient implementations of this strategy on the PC.

Throughout this section we will often divide the MCC at some level into squares of some size S. What we mean by this is that we will completely partition the MCC into disjoint squares of size S, where S is a power of 4. Using this partitioning, the concept of *the square of size S at level l containing* PE P is well defined (assuming that level 1 is of size S or greater). The term *picture square* will be used to refer to a square in the base.

TECHNIQUES FOR THE PYRAMID COMPUTER   47

The computations for our divide-and-conquer solutions will proceed in a bottom-up fashion. The first *stage* will involve analyzing picture squares of size 4^c, for some small constant c which depends upon the particular problem. In general, stage i has analyzed picture squares of size 4^{c+i-1}, and stage $i+1$ combines the results together to analyze picture squares of size 4^{c+i}. An important point of our solution strategy is that at the end of stage i, each picture square of size 4^{c+i-1} has been reduced to $O(2^i)$ records of data, from which stage $i+1$ can produce the analysis for picture squares of size 4^{c+i}. Our algorithms proceed rapidly by moving the data that represents a picture square up through the subpyramid whose base is the picture square.

For a picture square of size 4^{c+i-1}, the square of size $4^{c+\lceil i/2 \rceil -1}$ at level $\lfloor \frac{i}{2} \rfloor$ which contains all level $\lfloor \frac{i}{2} \rfloor$ ancestors of the PEs in the picture square is called the *data square corresponding to the picture square*. Notice that when stage i is working on picture squares of size 4^{c+i-1}, the corresponding data squares have enough PEs to contain the results from stage $i-1$, so all the work is performed in the data squares. Further, the data square corresponding to a picture square is either the union of the data squares corresponding to the picture's quadrants, or else it is the union of the parents of the quadrants' data squares. This means that the data from one stage is either already in place, or must move up only one level, for the next stage.

The last stage of the divide-and-conquer algorithm is stage $\log_4(n) - c + 1$, during which the entire picture is analyzed. Since the intermediate results are scattered in data squares throughout the pyramid, a final step is needed to move these results back down to the base. This final data movement is accomplished via a funnel read, described in § 5.1. Section 5.1 also introduces a data movement operation called reducing a function. This operation allows data squares to perform some calculations (such as computing the nearest neighbor for each point from a set of points) in time linear in the edgelength of the square, even though we do not know of an MCC algorithm which finishes in this time. The operation of reducing a function uses PEs below the data square to help perform the calculations in the desired time.

5.1. Data movement operations. We now describe data movement operations that will be used to implement divide-and-conquer algorithms on the PC.

Funnel read. Assume each base PE knows the key for data it wishes to read from its stage 1 data square. For a stage i data square which is responsible for supplying the data for a given key, there are three possibilities: either one of its PEs has the data, or it must read the data from its stage $i+1$ data square (where by its stage $i+1$ data square we mean the data square it supplies data to), or one of its PEs has an alias for the key and must read the data for the alias from its stage $i+1$ data square. (If i is the last stage, then the square must have the data.) Further, a data square of size S never receives more than S requests. The funnel read ultimately obtains the data for all of the base PEs in $\theta(S^{1/2})$ time, where S is the size of the data squares at the final stage. Figure 3 is a picture of a funnel read, and its detailed implementation is in § 6.3.

Reducing a function. Given sets Q, R, and S, let g be a function mapping $Q \times R$ into S, and let $*$ be a commutative, associative, binary operation over S. Define f, a map from Q into S, by $f(q) = *\{g(q, r): r \in R\}$. We say f is the *reduction of g*. For example, if Q and R are sets of points in some metric space, if S is the real numbers, if $g(q, r)$ is the distance from q to r, and if $*$ is the minimum, then $f(q)$ is the distance from q to the nearest point in R.

Suppose the elements of Q are stored one per PE in a square of size m at level i, and the elements of R are also stored one per PE in the square. (A PE may contain an element of Q and an element of R.) Suppose g and $*$ can both be computed in

stage 5 data square   level 2

stage 3 data square   level 1

stage 1 data square   base
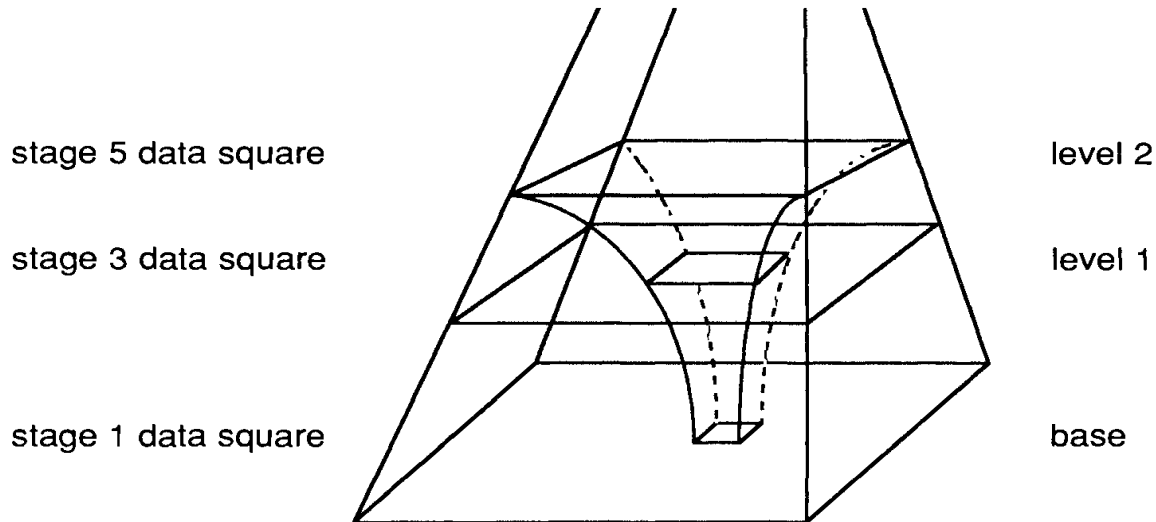
FIG. 3. *A single* PE's *view of a funnel read.*

$\theta(1)$ time. Then the operation of reducing a function will compute $f(q)$ and store it in the PE containing q, for all $q \in Q$, in $\theta(m^{1/2} + m/4^i)$ time. The detailed explanation of this operation appears in § 6.3.

5.2. Component labeling for digitized pictures. As was noted in § 2, a digitized picture can be viewed as being an undirected graph, where the black pixels are the vertices, and adjacent black pixels have an edge between them. Upon termination of our component labeling algorithm for digitized pictures, every base PE containing a black pixel will also contain the label of the pixel's component.

ALGORITHM. We follow the basic divide-and-conquer strategy outlined above. Our algorithm is similar to the MCC algorithm of [18], but is significantly faster.

First, each base PE P containing a black pixel generates a record (p, q, ∞) for each neighbor Q containing a black pixel, where p is the index of P, q is the index of Q, and the third component of the record will be used to store the component label of p and q generated at this stage. Thus, each PE may generate as many as four records, one for each of its neighbors.

At stage 1, picture squares of size 256 are labeled. (That is, $c = 4$ in the generic divide-and-conquer strategy.) This is accomplished by performing the component labeling algorithm for unsorted edge data, as described in § 3.2, simultaneously for every picture square of size 256 by using the edge data just generated in each picture square. That is, for each picture square we can apply the unsorted edge data labeling algorithm only to those records (x, y, ∞) for which both x and y are in the picture square, i.e., we omit those records for which x is on the border of the square and y is in an adjacent picture square. (Since x and y are concatenated coordinates of PEs, in $\theta(1)$ time a PE containing (x, y, ∞) can decide whether or not x and y are in the same picture square.)

When the edge data labeling algorithm is completed, those PEs with a record (x, y, ∞) use a RAR in their picture square to determine the (possibly) new label of x. The label is stored in the third field of this record. (If the third field was ∞ at the end of the edge data labeling algorithm, it must be that y lies outside of x's stage 1 picture square.) These records have all of the information needed for the next stage

of the algorithm, since the only components for which a single label has not yet been assigned are those which lie in two or more stage 1 picture squares. All records generated originally are kept in their stage 1 data squares, and each PE containing a record (x, y, z) with y outside the picture square generates a record (z, y, ∞) for use in the next stage. There are at most 64 such records generated within a single picture square of size 256. (Each corner pixel may generate 2 such records, and all other border pixels may generate 1.) These records are now spread out in their stage 1 data squares so that no PE holds more than one such record. This concludes stage 1.

At stage i, the algorithm labels picture squares of size 4^{3+i}, using data squares of size $4^{3+\lceil i/2 \rceil}$ at level $\lfloor \frac{i}{2} \rfloor$. If i is odd, then all of the data needed (the records generated at the end of the previous stage) is already present, while if i is even, then the data is in the four data squares one level below. In the latter case, the data is moved up and distributed so that no PE has more than one record. (This can be done in $\theta(2^{i/2})$ time.) Next, component labeling is performed for this edge input, again using only edges both of whose vertices lie in the picture squares. When finished, a PE containing a record with a vertex outside of the picture square generates a record for the next stage. Since we are working on picture squares of size 4^{3+i}, at most 2^{5+i} records can be generated for the next stage.

After stage $\log_4 (n) - 3$, all of the labels have been decided. Notice that if P's component extends outside of P's stage 1 picture square, then the labeling information in the stage 1 data square may be incorrect, and P would need to consult later stages. The component may extend outside of P's picture square for many stages, so in advance P does not know which data square has the needed labeling information. This is where a funnel read is used, moving labels from the data squares of the last stage back down towards the base, taking $\theta(n^{1/4})$ time. This gives the following result:

THEOREM 5. *The component labeling problem for digitized picture input on a pyramid computer of size n can be solved in $\theta(n^{1/4})$ time.*

This represents a substantial improvement over the $\theta(n^{1/2})$ MCC algorithm in [18]. Reference [29] recently presented a different PC algorithm for labeling components in a digitized picture. This algorithm is designed to label "convex blobs," but because it uses only child–parent links it takes $\theta(n^{1/2})$ time to label a $D \times n^{1/2}$ rectangle, for any constant D. In contrast, our algorithm will label any digitized picture, and hence all "convex blobs," in $\theta(n^{1/4})$ time.

5.3. Nearest neighbors. The solution to the nearest neighbor problem is quite similar to the solution just presented for the component labeling problem. Therefore, we will describe in detail only those aspects of the algorithm that change.

In the nearest neighbor problem, we wish to find the kin of each component, where the *kin* of a component is the label/distance pair representing the nearest labeled component with a different label. (In case of ties, the component of smallest label is chosen.) The input to the nearest neighbor problem is a digitized picture with its components already labeled, and at the conclusion of the algorithm each black pixel will have the kin information for its component.

Our divide-and-conquer algorithm is based on the following observation: assume that the 4 quadrants within a picture square have been analyzed. When combining the 4 quadrants, the only components whose kin could lie in a quadrant other than their own are those components that have at least one pixel that is an extreme point. An *extreme point* is a black pixel that is, relative to its component, either the northernmost or southernmost black pixel in its column, or the easternmost or westernmost black

pixel in its row. Within a quadrant, components with no extreme points must have determined their kin in earlier stages since they are totally surrounded by other components within their quadrant.

We again analyze squares of size 256 at stage 1. Within each picture square, for each component C we determine the closest component within the square, and store this in a record $(C, \text{kin}\,(C))$. (This kin information may be incorrect, but the final funnel read will bring the correct information down from a data square above.) For each column i in the picture square, form the records $(1, i, \text{tr}\,(i), \text{tl}\,(i))$, $(2, i, \text{br}\,(i),$ $\text{bl}\,(i))$, where $\text{tr}(i)\,(\text{br}(i))$ is the row of the topmost (bottommost) black pixel in the column restricted to the square, and $\text{tl}\,(i)\,(\text{bl}\,(i))$ is the label of this pixel. (If the column has no black pixel then set the coordinates to ∞.) Similarly, for each row j we form records $(3, j, \text{lc}\,(j), \text{ll}\,(j))$, $(4, j, \text{rc}\,(j), \text{rl}\,(j))$ for the leftmost and rightmost black pixels in the row. These are the records needed for the next stage.

In general, at stage $i + 1$ we first find, for each black pixel represented in one of the records passed on from stage i, the nearest black pixel (represented in a record) of a different label. We use the operation of reducing a function to do this, where Q and R are the records, S is the real numbers, $*$ is the minimum, and g is the distance, with the exception that g gives an infinite distance if the two points have the same label. When the operation is finished, we use an RAR to form a record $(C, \text{kin}\,(C))$ for each component C represented by one or more pixels. To generate the records for the next stage, notice that for each column in the stage $i + 1$ picture square there are two type 1 records. The one representing the topmost pixel is passed to the next stage, and similar reductions occur for records of types 2, 3 and 4.

Finally, after the last stage of the algorithm, a funnel read brings the correct kin information back to the base, giving the following result:

THEOREM 6. *The nearest neighbor problem for digitized picture input on a pyramid computer of size n can be solved in $\theta(n^{1/4})$ time.*

We note that if one is interested only in determining, for each black pixel, the location of the nearest black pixel, then the PC needs only $\theta(\log\,(n))$ time [26].

6. Data movement operations. In this section, we describe how to perform the data movement operations used in earlier sections. We also discuss the optimality of our algorithms.

6.1 Pyramid read, pyramid write, and count-keys. A pyramid read starts with records stored at some level i, each with a different key, and moves them down to level j where they can be read. Let $m = n/4^i$ and $S = 4 * \lfloor \log_4 {}^{\lceil m/(i-j+1)\rceil} \rfloor$. In this algorithm, we use the term *square* to mean "square of size S", and we divide levels $j \cdots i$ into squares. The squares on level i are numbered from 1 to m/S using a snake-like ordering. (See Fig. 4.) All of the data starting in square k on level i is called *packet k.*

By a *cycle* we mean $cS^{1/2}$ time units, where the constant c, independent of n and S, is chosen so that in one cycle a square can perform all of the following functions:

1. Exchange packets with the next square on the same level (where next is with respect to the snake-like ordering).
2. Copy a packet to the four squares on the level below.
3. Perform an MCC RAR.

We now describe the pyramid read algorithm. Packets are first passed backwards along level i towards square 1, using the snake-like ordering, one square per cycle. Once at square 1, a packet is moved forwards along level i, again using the snake-like ordering. Each time a square at level i receives a packet moving forwards, it first copies

0	1	2	3
7	6	5	4
8	9	10	11
15	14	13	12

FIG. 4 *Snake-like ordering.*

it to the four squares below before passing it along, all in one cycle. Each square at level $(j+1)\ldots(i-1)$ which receives a packet just copies it to the four squares on the level below. Finally, each time a square on level j receives a packet it does an MCC RAR.

PROPOSITION 1. *In a pyramid computer of size n, a pyramid read from level i to level j takes $\theta(i-j+1+[m*(i-j+1)]^{1/2})$ time, where $m = n/4^i$.*

Proof. The operation is finished when packet m/S has moved backwards to square 1, forwards to square m/S, down to level j, and all level j squares beneath square m/S have done a RAR. This takes $2*m/S-1+i-j$ cycles, or $\theta(i-j+1+[m*(i-j+1)]^{1/2}$ time. □

For the pyramid write, assume that the destination PEs are on level i, the source PEs are on level j, and m and S are defined as above. The pyramid write is basically performed by running the pyramid read in reverse. Slight differences arise because several base PEs can send records with the same key, but perhaps different data parts, in which case we need to take a minimum. Also, it is not initially known which packet a given record will end up in.

To accommodate these problems, in general a square Z will have a packet's worth of data from each square feeding into it (either the four squares below, or, for squares at level i, the four squares below and the preceding square in the snake-like ordering). From this, Z has enough to make at least one packet's worth of data. However, since the square to which it is feeding data may have some leftover data from the previous cycle, the square it is feeding informs Z how many records are needed. In one cycle, Z supplies the necessary data and informs each square feeding into it how many of that square's records need to be replaced. Since it takes one cycle to receive the data, and one cycle for Z to pass on data after the new data is received, each step of the pyramid write takes two cycles.

Making these minor changes to the pyramid read, we obtain:

PROPOSITION 2. *In a pyramid computer of size n, a pyramid write from level j to level i can be performed in $\theta(i-j+1+[m*(i-j+1)]^{1/2})$ time, where $m = n/4^i$.*

52   RUSS MILLER AND QUENTIN F. STOUT

Recall that count_keys is a function responsible for counting the number of distinct keys present in the base of a PC. If each key were represented only once, then count_keys could finish in $\theta(\log(n))$ time. However, keys in general are duplicated, so count_keys uses the pyramid write to eliminate duplicates. It first tries to determine if the number of keys is $\leq K$, where $K = 4^{\lceil \log_4(\log_4(n)) \rceil}$, by doing a pyramid write to level $L = \log_4(n/K)$, where each destination PE requests one record. When finished, all PEs below level L use a pyramid read to check whether their key was passed all the way up to level L. If this is so for all PEs, then the number of keys is the number of records at level L. Otherwise, count_keys sees if the number of keys is $\leq 4K$ by doing a new pyramid write to level $L-1$. It continues multiplying the number of keys by 4 at each stage until it reaches a stage where the pyramid write succeeds in moving all the keys to level L. This gives us the following result:

PROPOSITION 3. *If there are k different keys present in the base of a pyramid computer of size n, then in* $\theta(\log(n) + k^{1/2}[1 + \log(n/k)]^{1/2})$ *time count_keys will count them.*

6.2. Pyramid matrix read and pyramid matrix write.

Assume that a pyramid matrix write has its destination PEs at level i, and let $m = n/4^i$. (Recall that $m \leq 2n^{1/2}$.) The pyramid matrix write has two steps: the first moves the data to level $j = \log_4(m)$, and the second moves it to level i. (Note: if $m = 2n^{1/2}$ then set $j = i$ instead of $i+1$.)

To perform the first step of the row version of the pyramid matrix write (the column version is similar), we partition the PEs at level j into strings of $k = 2^j$ PEs all in the same row, and call such a string and all PEs beneath it a *prism*. Notice that a prism includes parts of k rows in the base, and hence sits over no more than k different keys. In each prism, at time j the first string PE receives the minimum record sent from any base PE beneath it in the first row of its prism. This PE passes the record on to the next PE in its string. In general, the computations are pipelined so that at time $j + r - 1 + p - 1$ the pth PE in the string of each prism receives the minimum record sent from any PE beneath it in the rth row of the prism, and also receives from the preceding string PE the minimum record sent from any base PE in the rth row beneath any of the preceding string PEs. The pth PE in the string takes the minimum of these two values and passes it to the $p+1$st PE in its string.

At time $j + k - 1$ the last PE in each string forms the minimum sent by any base PE in the first row of its prism, and this value is sent back towards the first PE of its string. These reverse messages are passed simultaneously with the previously mentioned ones. Finally, at time $j + 2 * k - 2$ the last string PE (the kth one) finds the minimum record sent by any base PE in the kth row of the prism. Simultaneously, the minimum record sent by any base PE in the first row of a prism has moved back to the first PE of its string, and the first step of the algorithm is finished.

The second step is just a pyramid write from level j to level i. This gives us the following result:

PROPOSITION 4. *In a pyramid computer of size n, a pyramid matrix write to level i, $i \geq \lfloor \log_4(n)/2 \rfloor$, takes* $\theta(\log(n) + m^{1/2}[2 + \log(n/m^2)]^{1/2})$ *time, where* $m = 4^i$.

Proof. If $m \leq n^{1/2}$, then the time for the first step is $\theta(m^{1/2})$, and the time for the second step is $\theta(i - j + 1 + m^{1/2}[i - j + 1]^{1/2})$. Since $j = \log_4(m)$ and $i = \log_4(n/m)$, we have the desired result. Otherwise, if $m = 2n^{1/2}$, then the time is $\theta(m^{1/2})$. In this case, $\log_4(n/m^2) = -1$, which is why there is a 2 instead of the usual 1 inside the brackets. □

For a pyramid matrix read, assume that the source PEs are at level i, and let $m = n/4^i$. Again we describe just the row version, which takes 3 steps. The first step uses prisms of height j, where j is as above. By using the first step of the matrix write,

in $\theta(m^{1/2})$ time the top row (string) of each prism contains the keys needed by the rows beneath. The second step is a pyramid read from level i to level j. The third step reverses the first one, taking the data to the base.

PROPOSITION 5. *In a pyramid computer of size n, a pyramid matrix read from level* i, $i \geqq \lfloor \log_4 (n)/2 \rfloor$, *takes* $\theta(\log(n) + m^{1/2}[2 + \log(n/m^2)]^{1/2})$ *time, where* $m = 4^i$.

6.3. Funnel read and reducing a function. The funnel read is straightforward. If the final stage of an algorithm is stage i, then we first have each stage $i - 1$ data square use a pyramid read to obtain the data from its stage i data square. (Notice that this runs in time linear in the size of the stage i data square.) We continue downwards, each stage $j - 1$ data square using a pyramid read to obtain its data from its stage j data square. When going down, the squares get smaller by a factor of 4 (see Fig. 3), so we obtain the following:

PROPOSITION 6. *Assume that the final stage of an algorithm is stage* i, *and that the stage* i *data square is an* MCC *of size S. Then a funnel read runs in* $\theta(S^{1/2})$ *time.*

Let g, Q, R, S, $*$, f, m, and l be as in the description of reducing a function in § 5.1. We use the notation $G(A, B)$ to denote the function defined on a subset A of Q whose value at $a \in A$ is given by $G(A, B)(a) = *\{g(a, b): b \in B\}$. Notice that f is $G(Q, R)$.

For a set of PEs S, by *computing* $G(A, B)$ *in S* we mean that for each element a in A there is a PE in S which computes and stores the value of $G(A, B)(a)$. Notice that if a set $A \subset Q$ is partitioned into subsets $A1$, $A2$, $A3$, and $A4$, then

$$G(A, B) = G(A1, B) \cup G(A2, B) \cup G(A3, B) \cup G(A4, B),$$

where we view a function as a set of ordered pairs. Also, if a set $B \subset R$ is partitioned into subsets $B1$, $B2$, $B3$, and $B4$, then

$$G(A, B)(a) = G(A, B1)(a) * G(A, B2)(a) * G(A, B3)(a) * G(A, B4)(a)$$

for any $a \in A$.

Using these observations, our operation of reducing a function is also straightforward. If $m = 1$, then obviously the single PE just computes its value in $\theta(1)$ time. If $l = 0$ (i.e., if all of the data is at the base), then we merely circulate all values of R among all PEs holding members of Q, and each such PE calculates the associated f value. This takes time proportional to m, the number of PEs. Otherwise, Q is partitioned into 4 subsets (as equal as possible), $Q1$, $Q2$, $Q3$ and $Q4$, and these are arranged in the four quadrants of level l, as in the left-hand side of Fig. 5. The quadrant holding Qi is responsible for computing $G(Qi, R)$. It does this by first having each PE copy its element of Q into the PE's four children. The quadrant also copies all of R to the square beneath it, creating the situation as in the right-hand side of Fig. 5. A square of size $m/4$ on level $l - 1$ holding Qi and Rj is now responsible for computing $G(Qi, Rj)$, which it does recursively. When this is finished, beneath each quadrant of level l the four squares of size $m/4$ at level $l - 1$ send up their results. A PE at level l holding an element $q \in Q$ receives $G(Q, R1)(q)$, $G(Q, R2)(q)$, $G(Q, R3)(q)$ and $G(Q, R4)(q)$ from its children, and by taking the $*$ of these computes $G(Q, R)(q)$.

PROPOSITION 7. *Suppose the elements of Q are stored one per* PE *in a square of size m at level* l, *and the elements of R are also stored one per* PE *in this square, and suppose g and $*$ can be computed in* $\theta(1)$ *time. Then the reduction of g can be computed in* $\theta(m^{1/2} + m/4^l)$ *time.*

Proof. It takes $\theta(m)$ time to copy the values of Q and R from level l to the appropriate places on level $l - 1$. Since the size of the squares reduces by a factor of 4 at each level, it takes $\theta(m)$ time to copy the values all the way down to the base.

54   RUSS MILLER AND QUENTIN F. STOUT

$Q_1\ R_1$	$Q_2\ R_2$
$Q_4\ R_4$	$Q_3\ R_3$

$Q_1\ R_1$	$Q_1\ R_2$	$Q_2\ R_1$	$Q_2\ R_2$
$Q_1\ R_4$	$Q_1\ R_3$	$Q_2\ R_4$	$Q_2\ R_3$
$Q_4\ R_1$	$Q_4\ R_2$	$Q_3\ R_1$	$Q_3\ R_2$
$Q_4\ R_4$	$Q_4\ R_3$	$Q_3\ R_4$	$Q_3\ R_3$

level i   level i-1
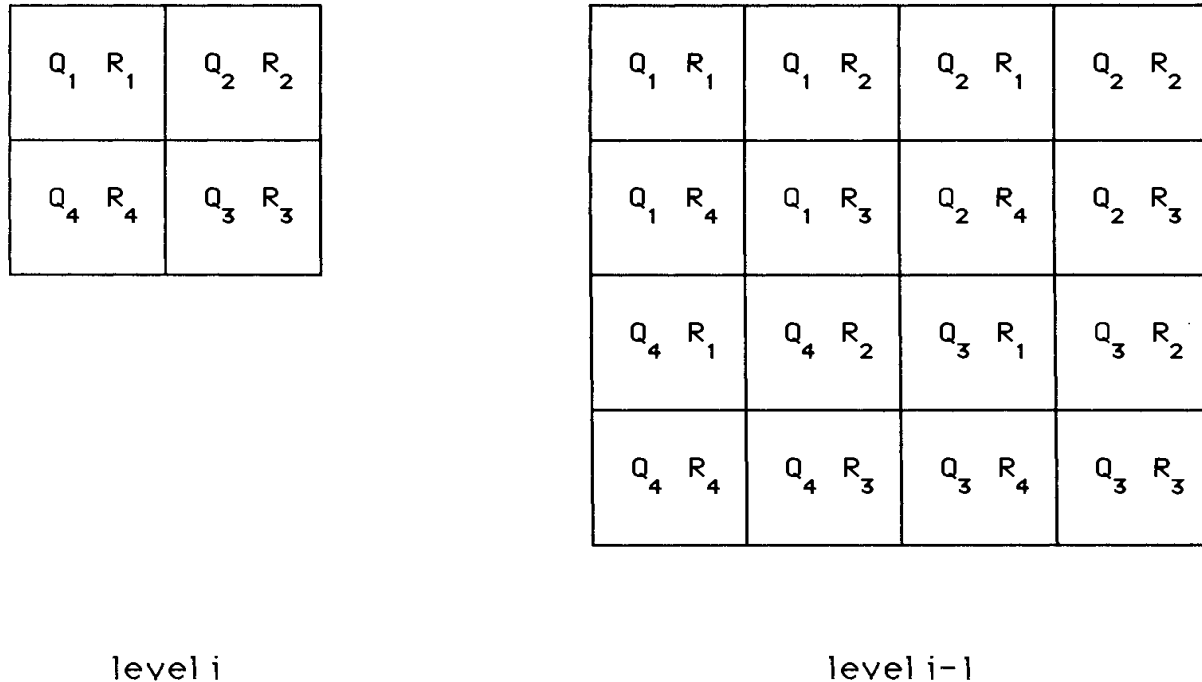
FIG. 5. *Reducing a function.*

(If $m < n^{1/2}$, then the data does not even reach the base, instead only moving down $\log_4(m)$ levels until the problem has been broken into squares of size 1.) The squares on the base have size $m/4^l$, so they take $\theta(m/4^l)$ time to compute their values. It now takes only $\theta(1)$ time per level to combine results and move them up. □

6.4. Optimality. In this section we show that some of our results are quite close to being optimal.

PROPOSITION 8. *In a pyramid computer of size n, the time needed to move $B \geqq 1$ bits of data from the first column of the base to the last column of the base is $\Omega(\log(n) + (B/\log(n))^{1/2})$.*

Proof. Assume $B \geqq \log_2(n)$, and let $L = \lfloor \log_4(n * \log_2(n)/B) \rfloor$ and $E = n^{1/2}/2^L$. For each column of PEs at level L, call the entire column and all of its descendents a *prism*. The data initially resides in the leftmost prism and must move to the rightmost one. If a bit only moves along communication links involving PEs at level L or below, then at least $E - 1$ communication links must be traversed, since there are E prisms, and each communication link either keeps the bit in the same prism or moves it to an adjacent one.

Figure 6 shows a side view of the PEs at level L and above. The usual way of drawing the pyramid has been slightly altered so that all PEs in the same column and level are represented by a single PE. The time spent traversing any vertically drawn wires (communication links) will be ignored. The weights along all other wires indicate the number of steps that could be saved by using the wire. There are $(E/2) * (E/2 - 1)$ horizontal wires labeled 1, $(E/4)^2 * 2$ slanted wires labeled 1, $(E/4) * (E/4 - 1)$ horizontal wires labeled 3, and so forth. Since each wire can carry at most $C * \log_2(n)$ bits per unit time, for some constant C, in 1 unit of time the maximum number of bits moved by the nonvertical wires above level L is

$$C * \log(n) * \left[\sum_{i=1}^{\log_4(n)-L} (2^i - 1)(E^2/4^i - E/2^i) + \sum_{i=1}^{\log_4(n)-L-1} 2(2^i - 1)(E^2/4^{i+1}) \right],$$

TECHNIQUES FOR THE PYRAMID COMPUTER   55
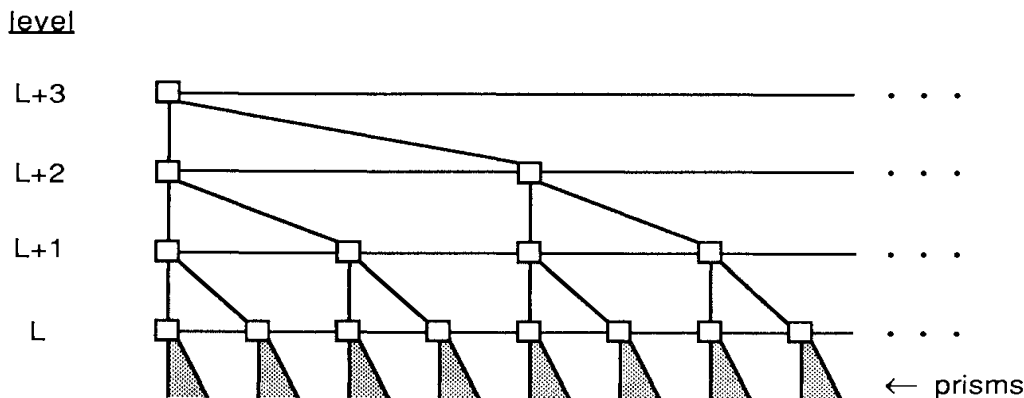
FIG. 6. *Another view of the pyramid computer.*

which is less than $C * E^2 * \log_2(n)$. Therefore, in t units of time the total savings is less than $C * E^2 * t * \log_2(n)$.

On the other hand, a bit of data that reaches the rightmost prism in t units of time must have crossed wires with a total weight of at least $E - 1 - t$. Furthermore, if all B bits of data reach the rightmost prism in t units of time, the total savings must be at least $B * (E - 1 - t)$. Therefore, t must be such that

$$B * (E - 1 - t) < C * E^2 * t * \log_2(n),$$

or

$$t > B * (E - 1)/(C * E^2 \log_2(n) + B) = \theta((B/\log(n))^{1/2}).$$

Since the pyramid computer of size n has a diameter (maximum distance between any two vertices) of $2 * \log_4(n)$, we also have $t \geq \log(n)$. Hence we have the desired result. $\square$

For each of the problems considered in this paper, it is easy to devise inputs to which Proposition 8 applies. For example, suppose one is performing component labeling of digitized pictures, and the input is known to be of the form in Fig. 7, where an X indicates a pixel which may or may not be black and a Y indicates a pixel that
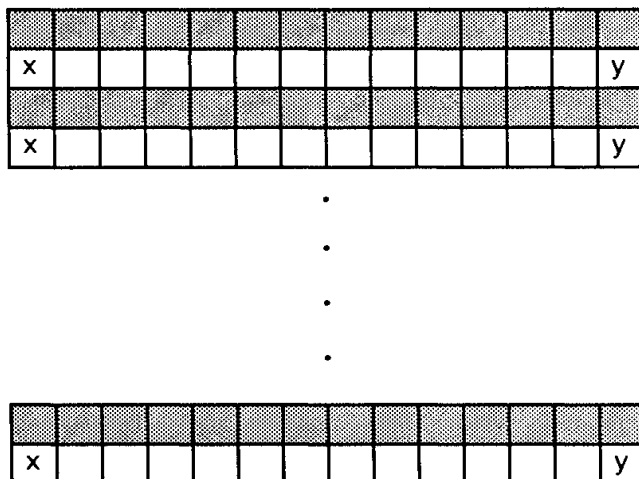
FIG. 7. *An image requiring extensive data movement.*

is always white. Notice that if the two black PEs neighboring a PE marked Y end up with the same label, then the PE marked X which is in Y's row must be black. Since a Y can determine if its black neighbors have the same label in $\theta(1)$ time after the component labeling algorithm is finished, the component labeling algorithm requires at least as much time as it takes to transmit $\theta(n^{1/2})$ bits from one edge to the opposite edge of the pyramid. By Proposition 8 this is $\Omega(n^{1/2}/\log^{1/2}(n))$. This lower bound is a factor of $\log^{1/2}(n)$ smaller than the time achieved in Theorem 5, showing that Theorem 5 is at most $\log^{1/2}(n)$ times optimal. We believe that Theorem 5, as well as Theorems 3, 4 and 6, and Corollary 2, are all optimal, and leave the proof of this as an open problem.

7. Extensions to other dimensions. Many of the data movement techniques that have been presented can be extended to pyramid computers of dimensions other than 2. Define a j-MCC *of size* n to be an SIMD machine in which n PEs are arranged in a j-dimensional cubic lattice. $PE(i(1), \cdots, i(j))$ is connected to $PE(k(1), \cdots, k(j))$, provided that $\sum_{l=1}^{j} |i(l) - k(l)| = 1$.

A j-PC *of size* n is an SIMD machine viewed as a full 2^j-ary tree with additional horizontal links. The base of the j-PC of size n is a j-MCC of size n. Each level of the pyramid is a j-MCC with $1/2^j$ as many PEs as the previous level. A PE at level i is connected to its neighbors (assuming they exist): $2*j$ adjacent PEs at level i, 2^j children at level $i-1$, and a parent at level $i+1$.

In our analyses of algorithms for j-MCCs and j-PCs, our times consider j fixed and n, m, i and S as the parameters. We do not determine the full dependence on j, e.g., whether each step really needs 2^j comparisons instead of a fixed number, since a PE in a $j-$MCC is fundamentally different from one in a k-MCC for $j \neq k$. In this convention, sorting can be performed on a j-MCC of size n in $\theta(n^{1/j})$ time [33]. Since the MCC RAR and the MCC RAW are implemented via sorting, we can

(1) Perform an RAR in $\theta(n^{1/j})$ time on a j-MCC of size n,

(2) Perform an RAW in $\theta(n^{1/j})$ time on a j-MCC of size n.

We can also extend some of the PC data movement techniques to the j-PC. We first consider the case $j > 1$, and discuss the case $j = 1$ separately at the end.

(1) j-PC *write*: Given that the destination level is a j-MCC of size k and the source PEs are $i-1$ levels below, a j-PC write ($j > 1$) can be performed in $\theta(\log(n) + [k*i^{j-1}]^{1/j})$ time.

(2) j-PC *read*: Given that the source level is a j-MCC of size k and the destination PEs are $i-1$ levels below, a j-PC read ($j > 1$) can be performed in $\theta(\log(n) + [k*i^{j-1}]^{1/j})$ time.

(3) *Count keys*: In a j-PC of size n ($j > 1$), if there are t different keys in the base then count_keys will finish in $\theta(\log(n) + [t + t*\log(n/t)^{j-1}]^{1/j})$ time.

(4) *Funnel read*: Assume that the last stage (j-dimensional) data cubes are of size S. (See the comments preceding Theorem 9 below for a discussion of the sizes of data cubes in a j-PC.) Then a j-PC funnel read ($j > 1$) can be performed in $\theta(S^{1/j})$ time.

(5) *Reducing a function*: Given sets Q and R, stored 1 per PE in a j-MCC of size m at level i, and given that g and $*$ can be computed in $\theta(1)$ time, in $\theta(m^{1/j} + m/2^{ij})$ time a j-PC ($j > 1$) can compute the reduction of g.

We have omitted a discussion of the pyramid matrix read and write since the mapping of a matrix into a j-MCC is not natural. Given the data movement techniques that do extend, we are able to adapt several of our algorithms to the j-PC:

THEOREM 7. *Given a j-dimensional pyramid computer of size n ($j > 1$), if the base contains the unsorted edges of an undirected graph with v vertices, then the extension of*

TECHNIQUES FOR THE PYRAMID COMPUTER   57

the component labeling algorithm in § 3.2 labels the components in $\theta(\log{(n)} + v^{1/j}[1 + \log{(n/v)^{j-1}}]^{1/j})$ time.

THEOREM 8. *Given a j-dimensional pyramid computer of size n $(j > 1)$, if the base contains the unsorted weighted edges of an undirected graph with v vertices, then the extension of the algorithm in § 3.3 finds a minimal spanning forest in $\theta(\log{(n)} + v^{1/j}[1 + \log{(n/v)^{j-1}}]^{1/j})$ time.*

COROLLARY 4. *Given a j-dimensional pyramid computer of size n $(j > 1)$, if the base contains the unsorted edges of an undirected graph G with v vertices, then in $\theta(\log{(n)} + v^{1/j}[1 + \log{(n/v)^{j-1}}]^{1/j})$ time one can*

    (a) *decide if G is bipartite,*

    (b) *determine the cyclic index of G,*

    (c) *find all bridge edges of G,*

    (d) *find all articulation points of G,*

    (e) *decide if G is biconnected.*

In the component label problem with digitized j-dimensional picture input, we again use stages of (j-dimensional) picture cubes and their associated data cubes, where the picture cubes increase in size by a factor of 2^j at each stage. We reduce a picture cube to an amount of data proportional to the cube's surface area, so the data squares at the final stage have $\theta(n^{(j-1)/j})$ PEs, and are on level $\theta((j-1) * \log_2{(n)})$. The extension of component labeling for such data is now straightforward.

THEOREM 9. *Given a digitized j-dimensional picture stored one pixel per PE at the base of a j-PC of size n $(j > 1)$, the components can be labeled in $\theta(n^{(j-1)/j^2})$ time.*

Despite our success in extending component labeling to the j-PC, we cannot do as well for the nearest neighbor problem. The difficulty is that, since the final data cubes have size $\theta(n^{(j-1)/j})$, when we try reducing a function it takes $\theta(n^{(j-2)/j})$ time for $j > 2$, which is no better than the edgelength of the base $(n^{1/j})$. Therefore our extension to a j-PC does not seem to be able to do any better than a j-MCC algorithm for the nearest neighbor problem.

Finally, a 1-PC behaves differently than a j-PC for $j > 1$. One important difference is that a 1-PC of size n can sort in $\theta(n/\log{(n)})$ time, versus $\theta(n)$ time for a 1-MCC of size n, while for $j > 1$ a j-PC and a j-MCC sort in the same time [25]. This sorting difference is most apparent when considering problems such as component labeling, finding a minimal spanning forest, deciding if a graph is bipartite, etc., for a graph with $\theta(n)$ vertices, given as unsorted edge input. For a j-PC, $j > 1$, these problems take $\theta(n^{1/j})$ time, but on a 1-PC they can be solved in $\theta(n/\log{(n)})$ time.

Another important difference is that one-dimensional digitized pictures are simplistic. Using the special properties of such input, there are easy 1-PC algorithms to do component labeling and finding nearest neighbors in $\theta(\log{(n)})$ time.

8. Conclusion. Because of its similarity to some animal optic systems, its similarity to the (region) quadtree structure, and its natural use in multiresolution image processing, the pyramid computer has long been suggested for low-level image processing [5], [6], [25], [29], [30], [32], [34], [35]. Due to advances in technology, some pyramid computers are currently under construction [8], [12], [23], [30], leading us to believe that they deserve further algorithm study. This paper begins that study by showing that the pyramid computer can be used for more complex tasks than originally considered. For instance, we have shown that the pyramid computer can be used to efficiently solve problems in higher-level image processing, graph theory, and digital geometry.

58   RUSS MILLER AND QUENTIN F. STOUT

A major contribution of this paper is the introduction of fundamental data movement operations for the pyramid computer to be used with a variety of standard input formats. These data movement operations are quite different from those used by earlier authors, in that most previous pyramid computer algorithms either concentrated soley on the child-parent links (adapting quadtree algorithms to the pyramid), or solely on the mesh-connected links of the base. In contrast, the data movement operations that we have presented intermingle the use of both types of links. They also make extensive use of intermediate levels of the pyramid to do calculations, store results and communicate data. Furthermore, we have shown how to use the base of the pyramid to aid in the computation of functions being performed at higher levels.

We have used our data movement operations to efficiently solve several geometric and graph-theoretic problems. Since there are numerous other problems in these and related fields which have divide-and-conquer solutions, the problem-solving techniques and data movement operations presented here should have a wide range of applications. For example, [15], [17] contain algorithms which use the pyramid read, pyramid write, and reducing a function operations to compute geometric properties such as convexity, diameter and smallest enclosing box.

It is interesting to compare the pyramid computer with other parallel architectures. Using the standard VLSI model in which processing elements are separated by at least unit distance and a wire has unit width, [5] has shown that a pyramid computer with a base of n processing elements can be laid out in $\theta(n)$ area by a simple modification of the standard "H tree" layout scheme. The space of a layout for an interconnection scheme is one measure of its cost, as is the regularity of the layout. A mesh-connected computer of n processing elements also requires only $\theta(n)$ with an extremely regular layout, but because it has a communication diameter of $\theta(n^{1/2})$ it requires $\Omega(n^{1/2})$ time to solve all of the problems considered here, compared with, say, the $\theta(n^{1/4})$ time needed by the pyramid computer to label the connected components of an image. (Mesh-connected computer algorithms taking $\theta(n^{1/2})$ time to solve problems presented in this paper appear in [19], [16].)

Another simple model that can be easily laid out in $\theta(n)$ area is the quadtree machine, which is simply a pyramid computer without the nearest neighbor links. Like the pyramid, the quadtree has a logarithmic communication diameter, but unlike the pyramid, the apex often acts as a bottleneck. For example, it is easy to show that the quadtree needs $\Omega(n^{1/2})$ time to label components or find nearest neighbors of an image, even if higher PEs have additional memory (as suggested in [1]). On the pyramid, we have used nearest neighbor connections at the intermediate levels to circumvent this bottleneck.

General-purpose interconnection schemes such as the shuffle-exchange, butterfly and cube-connected cycles can be used to provide poly-log time solutions to all the problems considered herein. (An algorithm is poly-log if it finishes in $P(\log^k (n))$ time for some constant k.) Unfortunately, these interconnection schemes require area that is nearly proportional to the square of that required to lay out the pyramid computer [36]. Although this extra area and complexity provide the capability of poly-log sorting, it is more than is needed for the problems considered here.

A more interesting model is the orthogonal trees or mesh of trees [36]. This model has a mesh-connected base of size n augmented so that each row and column of the base mesh has a binary tree over it, with these trees being disjoint except at their leaves. In this model $\theta(n^{1/2} \log^2 (n))$ bits can be moved from the leftmost $\log (n)$ columns to the rightmost $\log(n)$ columns in $\theta(\log (n))$ time. This is a significant improvement over the pyramid computer bound in Proposition 8, though not enough

TECHNIQUES FOR THE PYRAMID COMPUTER   59

to provide poly-log sorting. This machine model has not received much consideration as an image processing machine, but for all of the problems considered herein involving images or adjacency matrices, orthogonal trees can solve them in poly-log time.

Orthogonal trees do have some drawbacks, however. While the pyramid computer can be laid out in linear area, orthogonal trees need a factor of $\log^2(n)$ more area [36]. Further, orthogonal trees seem to have few ties to other objects of interest for researchers in image processing, as opposed to the neural, data structure, and multi-resolution ties mentioned above for the pyramid computer. It is because of these ties that the image processing community is building pyramids and not orthogonal trees. Additional models which are closer to the pyramids, and which solve all of the image processing problems considered herein in poly-log time, have recently been suggested [28]. These models were designed by starting with the pyramid computer and modifying it to be much faster on the algorithms presented here.

Acknowledgment. The authors would like to express their appreciation to the referees for their constructive comments.

REFERENCES

[1] N. AHUJA AND S. SWAMY, Multiprocessor pyramid architecture for bottom-up image analysis, IEEE Trans. PAMI, PAMI-6 (1984), pp. 463–474.

[2] M. J. ATALLAH AND S. R. KOSARAJU, Graph problems on a mesh-connected processor array, J. Assoc. Comput. Mach. 3 (1984), pp. 649–667.

[3] F. Y. CHIN, J. LAM AND I.-N. CHEN, Efficient parallel algorithms for some graph problems, Comm. Assoc. Comput. Mach., 25 (1982), pp. 659–665.

[4] C. R. DYER, A quadtree machine for parallel image processing, Tech. Report KSL 51, U. of Illinois at Chicago Circle, Chicago, IL, 1981.

[5] ———, A VLSI pyramid machine for hierarchical parallel image processing, Proc. PRIP, 1981, pp. 381–386.

[6] ———, Pyramid algorithms and machines, in Multicomputers and Image Processing Algorithms and Programs, K. Preston and L. Uhr, eds., Academic Press, New York, 1982, pp. 409–420.

[7] C. R. DYER AND A. ROSENFELD, Parallel image processing by memory augmented cellular automata, IEEE Trans. PAMI, PAMI-3 (1981), pp. 29–41.

[8] G. FRITSCH, W. KLEINOEDER, C. U. LINSTER AND J. VOLKERT, EMSY85—The Erlanger multi-processor system for a broad spectrum of applications, Proc. 1983 Internat. Conf. on Parallel Processing, 1983, pp. 325–330.

[9] S. E. HAMBRUSCH, VLSI algorithms for the connected component problem, this Journal, 12 (1983), pp. 354–365.

[10] S. E. HAMBRUSCH AND J. SIMON, Solving undirected graph problems on VLSI, Tech. Rep. CS-81-23, Computer Science, Penn. State Univ., State College, PA, 1981.

[11] D. S. HIRSCHBERG, A. K. CHANDRA AND D. V. SARWATE, Computing connected components on parallel computers, Comm. Assoc. Comput. Mach., 22 (1979), pp. 461–464.

[12] S. LEVIALDI, A pyramid project using integrated technology, in Integrated Technology for Parallel Image Processing, S. Levialdi, ed., Academic Press, New York, 1985, to appear.

[13] R. MILLER, Writing SIMD algorithms, Proc. 1985 Internat. Conference on Computer Design: VLSI in Computers, 1985, pp. 122–125.

[14] R. MILLER AND Q. F. STOUT, Computational geometry on a mesh-connected computer, Proc. 1984 Internat. Conf. on Parallel Processing, 1984, pp. 66–74.

[15] ———, Convexity algorithms for pyramid computers, Proc. 1984 Internat. Conf. on Parallel Processing, 1984, pp. 177–184.

[16] ———, Geometric algorithms for digitized pictures on a mesh-connected computer, IEEE Trans. PAMI, PAMI-7 (1985), pp. 216–228.

[17] ———, Pyramid computer algorithms for determining geometric properties of images, Proc. 1985 ACM Symposium on Computational Geometry, 1985, pp. 263–277.

[18] D. NASSIMI AND S. SAHNI, Finding connected components and connected ones on a mesh-connected parallel computer, this Journal, 9 (1980), pp. 744–757.

[19] ———, Data broadcasting in SIMD computers, IEEE Trans. Comput. C-30 (1981), pp. 101–107.

60   RUSS MILLER AND QUENTIN F. STOUT

[20] A. P. REEVES, *On efficient global feature extraction methods for parallel processing*, Comput. Graphics Image Processing, 14 (1980), pp. 159–169.

[21] B. SAKODA, *Parallel construction of polygonal boundaries from given vertices on a raster*, Tech. Rep. CS81 1–21, Computer Science, Penn. State Univ., State College, PA, 1981.

[22] C. SAVAGE AND J. JA'JA', *Fast, efficient parallel algorithms for some graph problems*, this Journal, 10 (1981), pp. 682–691.

[23] D. H. SCHAEFER et al., *The PMMP—a pyramid of MPP processing elements*, Proc. of the 18th Annual Hawaiian Internat. Conf. on Systems Science, 1, 1985, pp. 178–184.

[24] Q. F. STOUT, *Drawing straight lines with a pyramid cellular automaton*, Inform. Proc. Lett., 15 (1982), pp. 233–237.

[25] ———, *Sorting, merging, selecting, and filtering on tree and pyramid machines*, Proc. 1983 Internat. Conf. on Parallel Processing, 1983, pp. 214–221.

[26] ———, *Pyramid computer solutions of the closest pair problem*, J. Algorithms, 6 (1985), pp. 200–212.

[27] ———, *Tree-based graph algorithms for some parallel computers*, Proc. 1985 Internat. Conf. on Parallel Processing, 1985, pp. 727–730.

[28] ———, *Mesh and pyramid computers inspired by geometric algorithms*, Proc. Workshop on Algorithm-Guided Parallel Architectures for Automatic Target Recognition, 1985, pp. 293–315.

[29] S. L. TANIMOTO, *Programming techniques for hierarchical parallel image processors*, in Multicomputers and Image Processing Algorithms and Programs, K. Preston and L. Uhr, eds., Academic Press, New York, 1982, pp. 421–429.

[30] ———, *Towards hierarchical cellular logic: Design considerations for pyramid machines*, Tech. Rep. 81–02–01, Computer Science, Univ. Washington, Seattle, WA, 1981.

[31] ———, *Sorting, histogramming, and other statistical operations on a pyramid machine*, Tech. Rep. 82–08–02, Computer Science, Univ. Washington, Seattle, WA, 1982.

[32] S. L. TANIMOTO AND A. KLINGER, *Structured Computer Vision: Machine Perception through Hierarchical Computation Structures*, Academic Press, New York, 1980.

[33] C. D. THOMPSON AND H. T. KUNG, *Sorting on a mesh-connected parallel computer*, Comm. Assoc. Comput. Mach., 20 (1977), pp. 263–271.

[34] L. UHR, *Layered 'Recognition Cone' networks that preprocess, classify, and describe*, IEEE Trans. Comput., C-21 (1972), pp. 758–768.

[35] ———, *Algorithm-Structured Computer Arrays and Networks*, Academic Press, New York, 1984.

[36] J. D. ULLMAN, *Computational Aspects of VLSI*, Computer Science Press, Baltimore, MD, 1984.

[37] F. L. VAN SOY, *The parallel recognition of classes of graphs*, IEEE Trans. Comput., C-29 (1980), pp. 563–570.

McGraw-Hill Series in Computer Science

SENIOR CONSULTING EDITOR

C. L. Liu, University of Illinois at Urbana-Champaign

CONSULTING EDITOR

Allen B. Tucker, Bowdoin College

Fundamentals of Computing and Programming
Computer Organization and Architecture
Systems and Languages
Theoretical Foundations
Software Engineering and Database
Artificial Intelligence
Networks, Parallel and Distributed Computing
Graphics and Visualization
The MIT Electrical Engineering and Computer Science Series

Networks, Parallel and Distributed Computing

Ahuja: Design and Analysis of Computer Communication Networks
Filman and Friedman: Coordinated Computing: Tools and Techniques for Distributed
  Software
Hwang: Advanced Computer Architecture: Parallelism, Scalability, Programmability
Keiser: Local Area Networks
Kershenbaum: Telecommunications Network Design Algorithms
Lakshmivarahan and Dhall: Analysis and Design of Parallel Computers
Quinn: Parallel Computing: Theory and Practice

PARALLEL COMPUTING

THEORY AND PRACTICE

SECOND EDITION

Michael J. Quinn

Oregon State University

Figure Credits

Figures 1-2 and 1-27: Reprinted from *Computer Architecture: A Quantitative Approach* by John L. Hennessy and David A. Patterson, © 1990, with the permission of the publisher, Morgan Kaufmann Publishers, Inc.

Figure 2-1: Aho, Hopcroft, and Ullman, *The Design and Analysis of Computer Algorithms,* © 1974, Addison-Wesley, Reading, Massachusetts, page 5, Figure 1.3 Reprinted with permission.

Figure 3-5: Reprinted from *Computational Aspects of VLSI* by Jeffrey D. Ullman, © 1984, with the permission of the publisher, Computer Science Press, Inc., 1803 Research Blvd., Rockville, MD 20850 USA.

Figures 3-16, 3-21, 4-1, 4-8, 4-12, 10-32, and 10-33: Reprinted from *Data-Parallel Programming on MIMD Computers* by Philip J. Hatcher and Michael J. Quinn, © 1991, with the permission of the publisher, The MIT Press.

Figure 8-6: Reprinted from *Introduction to Algorithms* by Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest, © 1990, with the permission of the publisher, McGraw-Hill, Inc.

Figure 9-1: David Halliday and Robert Resnick, *Fundamentals of Physics, Revised Printing,* © 1974, John Wiley & Sons, Inc. Reprinted by permission of John Wiley & Sons, Inc.

Figure 10-15: Robert Sedgewick, *Algorithms,* © 1983, Addison-Wesley, Reading, Massachusetts, page 465 (figure). Reprinted with permission.

Figure 10-18: Donald E. Knuth, *The Art of Computer Programming, Volume 3, Sorting and Searching,* © 1973, Addison-Wesley, Reading, Massachusetts, page 237, Figure 36. Reprinted with permission.

Quotation at beginning of Chapter 6: *Zorba the Greek,* COPYRIGHT © 1953 by Simon & Schuster. © Renewed by Simon & Schuster. Reprinted by permission from Simon & Schuster, Inc.

PARALLEL COMPUTING
Theory and Practice

1234567890 DOH   DOH 909876543

ABOUT
THE AUTHOR

MICHAEL J. QUINN is an associate professor of computer science at Oregon State University. He received his Ph.D. in computer science from Washington State University. He has also taught at the University of New Hampshire, and he worked for two years at Tektronix, Inc. as a software engineer before earning his doctorate. He is author of *Designing Efficient Algorithms for Parallel Computers* (McGraw-Hill, 1987) and co-author (with Philip J. Hatcher) of *Data-Parallel Programming on MIMD Computers* (The MIT Press, 1991). Dr. Quinn has published dozens of technical papers in the areas of parallel algorithms and data-parallel programming environments. Currently he is editor-in-chief of *IEEE Parallel and Distributed Technology: Systems and Applications* magazine.

CONTENTS

x   CONTENTS

CONTENTS   xi

xii   CONTENTS

xiv   CONTENTS

3

PROCESSOR ARRAYS, MULTIPROCESSORS, AND MULTICOMPUTERS

E pluribus unum (Out of many, one)

From the Great Seal of the United States of America

The goal of this chapter is to introduce three important models of parallel computation and several associated parallel computer designs. The models are processor arrays, multiprocessors, and multicomputers, all of which have fostered actual parallel computers.

We present a number of processor organizations in Sec. 3.1; these are mesh, binary tree, hypertree, pyramid, butterfly, hypercube, cube-connected cycles, shuffle-exchange, and de Bruijn. These processor organizations are evaluated according to criteria that help determine their practicality and versatility. Section 3.2 surveys a number of processor array models including the Connection Machine CM-200™, a well-known processor array. Section 3.3 discusses multiprocessors, multiple-CPU computers with global address space. We describe two commercial multiprocessors—the Sequent Symmetry™ and the BBN Butterfly TC2000™. In Sec. 3.4 we examine multicomputers, multiple-CPU computers with no global address space. Our example architectures are the nCUBE 2™, Thinking Machines' CM-5™, and the Intel Paragon XP/S™. Section 3.5 presents Flynn's taxonomy, the most common classification of sequential and parallel computer architectures. In Sec. 3.6 we discuss the terms used to describe the performance of parallel algorithms. These terms include speedup, scaled speedup, and parallelizability.

88

CHAPTER 3:   PROCESSOR ARRAYS, MULTIPROCESSORS, AND MULTICOMPUTERS   53

3.1   PROCESSOR ORGANIZATIONS

This section defines nine important processor organizations—methods of connecting processors in a parallel computer. A processor organization can be represented by a graph in which the nodes (vertices) represent processors and the edges represent communication paths between pairs of processors. (For readers unfamiliar with graph theory, a brief introduction to graph theoretic terms is given in Appendix A.) We evaluate these processor organizations according to criteria that help us understand their effectiveness in implementing efficient parallel algorithms on real hardware. These criteria are:

1 Diameter. The diameter of a network is the largest distance between two nodes. Low diameter is better, because the diameter puts a lower bound on the complexity of parallel algorithms requiring communication between arbitrary pairs of nodes.

2 Bisection width of the network. The bisection width of a network is the minimum number of edges that must be removed in order to divide the network into two halves (within one). High bisection width is better, because in algorithms requiring large amounts of data movement, the size of the data set divided by the bisection width puts a lower bound on the complexity of the parallel algorithm.

3 Number of edges per node. It is best if the number of edges per node is a constant independent of the network size, because then the processor organization scales more easily to systems with large numbers of nodes.

4 Maximum edge length. For scalability reasons it is best if the nodes and edges of the network can be laid out in three-dimensional space so that the maximum edge length is a constant independent of the network size.

We now discuss the various type of processor organizations.

3.1.1   Mesh Networks

In a mesh network, the nodes are arranged into a q-dimensional lattice. Communication is allowed only between neighboring nodes; hence interior nodes communicate with $2q$ other processors. Figure 3-1a illustrates a two-dimensional (2-D) mesh. Some variants of the mesh model allow wrap-around connections between processors on the edge of the mesh. These connections can connect processors in the same row or column (Fig. 3-1b) or adjacent rows or columns (Fig. 3-1c).

Let's evaluate the mesh network according to our four criteria. We assume that the mesh has no wrap-around connections. The diameter of a q-dimensional mesh with k^q nodes is $q(k - 1)$. Hence, from a theoretical point of view, mesh networks have the disadvantage that data routing requirements often prevent the

54   PARALLEL COMPUTING: THEORY AND PRACTICE
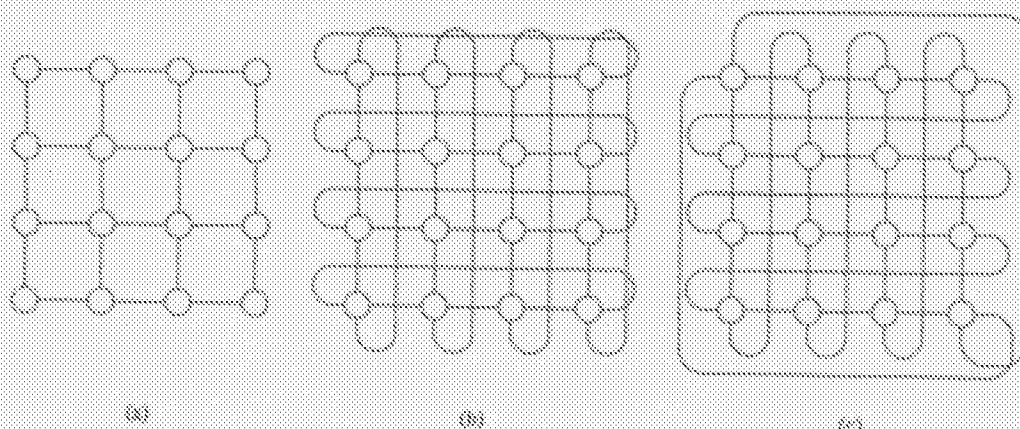
FIGURE 3-1   Two-dimensional meshes. (a) Mesh with no wrap-around connections. (b) Mesh with wrap-around connections between processors in same row or column. (c) Mesh with wrap-around connections between processors in adjacent rows or columns.

development of polylogarithmic time parallel algorithms. In practice, however, some computer architects would rather implement fewer, faster links than more, slower links.

The bisection width of a q-dimensional mesh with k^q nodes is k^{q-1}. The maximum number of edges per node is $2q$. The maximum edge length is a constant, independent of the number of nodes, for two- and three-dimensional meshes.

The two-dimensional mesh has been a popular topology for processor arrays, including Goodyear Aerospace's MPP™, the AMT DAP™, and MasPar's MP-1™. The Intel Paragon XP/S multicomputer connects processors with a two-dimensional mesh.

3.1.2 Binary Tree Networks

In a binary tree network the $2^k - 1$ nodes are arranged into a complete binary tree of depth $k - 1$ (Fig. 3-2). A node has at most three links. Every interior node can communicate with its two children and every node other than the root
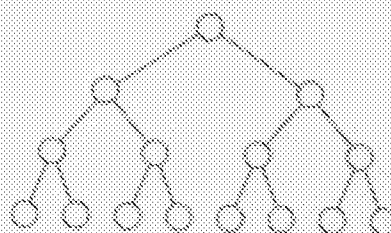
FIGURE 3-2   Binary tree network of size 15 and depth 3.

CHAPTER 3: PROCESSOR ARRAYS, MULTIPROCESSORS, AND MULTICOMPUTERS   85

can communicate with its parent. The binary tree has low diameter, $2(k-1)$, but has a poor bisection width of one. Assuming nodes have volume, it is impossible to arrange the nodes of a binary tree in three-dimensional space such that as the number of nodes increases, the length of the longest edge is always less than a specified constant.

3.1.3 Hypertree Networks

A hypertree represents one approach to building a network with the low diameter of a binary tree but an improved bisection width. The easiest way to think of a hypertree network of degree k and depth d is to consider the network from two different angles (Fig. 3-3). From the front a hypertree network of degree k and depth d looks like a complete k-ary tree of height d (Fig. 3-3a). From the side, the same hypertree network looks like an upside down binary tree of height d (Fig. 3-3b). Joining the front and side views yields the complete network. Figure 3-3c illustrates a hypertree network of degree 4 and height 2.

A 4-ary hypertree with depth d has 4^d leaves and $2^d(2^{d+1}-1)$ nodes in all. The diameter of this network is $2d$ and its bisection width is 2^{d+1}. The number of edges per node is never more than six and the maximum edge length is an increasing function of the problem size.

FIGURE 3-3   Hypertree network of degree 4 and depth 2. (a) Front view. (b) Side view. (c) Complete network.

50   PARALLEL COMPUTING: THEORY AND PRACTICE
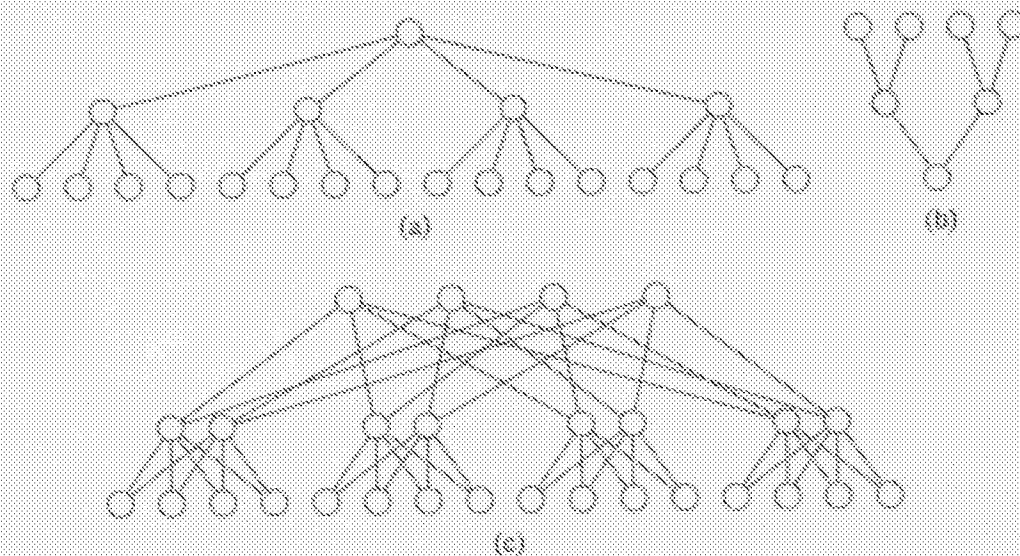
The data routing network of the Connection Machine CM-5 multicomputer is a 4-ary hypertree.

3.1.4   Pyramid Networks

The pyramid network can be seen as an attempt to combine the advantages of mesh networks with those of tree networks. A pyramid network of size k^2 is a complete 4-ary rooted tree of height $\log_2 k$ augmented with additional interprocessor links so that the processors in every tree level form a 2-D mesh network (Miller and Stout 1987). A pyramid of size k^2 has at its base a 2-D mesh network containing k^2 processors. The total number of processors in a pyramid of size k^2 is $(4/3)k^2 - (1/3)$. The levels of the pyramid are numbered in ascending order such that the base has level number 0, and the single processor at the apex of the pyramid has level number $\log_2 k$. Every interior processor is connected to nine other processors: one parent, four mesh neighbors, and four children. Figure 3-4 illustrates a pyramid network of size 16.

The advantage of the pyramid over the 2-D mesh is that the pyramid reduces the diameter of the network. For example, when a message must travel from one side of the mesh to the other, fewer link traversals are required if the message travels up and down the tree rather than across the mesh. The diameter of a pyramid of size k^2 is $2 \log k$.

The addition of tree links does not give the pyramid a significantly higher bisection width than a 2-D mesh. The bisection width of a pyramid of size k^2 is $2k$.

The maximum number of links per node is no greater than nine, regardless of the size of the network. Unlike a 2-D mesh, however, the length of the longest edge in the pyramid network is an increasing function of the network size.
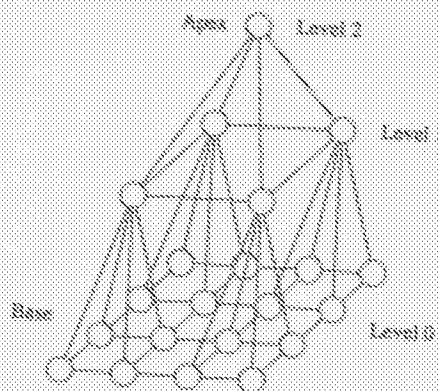
FIGURE 3-4   A pyramid network of size 16.

CHAPTER 3: PROCESSOR ARRAYS, MULTIPROCESSORS, AND MULTICOMPUTERS   97

3.1.5 Butterfly Network

A butterfly network consists of $(k + 1)2^k$ nodes divided into $k + 1$ rows, or ranks, each containing $n = 2^k$ nodes (Fig. 3-5). The ranks are labeled 0 through k, although the ranks 0 and k are sometimes combined, giving each node four connections to other nodes.

Let node(i, j) refer to the jth node on the ith rank, where $0 \leq i \leq k$ and $0 \leq j < n$. Then node(i, j) on rank $i > 0$ is connected to two nodes on rank $i - 1$, node$(i - 1, j)$ and node$(i - 1, m)$, where m is the integer found by inverting the ith most significant bit in the binary representation of j. Note that if node(i, j) is connected to node$(i - 1, m)$, then node(i, m) is connected to node$(i - 1, j)$. The entire network is made up of such "butterfly" patterns, hence the name. As the rank numbers decrease, the widths of the wings of the butterflies increase exponentially. For this reason the length of the longest network edge increases as the number of network nodes increases.

The diameter of a butterfly network with $(k + 1)2^k$ nodes is $2k$ and the bisection width of a network of that size is 2^{k-1}.

A butterfly network serves to route data from nonlocal memory to processors on the BBN TC2000 multiprocessor described later in this chapter.

3.1.6 Hypercube (Cube-Connected) Networks

A cube-connected network, also called a binary n-cube network, is a butterfly with its columns collapsed into single nodes. Formally, this network consists of 2^k nodes forming a k-dimensional hypercube. The nodes are labeled $0, 1, \ldots, 2^k - 1$; two nodes are adjacent if their labels differ in exactly one bit position. A four-dimensional hypercube is shown in Fig. 3-6.

The diameter of a hypercube with 2^k nodes is k, and the bisection width of that size network is 2^{k-1}, the hypercube organization has low diameter and high bisection width at the expense of the number of edges per node and the
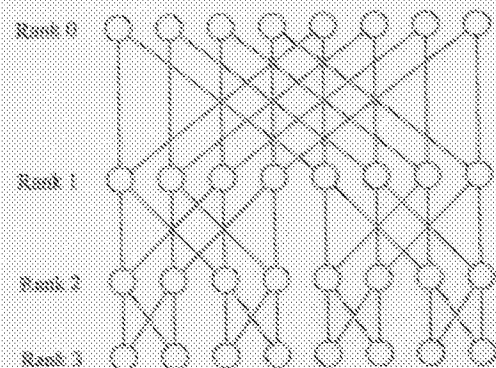
FIGURE 3-5 Butterfly network with 32 nodes. (Ullman [1984]).
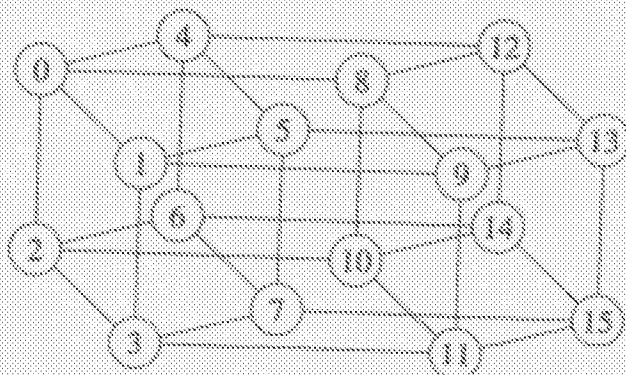
FIGURE 3-6   A four-dimensional (16-node) hypercube.

length of the longest edge. The number of edges per node is k—the logarithm of the number of nodes in the network. The length of the longest edge in a hypercube network increases as the number of nodes in the network increases.

The hypercube was the most popular processor organization for first- and second-generation multicomputers, and nCUBE continues to build systems based on this topology. In addition, processing element clusters on the Connection Machine CM-200 processor array are connected in a hypercube.

3.1.7   Cube-Connected Cycles Networks

The cube-connected cycles network is a k-dimensional hypercube whose 2^k "vertices" are actually cycles of k nodes formed by the columns of a butterfly network whose ranks 0 and k have been combined. For each dimension, every cycle has a node connected to a node in the neighboring cycle in that dimension. See Fig. 3-7 for a drawing of a 24-node cube-connected cycles network.

Formally, node(i, j) is connected to node(i, m) if and only if m is the result of inverting the ith most significant bit of the binary representation of j. Note that the connections are slightly different from those in the butterfly network, that is, if node(i, j) is connected to node($i - 1$, m) in the butterfly network, where $j \neq m$, then node(i, j) is connected to node(i, m) in the cube-connected cycles network. However, in the cube-connected cycles network, node(i, j) can still communicate with node($i - 1$, m) by following two links, since there is a direct path from node(i, m) to node($i - 1$, m).

Compared to the hypercube, the cube-connected cycles processor organization has the advantage that the number of edges per node is three—a constant independent of network size. However, the cube-connected cycles network has the disadvantage that the network diameter is twice that of a hypercube and the bisection width is lower. Given a cube-connected cycles network of size $k2^k$, its diameter is $2k$ and its bisection width is 2^{k-1}.

CHAPTER 3: PROCESSOR ARRAYS, MULTIPROCESSORS, AND MULTICOMPUTERS 59
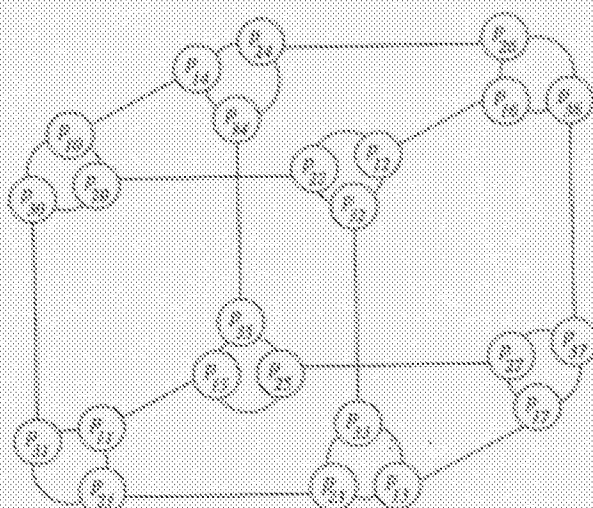
FIGURE 3-7 Cube-connected cycles network with 24 nodes. The first subscript of each node denotes the rank; the second subscript of each node denotes the column.

3.1.8 Shuffle-Exchange Networks

A shuffle-exchange network consists of $n = 2^k$ nodes, numbered $0, 1, \ldots, n - 1$, and two kinds of connections, called shuffle and exchange. Exchange connections link pairs of nodes whose numbers differ in their least significant bit. The perfect shuffle connection links node i with node $2i$ modulo $(n - 1)$, with the exception that node $n - 1$ is connected to itself. See Fig. 3-8 for a drawing of an eight-node shuffle-exchange network. Shuffle connections are indicated by the solid arrows and the exchange links are represented by the dashed arrows.

To understand the derivation of the name *perfect shuffle*, consider shuffling a deck of eight cards, numbered 0, 1, 2, 3, 4, 5, 6, 7. If the deck is divided into two exact halves and shuffled perfectly, then the result is the following order: 0, 4, 1, 5, 2, 6, 3, 7. Reexamine Fig. 3-8 and notice that the final position of the card that began at index i can be determined by following the shuffle link from node i.

Let $a_{n-1}a_{n-2} \cdots a_1 a_0$ be the address of a node in a perfect shuffle network, expressed in binary. A datum at this address will be at address $a_{n-2} \cdots a_1 a_0 a_{n-1}$

FIGURE 3-8 Shuffle-exchange network with eight nodes. Solid arrows denote shuffle connections. Dashed arrows denote exchange connections.

60   PARALLEL COMPUTING: THEORY AND PRACTICE
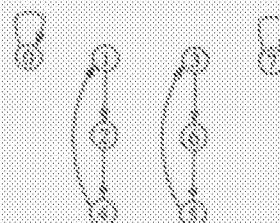
FIGURE 3-9   Necklaces of the shuffle-exchange network with eight nodes.

following a shuffle operation. In other words, the change in the address of a piece of data after a shuffle operation corresponds to a left cyclic rotation of the address by 1 bit (Prob. 3.4). If $n = 2^k$, then k shuffling operations move a datum back to its original location. The nodes through which a data item beginning at address i travels in response to a sequence of shuffles are called the necklace of i. No necklace is longer than k and a necklace shorter than k is called a short necklace. Figure 3-9 illustrates the necklaces of the perfect shuffle network with eight nodes.

Every node in a shuffle-exchange network has two outgoing and two incoming links. The length of the longest link increases as a function of network size. The number of long links has advantages with respect to network diameter and bisection width since the diameter of a shuffle exchange network is logarithmic in the number of nodes, that is, a network with 2^k nodes has diameter $2k - 1$. The bisection width is at least $2^{k-1}/k$.

Siegel (1979) has shown that a composition of k shuffle-exchange networks, called an omega network, is equivalent to a hypercube network with degree k. The same effect can be achieved by building only one stage of the network and cycling through it k times (Lawrie 1975).

3.1.9 de Bruijn networks

A de Bruijn network consists of $n = 2^k$ nodes. Let $a_{k-1}a_{k-2}\cdots a_1a_0$ be the address of a node in the de Bruijn network. The two nodes reachable via directed edges from that node are

$a_{k-2}a_{k-3}\cdots a_1a_00$
$a_{k-2}a_{k-3}\cdots a_1a_01$

Figure 3-10 illustrates an eight-processor de Bruijn network.

FIGURE 3-10   An 8-processor de Bruijn network.

CHAPTER 3: PROCESSOR ARRAYS, MULTIPROCESSORS, AND MULTICOMPUTERS   61
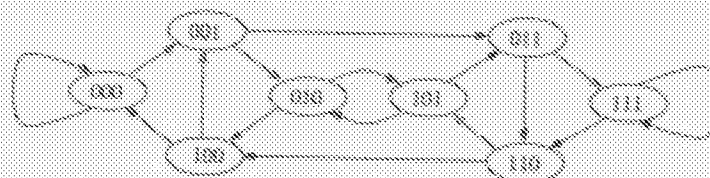
TABLE 3-1   CHARACTERISTICS OF VARIOUS PROCESSOR ORGANIZATIONS

Network	Nodes	Diameter	Bisection Width	Constant Number of Edges	Constant Edge Length
1-D mesh	k	$k-1$	1	Yes	Yes
2-D mesh	k^2	$2(k-1)$	k	Yes	Yes
3-D mesh	k^3	$3(k-1)$	k^2	Yes	Yes
Binary tree	$2^k - 1$	$2(k-1)$	1	Yes	No
4-ary hypertree	$2^k(2^{k+1} - 1)$	$2k$	2^{k+1}	Yes	No
Pyramid	$(4k^2 - 1)/3$	$2\log k$	$2k$	Yes	No
Butterfly	$(k + 1)2^k$	$2k$	2^k	Yes	No
Hypercube	2^k	k	2^{k-1}	No	No
Cube-connected cycles	$k2^k$	$2k$	2^{k-1}	Yes	No
Shuffle-exchange	2^k	$2k-1$	$\geq 2^{k-1}/k$	Yes	No
de Bruijn	2^k	k	$2^k/k$	Yes	No

The number of edges per node is a constant independent of the network size. The bisection width of a network with 2^k nodes is $2^k/k$, and the length of the longest edge increases with the size of the network. As with shuffle-exchange networks, de Bruijn networks contain shuffle connections.

The diameter of a de Bruijn network with 2^k nodes is k, which is about half the diameter of a shuffle-exchange network with the same number of nodes.

The processors of the Triton/1™, a SIMD/MIMD parallel computer developed at the University of Karlsruhe, are connected with a de Bruijn network (Hermer et al. 1992).

3.1.10   Processor Organization Summary

Table 3.1 summarizes the characteristics of the processor organizations we have considered. Of the nine organizations, only the mesh has constant edge length. The hypercube is noteworthy as the only processor organization we have considered in which the number of edges per node is an increasing function of the network size.

3.2   PROCESSOR ARRAYS

A vector computer is a computer whose instruction set includes operations on vectors as well as scalars. Generally, there are two ways of implementing a vector computer. A pipelined vector processor streams vectors from memory to the CPU, where pipelined arithmetic units manipulate them. The Cray-1™ and Cyber-205™ are well known pipelined vector processors. We do not consider these architectures further.

This text provides an exceptional introduction to parallel computing by balancing theory and practice. The emphasis is on designing, analyzing and implementing parallel algorithms suitable for execution on real parallel computers. Early chapters set the stage by introducing key concepts, illustrating fundamental parallel algorithms, and describing ways to incorporate high-level parallelism into hardware and software. Later chapters explore the development of parallel algorithms for matrix multiplication, the fast Fourier transform, solving linear systems, sorting, searching, graph theoretic problems, and combinatorial search. Numerous graphs illustrate the speedups that can be achieved on actual parallel hardware by implementing the parallel algorithms developed in the text. As a result, students learn how to make efficient use of emerging parallel computer technology.

A number of additional features make this book distinctive:

It surveys some of the most popular parallel computer architectures, including Thinking Machines' CM-5 (TM), Intel's Paragon XP/S (TM), and the Sequent Symmetry (TM)

* It covers some of the most popular parallel programming languages, including Fortran 90, C*, Linda and OCCAM

* A glossary of parallel computing terminology contains all the terms defined in the text

* More than 200 exercises cover the gamut from proofs to programming assignments

* The exceptionally large bibliography provides ample opportunities for further study

* Library of Congress call numbers simplify access to frequently cited journals and proceedings

Also from McGraw-Hill:

Hwang, *Advanced Computer Architecture: Parallelism, Scalability, Programmability,* 1993.

ISBN 0-07-051294-9
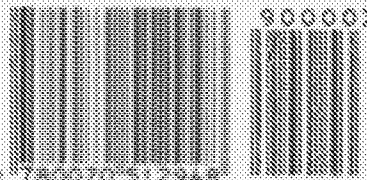
Electronic Acknowledgement Receipt

EFS ID:	34485963
Application Number:	16202067
International Application Number:	
Confirmation Number:	8134
Title of Invention:	VLSI Layouts of Fully Connected Generalized Networks
First Named Inventor/Applicant Name:	Venkat Konda
Customer Number:	38139
Filer:	Venkat Konda
Filer Authorized By:	
Attorney Docket Number:	V-0070US
Receipt Date:	04-DEC-2018
Filing Date:	27-NOV-2018
Time Stamp:	20:49:27
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Information Disclosure Statement (IDS) Form (SB08)	16-202067-sb0008b.pdf	388848 3b121adaeb17903bf8a086d899fbc7f0438b9090	no	2

Warnings:

Information:					
This is not an USPTO supplied IDS fillable form					
2	Non Patent Literature	Chang-Melhelm97.pdf	106122 5183c0b2ba11fa5a3ca23e20e040ceb18e60a2fb	no	6
Warnings:					
Information:					
3	Non Patent Literature	Clos1953.pdf	7975907 d6a9c66d332e4c29888ccde6179427c0ba887aa9	no	19
Warnings:					
Information:					
4	Non Patent Literature	EL-SAYED-YOUSSEF97.PDF	354218 4067f42e02f9e1d529580727ac1dd5d59a9aef39	no	5
Warnings:					
Information:					
5	Non Patent Literature	fixedws_fpga99.pdf	1537050 ed8daa940d6fb2f63f9231280b1ee7e181653f8b	no	10
Warnings:					
Information:					
6	Non Patent Literature	Goodman-Sequin1981.pdf	3211686 d6f9f7eaab18a0e7ed42e01b567354e2bacbab38	no	39
Warnings:					
Information:					
7	Non Patent Literature	Greenberg1994.pdf	12957830 9e92f9830a8848dcde5fe6f13cf9b346da84f167	no	12
Warnings:					
Information:					
8	Non Patent Literature	lemieux-fpga2001.pdf	126750 07b0d11059ea3429b3edb856862db06303bceb9c	no	10
Warnings:					
Information:					

9	Non Patent Literature	Manuel07.pdf	324448	no	7
			28c24822eaa8b56ffa3fa1e02e3246e229c28ac1		

Warnings:

Information:

10	Non Patent Literature	Miller-Stout1987.pdf	3124080	no	23
			83ca3b770a7eaffe22b5897c91b352aecfbb8132		

Warnings:

Information:

11	Non Patent Literature	Parallel-Computing-MQuinn-1994.pdf	3499947	no	22
			49bc645440cbb88779a58e37a350e398a13c9ef9		

Warnings:

Information:

Total Files Size (in bytes):	33606886

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.
National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.
New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

Substitute for form 1449/PTO **INFORMATION DISCLOSURE STATEMENT BY APPLICANT** *(Use as many sheets as necessary)*	*Complete if Known*	
	Application Number	16/202067
	Filing Date	12-4-2018
	First Named Inventor	Venkat Konda
	Art Unit	
	Examiner Name	
Sheet 1 of 1	Attorney Docket Number	V-0070US

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
	1	A. DeHon, "Unifying Mesh- and Tree-Based Programmable Interconnect," IEEE Trans. on Very Large Scale Int. Systems, vol. 12, no. 10, pp. 1051-1065, Oct. 2004	
	2	Guy Lemieux and David Lewis. Analtyical framework for switch block design. In Intl. Conference on Field Programmable Logic and Applications, pages 122-131, September 2002.	
	3	Chen, G; Lau, FCM, "A tight layout of the cube-connected cycles", The 4th International Conference on High Perf. Computing, Bangalore, India, 18-21 December 1997, p. 422-427	
	4	Michael Shyu, Yu-Dong Chang, Guang-Ming Wu, and Yao-Wen Chang, "Generic universal switch blocks. IEEE Transactions on Computers,49(4):348-359, April 2000.	
	5	Y. Yamada, et. al. , ```Folded Fat H-Tree: an interconnection topology for Dynamically Reconfigurable Processor Array", Embed and Ubiq. Cmpting, Intl Conf. EUC 2004.	
	6	A. DeHon. Balancing Interconnect and Computation in a Reconfigurable Computing Array (or, why you don't really want 100% LUT utili). In Proc. of intl. symp. on FPGAs, Feb 99.	
	7	André DeHon. Compact, Multilayer Layout for Butterfly Fat-Tree. In Twelfth Annual ACM Symposium on Parallel Algs and Architectures (SPAA 2000), pages 206--215, July 9-12, 2000	
	8	V. P. Roychdowdhury et. al., "Segmented Channel Routing," IEEE Trans on Computer-Aided Design of Integrated Circuits and Systems, Vol. 12, No. 1, pp. 79-95, January 1993.	

Examiner Signature		Date Considered	

Privacy Act Statement

The **Privacy Act of 1974 (P.L. 93-579)** requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (*i.e.*, GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

Tighter Layouts of the Cube-Connected Cycles

Guihai Chen and Francis C.M. Lau*

Department of Computer Science

The University of Hong Kong

{gchen,fcmlau}@cs.hku.hk

May 1997

Abstract

Preparata and Vuillemin proposed the cube-connected cycles ($\mathcal{CCC}$) and its compact layout in 1981 [16]. We give a new layout scheme for the $\mathcal{CCC}$ which uses less than half the area of the Preparata-Vuillemin layout. We also give a lower bound on the layout area of the $\mathcal{CCC}$. The area of the new layout deviates from the bound by a small constant factor. We then consider the unfolded $\mathcal{CCC}$, called the *cube-connected lines* ($\mathcal{CCL}$), for which we give a compact layout. This latter layout is tight when compared with the lower bound above which is also applicable to the $\mathcal{CCL}$.

Keywords: Interconnection networks, cube-connected cycles, VLSI, embedding, routing, layout.

1   Introduction

Interconnection network is an important design element in the construction of parallel computers. Many issues are involved in deciding on a specific topology for connecting a set of processors. With the rapid technological progresses in VLSI, it is reasonable to conceive of a huge number of processors being integrated tightly together to solve problems in a cooperative, parallel fashion. Therefore, one of the criteria to judge the suitability of an interconnection network for the implementation of parallel computers is whether the network can be laid out compactly in a VLSI grid.

The cube-connected cycles ($\mathcal{CCC}$) is one of the most popular interconnection networks, which was proposed by Preparata and Vuillemin [16] as a substitute for the hypercube in 1981. In the same paper, they gave an asymptotically-optimal layout scheme for the $\mathcal{CCC}$. Their layout scheme, however, cannot produce the minimal layout for the $\mathcal{CCC}$. Our work aims at finding better layout schemes for the $\mathcal{CCC}$. Research in the fields of graph embedding and VLSI layout has developed

* Correspondence: F.C.M. Lau, Department of Computer Science, The University of Hong Kong, Hong Kong. Email: fcmlau@cs.hku.hk / Fax: (+852) 2559 8447.

1

powerful techniques [2, 5] that can produce embeddings and layouts which are quite efficient—often within a constant factor from the optimal. However, even a modest constant factor may render an asymptotically-optimal layout or embedding unacceptable for real implementation. It is necessary to try to achieve the minimal. This is the motivation behind our work.

Our project has two goals: (1) to give a more compact layout for the CCC than the Preparata-Vuillemin layout, and (2) to reduce the long wires of the layout while keeping the asymptotically-optimal area. We have achieved the first goal—a new layout scheme which uses less than half the area of the Preparata-Vuillemin layout. Section 2 reviews the Thompson Model and the Preparata-Vuillemin layout. Section 3 presents the new layout and compares it with the Preparata-Vuillemin layout. Section 4 gives a lower-bound on the layout area, which is met (save some low-order terms) by our layout of the CCL as given in Section 5.

2   Preliminaries

2.1   Thompson Model

Among the many mathematical models that have been proposed for VLSI computations, the most widely accepted one is due to Thompson [17, 18]. In his model, the chip is presumed to consist of a grid of vertical and horizontal tracks which are spaced apart at unit intervals. Two layers of interconnect are used to route the wires. Vertical wires are routed in the top layer of the interconnect and horizontal wires are routed in the bottom layer. Hence, wires may cross each other but cannot overlap for any distance or cross node to which they are not incident. To change direction, wires may turn into the other layer by contact cuts or vias that facilitate connections between the two layers. In our discussion, no knock-knees are allowed—that is, two wires cannot turn at the same grid point [14, 15].

Formally, an embedding or layout of a graph $\mathcal{G}$ in a Thompson grid is an assignment of the nodes of $\mathcal{G}$ to intersection points in the grid and the edges of $\mathcal{G}$ to paths along the grid tracks. One of the important measures of a layout is the layout area which is defined as the product of the number of vertical tracks and the number of horizontal tracks that the layout uses to contain all the nodes and all the path segments.

2.2   Cube-Connected Cycles

The s-dimensional (s-d for short) cube-connected cycles (CCC) is constructed from the s-dimensional hypercube by replacing each node of the hypercube with a cycle of s nodes [13, 16]. The ith-d edge of a node of the hypercube is then connected to the ith node of the corresponding cycle of the CCC. For example, see Fig. 1(a,b). The resulting graph has $s \cdot 2^s$ nodes, each of degree 3. By extending the labeling scheme of the hypercube, we can represent each node of the CCC by $\langle w, i \rangle$ where i

$(1 \leq i \leq s)$ is the position of the node within its cycle and w (an s-bit binary string with the 0th-d at the rightmost) is the label of the node in the hypercube that corresponds to the cycle. Two nodes, $\langle w, i \rangle$ and $\langle w', i' \rangle$, are linked by an edge in the $\mathcal{CCC}$ if and only if either

  1. $w = w'$ and $i - i' = \pm 1 \pmod{s}$, or

  2. $i = i'$ and w differs from w' in precisely the ith bit.

Edges of kind (1) are cycle-edges and edges of kind (2) are cube-edges. As shown in Fig. 1(c), $\mathcal{CCC}$ is often drawn in the multi-stage format which will directly give rise to the Preparata-Vuillemin layout. The first and the last stage, stages 1 and s, are called the two end stages, and they consist of all the nodes $\langle w, i \rangle$ for $i = 1$ and $i = s$ respectively.

The $\mathcal{CCC}$ is closely related to the butterfly network just as the shuffle-exchange network is to the deBruijn network. The group-theoretic relations of the four networks are well studied in [1] where the $\mathcal{CCC}$ and the butterfly are proved to be Cayley graphs derivable from the shuffle-exchange network and the deBruijn network respectively; and inversely, the shuffle-exchange network and the deBruijn network are proved to be some coset graph of the $\mathcal{CCC}$ and the butterfly network respectively.

We introduce an unfolded version of the $\mathcal{CCC}$. Like the butterfly network, the $\mathcal{CCC}$ now has the traditional folded version and the new unfolded version. For the unfolded $\mathcal{CCC}$, condition (1) in the above definition is changed to

  1. $w = w'$ and $i - i' = \pm 1$.

Each cycle of the $\mathcal{CCC}$ is replaced by a line in the unfolded $\mathcal{CCC}$. We therefore call the unfolded $\mathcal{CCC}$ the cube-connected lines, denoted by $\mathcal{CCL}$ hereafter. The 3-d $\mathcal{CCL}$ is shown in Fig. 1(d).

2.3   The Preparata-Vuillemin Layout

Fig. 2(b) shows the Preparata-Vuillemin layout of a 4-d $\mathcal{CCC}$, which is recursively constructed from two 3-d $\mathcal{CCC}$s (identified by the dotted lines). Based on the recursive construction, it can be easily proved that an $\mathcal{CCC}$ of $N = s \cdot 2^s$ nodes can be placed on a $2 \cdot 2^s \times (2^s + 1)$ chip. Since $s \simeq \log(N/\log N)$, the chip size is about $O((N/\log N)^2)$. In general, we say that a network of N nodes has asymptotically-optimal layout if it can be laid out in area $O(N^2/T^2)$, where T is the time to execute an ascend-descend algorithm [4, 18]. $\mathcal{CCC}$ can execute the ascend-descend algorithm in time $O(\log N)$ [16]. Therefore, the Preparata-Vuillemin layout is asymptotically optimal.

More precisely, for an s-d $\mathcal{CCC}$ with $n = 2^s$ cycles, denoted by $\mathcal{CCC}(n)$ hereafter, let $W(s)$ and $H(s)$ be the numbers of vertical and horizontal tracks respectively—$i.e.$, the width and the height
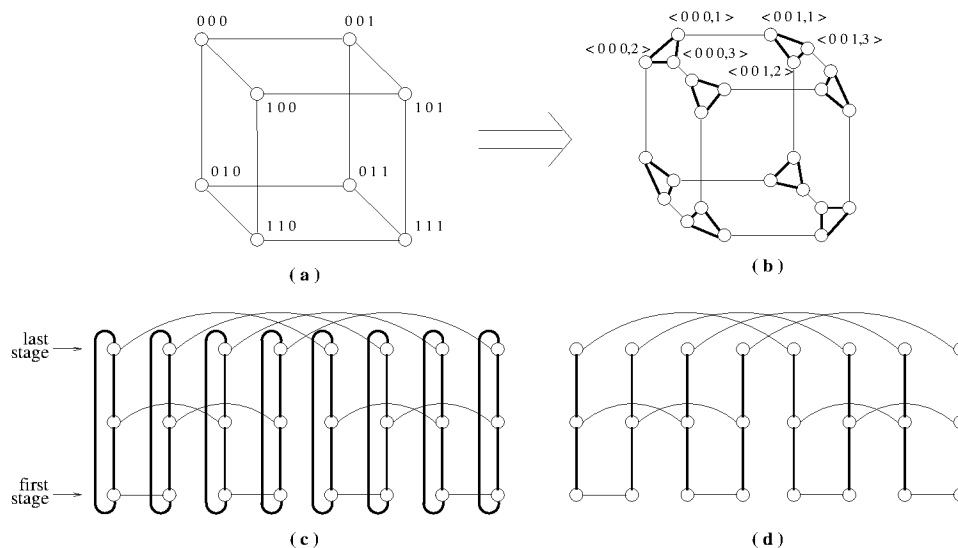
Figure 1: (a) 3-d hypercube. (b) 3-d $\mathcal{CCC}$. (c) Another drawing of 3-d $\mathcal{CCC}$: cycle-edges in in thick lines and cube-edges in thin lines. (d) 3-d $\mathcal{CCL}$.

of a layout. Then, for the Preparata-Vuillemin layout,

$$W(1) = 4,$$
$$H(1) = 3,$$
$$W(s) = 2W(s-1),$$
$$H(s) = H(s-1) + 2^{s-1}.$$

We get $W(s) = 2^{s+1} = 2n$ and $H(s) = 2^s + 1 = n + 1$. Hence, the area occupied by the Preparata-Vuillemin layout, $W(s) \times H(s)$, is

$$2n(n+1) = 2n^2 + 2n. \tag{1}$$

For the "more economical" Preparata-Vuillemin layout which is shown in Fig. 2(c),

$$W(1) = 4,$$
$$H(1) = 3,$$
$$W(s) = 2W(s-1),$$
$$H(s) = \begin{cases} H(s-1) + 2^{s-1} & \text{if s is odd} \\ H(s-1) + 2^{s-2} + 1 & \text{if s is even.} \end{cases}$$

The saving in the number of horizontal tracks for the case of even s comes from the overlapping of some of the sth-d tracks with the embedded layouts for the $(s-1)$-d $\mathcal{CCC}$ (see the dotted region in Fig. 2(c)). From the above, we get $W(s) = 2^{s+1} = 2n$ and $H(s) = 3 + (2 + 4 + 5 + \cdots 2^{s-2} + (2^{s-2} + 1)) = \frac{2}{3}2^s + \frac{1}{2}s + \frac{4}{3}$ for even s, and $H(s) = \frac{5}{6}2^s + \frac{1}{2}s + \frac{5}{6}$ for odd s. For simplicity we only consider even s. Hence, the area is

$$\frac{4}{3}n^2 + n \log n + \frac{8}{3}n. \tag{2}$$

4
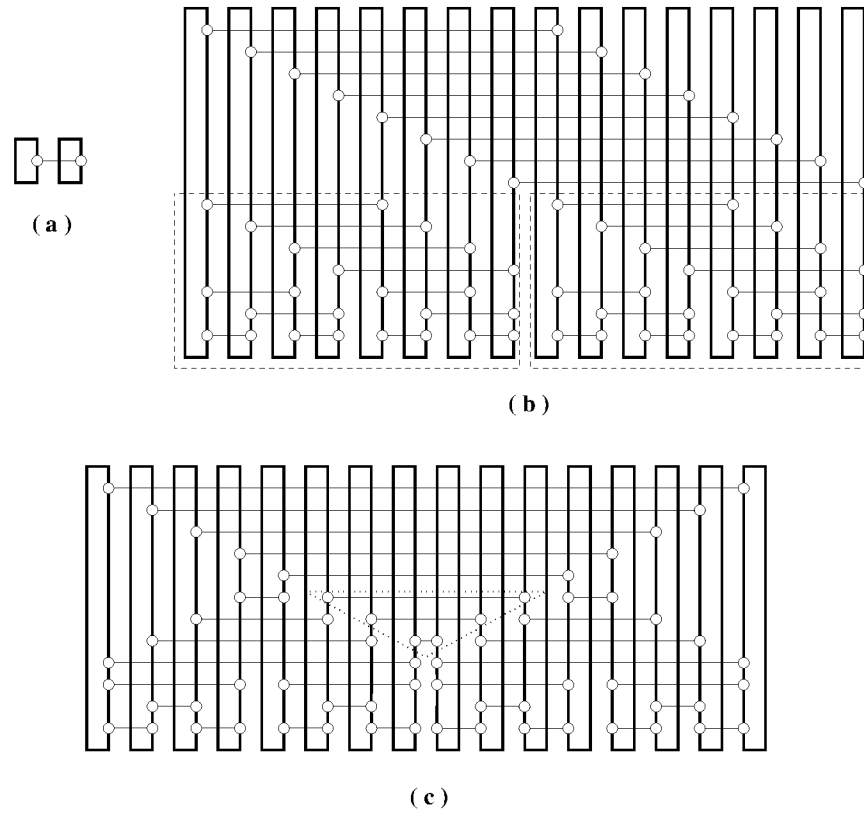
Figure 2: (a) Preparata-Vuillemin layout of 1-d $\mathcal{CCC}$—the base case. (b) Preparata-Vuillemin layout of 4-d $\mathcal{CCC}$. (c) More economical Preparata-Vuillemin layout of 4-d $\mathcal{CCC}$.

3   New Layouts

Although the Preparata-Vuillemin layout for CCC is asymptotically optimal, it is not the minimal layout. For real implementations, we would prefer using as tight a layout as possible. Here we give a new layout for the CCC. It is more compact than the Preparata-Vuillemin layout; whether it is minimal is an open question.

Referring to Fig. 2 again, there are two obvious shortcomings in the Preparata-Vuillemin layout:

- the layout does not try to make use of the corner positions of a cycle by putting some nodes there, and

- the layout places all the cycles along the same horizontal axis.

In the new layout, these two problems are corrected, and the resulting layout uses less area and has a better aspect ratio.

Fig. 3(a,b,c) show the layouts of the first three CCC's, starting from the second dimension. These layouts use minimal areas. As our interest is in the layout of the general CCC, we omit the proofs of these specific cases here. Like the Preparata-Vuillemin layout, the new layout is based on recursive construction. Unlike the Preparata-Vuillemin layout for which the recursion begins at the first dimension, the base case for recursion in the new layout is the 4-d CCC (Fig. 3(c)). The reason for this is that the layout of the 4-d CCC is the first one (starting from the first dimension) that puts a node in every corner of a cycle. This layout is correct in the sense that it is indeed a valid CCC that is being laid out. This can be easily verified by examining the connections against the labels of the cycles in Fig. 3(c). Similarly for the smaller cases.

3.1   Recursive Construction

The procedure is as follows, for $s \geq 5$.

> Take two copies of the layout for the $(s-1)$-d CCC; place them side by side. Stretch every cycle vertically by an extra height of 2^{s-3} for the embedding of the sth-d nodes and edges.

Since there are four rows of cycles from top to bottom, a total of 2^{s-1} extra horizontal tracks are added. Note how the sth-d nodes and edges are embedded (refer to Fig. 3(d) and Fig. 4) within these extra tracks: one node is added to every cycle, and its corresponding node in the other copy of the layout of the $(s-1)$-d CCC is placed at the same horizontal position, and the two are joined by a horizontal wire. We label the cycles in the left copy by extending the original labels by a 0 on the left, and the cycles in the right copy by a 1 on the left. The correctness of the s-d layout immediately follows from this labeling scheme. In Fig. 3(d), which is recursively constructed from Fig. 3(c), the labels of the bottom row of cycles are shown.
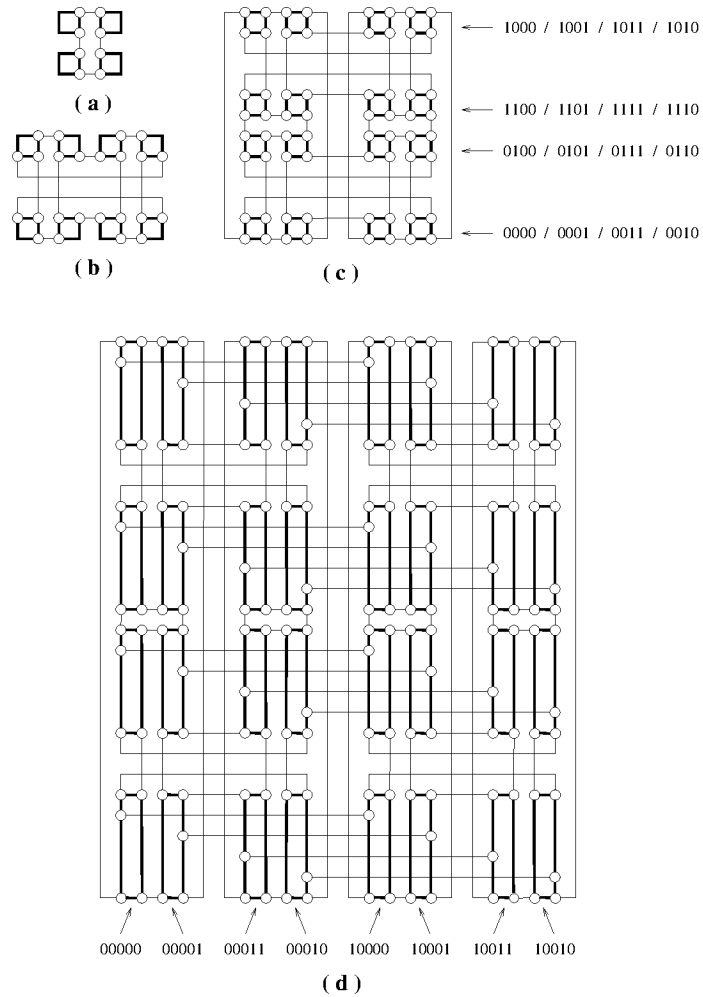
Figure 3: New layouts of small $\mathcal{CCC}$'s: (a) 2-d $\mathcal{CCC}$ needs area 4×4; (b) 3-d $\mathcal{CCC}$ needs area 8×6; (c) 4-d $\mathcal{CCC}$ needs area 12×12; (d) 5-d $\mathcal{CCC}$ needs area 24×28.

Using the procedure, the layout of the 6-d CCC can be easily constructed. The result is shown in Fig. 4.

For an s-d CCC with $n = 2^s$ cycles,

$$W(4) = 12,$$
$$H(4) = 12,$$
$$W(s) = 2W(s-1),$$
$$H(s) = H(s-1) + 2^{s-1}.$$

We get $W(s) = 12 \times 2^{s-4} = \frac{3}{4}n$ and $H(s) = 2^s - 4 = n - 4$. Hence, the area, $W(s) \times H(s)$, is

$$\frac{3}{4}n^2 - 3n. \tag{3}$$

In the same way that a more economical layout can be derived from the Preparata-Vuillemin layout, the new layout has a more economical version. The more economical version for the 6-d CCC is shown in Fig. 5. For this improved layout,

$$W(4) = 12,$$
$$H(4) = 12,$$
$$W(s) = 2W(s-1),$$
$$H(s) = \begin{cases} H(s-1) + 2^{s-1} & \text{if s is odd} \\ H(s-1) + 2^{s-2} + 4 & \text{if s is even.} \end{cases}$$

We get $W(s) = 12 \times 2^{s-4} = \frac{3}{4}n$ and $H(s) = 12 + (16 + 20 + 64 + 68 + \cdots + 2^{s-2} + (2^{s-2} + 4)) = \frac{2}{3}2^s + 2s - \frac{20}{3}$ for even s. Hence, the area is

$$\frac{1}{2}n^2 + \frac{3}{2}n \log n - 5n. \tag{4}$$

3.2   Comparison

By ignoring the low-order terms in Formulae 1, 2, 3 and 4, the four layout schemes of $CCC(n)$ discussed above take areas of approximately $2n^2$, $\frac{4}{3}n^2$, $\frac{3}{4}n^2$ and $\frac{1}{2}n^2$ respectively. We compare the new layout with the Preparata-Vuillemin layout, and the more economical version of the new layout with the more economical version of the Preparata-Vuillemin layout. In either case, the new layout scheme uses less than half the area of the Preparata-Vuillemin layout. The other important advantage of the new layout is that it has a more practical aspect ratio $(W(s)/H(s))$, which is close to 1, whereas the aspect ratio of the Preparata-Vuillemin layout could be as large as 3. Because of a better aspect ratio, the new layout has a shorter maximum wire length than the Preparata-Vuillemin layout.

The new layout also shows the superiority of the CCC in layout area over other hypercube substitutes such as the shuffle-exchange network and the butterfly network [13]. The optimal layout of

8
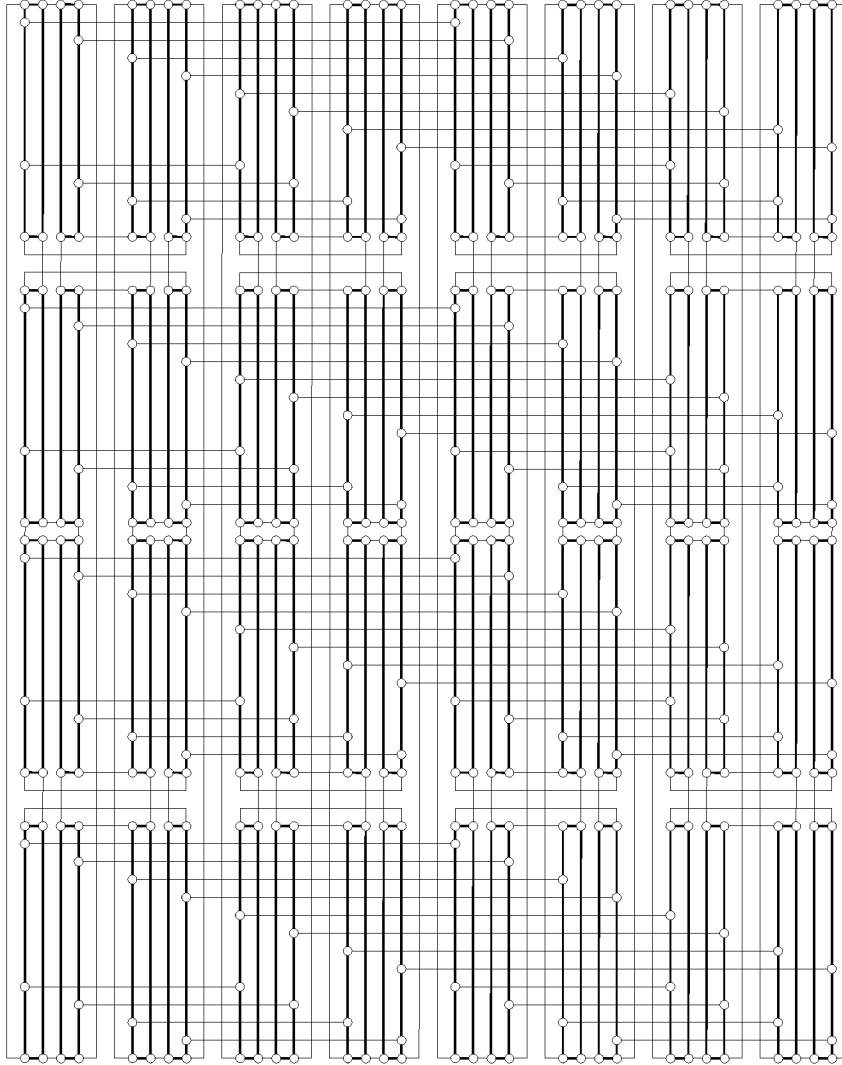
Figure 4: New layout of 6-d $\mathcal{CCC}$ with area 48×60.
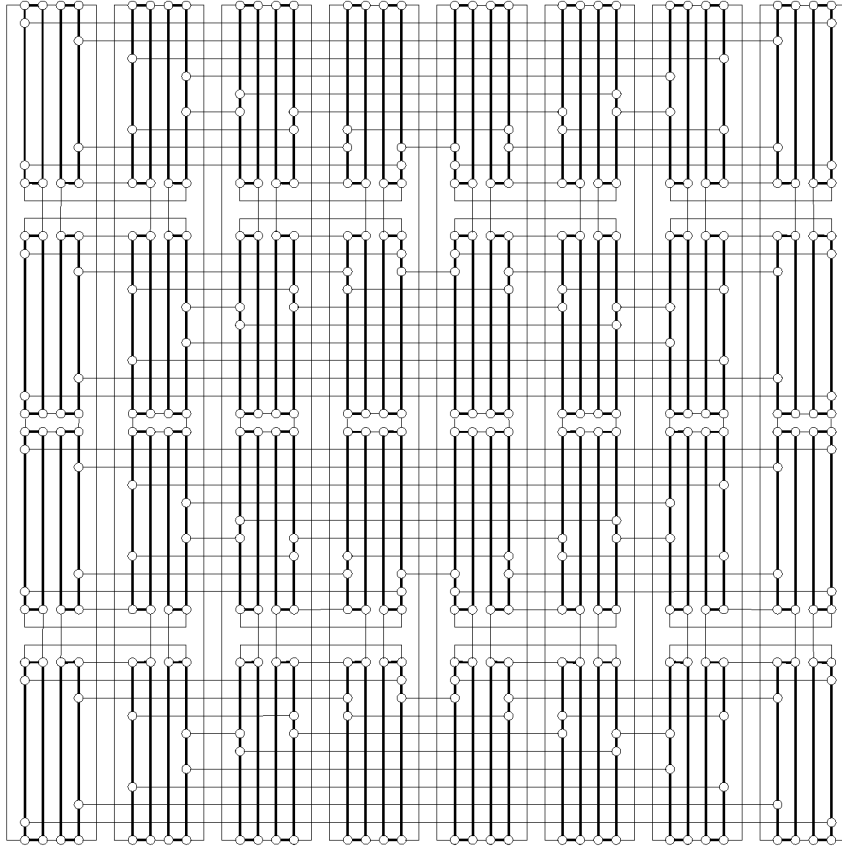
Figure 5: More economical new layout of 6-d $\mathcal{CCC}$ with area 48×48.

the shuffle-exchange network was due to Leighton [12]. His layout of the shuffle-exchange network, as well as the other related ones, however, are complicated, not regular or recursive. For years, the best known layout of the butterfly network with n inputs or outputs was that by Wise [19] which has area $\simeq 2n^2$. Recently, a more compact layout for the butterfly was found with area $\simeq \frac{11}{6}n^2$ [9]. The butterfly networks discussed in these two papers, however, are unfolded. To be fair, the folded butterfly network (*i.e.*, the first and the last stage are merged) [13] should be considered when comparing with the $\mathcal{CCC}$. The corresponding areas of the folded butterfly given in [9, 19] would then need to be doubled or quadrupled. On the other hand, as will be seen in Section 5, the unfolded $\mathcal{CCC}$ can be laid out with area $\simeq \frac{1}{4}n^2$.

4   Lower Bound on Layout Area

We give below a lower bound of $(\frac{1}{2}n - 1)^2$ for $\mathcal{CCC}(n)$. Our new layout of the $\mathcal{CCC}$ as presented in the previous sections deviates from this bound by a factor of 2. The construction of the lower bound does not consider the laying out of the cycles in a $\mathcal{CCC}$(see Fig. 6), and hence the lower bound is also valid for the $\mathcal{CCL}$. As will be shown in the next section, the $\mathcal{CCL}$ can be laid out in an area of $(\frac{1}{2}n + o(n))^2$, which is tight when compared with the lower bound.

The lower bound of $(\frac{1}{2}n - 1)^2$ is easily seen from the bounding strategy invented in [18] which is in terms of the bisection width of a graph.

Lemma 1 *[18] For any graph $\mathcal{G}$ with bisection width $BW(\mathcal{G})$, $AREA(\mathcal{G}) \geq (BW(\mathcal{G}) - 1)^2$.*

The proof of the bisection width, $\frac{1}{2}n$, of $\mathcal{CCC}(n)$, however, is complicated. Alternatively, we can turn to the modified bounding strategy, from [2] where a lower bound of the butterfly network layout is proved by the same technique, but is in terms of the minimum special bisection width.

Let $\mathcal{G}$ be a graph having a designated set of special nodes. The minimum special bisection width of $\mathcal{G}$, denoted $MSBW(\mathcal{G})$, is the smallest number of edges whose removal partitions $\mathcal{G}$ into two disjoint subgraphs, each containing half of $\mathcal{G}$'s special nodes.

The following three lemmas are due to Avior *et al.* [2].

Lemma 2 *For any graph $\mathcal{G}$ with $MSBW(\mathcal{G})$, $AREA(\mathcal{G}) \geq (MSBW(\mathcal{G}) - 1)^2$.*

In order to bound the $MSBW$ of $\mathcal{CCC}(n)$, the congestion argument originated in [12, 13] which is used for bounding unknown $MSBW$'s from known ones is used.

Lemma 3 *Let $\mathcal{G}$ and $\mathcal{H}$ be graphs having equal numbers of special nodes. If there is an embedding of $\mathcal{G}$ into $\mathcal{H}$ which maps special nodes to special nodes and which has congestion $\leq C$, then*

$$MSBW(\mathcal{H}) \geq (1/C)MSBW(\mathcal{G}).$$

The complete bipartite graph $\mathcal{K}_{n,n}$ plays the role of the guest graph $\mathcal{G}$ with known $MSBW = \frac{1}{2}n^2$.

Lemma 4 $MSBW(\mathcal{K}_{n,n}) = \frac{1}{2}n^2$ when all nodes of $\mathcal{K}_{n,n}$ are special.

Now we give an embedding of the guest graph $\mathcal{K}_{n,n}$ into the host graph $CCC(n)$.

Lemma 5 One can embed $\mathcal{K}_{n,n}$ into $CCC(n)$ with congestion $2^s = n$ in such a way that the inputs and outputs of $\mathcal{K}_{n,n}$ map, respectively, to the first stage and the last stage of $CCC(n)$.

Proof: Consider the embedding of $\mathcal{K}_{n,n}$ into $CCC(n)$, which assigns inputs of $\mathcal{K}_{n,n}$ to the first stage of $CCC(n)$ and outputs of $\mathcal{K}_{n,n}$ to the last stage of $CCC(n)$, and which routes the edges of $\mathcal{K}_{n,n}$ in increasing order of dimensions—*i.e.*, from right to left.

With no loss of generality, see Fig. 6 for an embedding of $\mathcal{K}_{8,8}$ into $CCC(8)$. Since the long wrap-around cycle-edges of the CCC are not used for routing in the embedding, Fig. 6(a) is simplified to Fig. 6(b)—*i.e.*, $CCL(8)$. Fig. 6(b) can be isomorphically arranged to become Fig. 6(c) in which all stages of nodes except the first stage are reordered so that pairs of nodes connected by cube-edges are placed together like the first stage while the cycle-edges at each stage appear to be in the unshuffle-connection pattern [1, 8]. Fig. 6(c) can be transformed into Fig. 6(e) by replacing every pair of nodes with a complex node as shown in Fig. 6(d). Fig. 6(e) is a reverse Omega network (or a flip network [3]). Hence, the original $CCC(n)$ is transformed into a reverse Omega network with $\frac{1}{2}n$ inputs and $\frac{1}{2}n$ outputs.

Note that the reverse Omega network (with $\frac{1}{2}n$ inputs and $\frac{1}{2}n$ outputs) has the banyan property [10]: each input node u is connected to each output node v by exactly one path of length $s-1$. Let e be a stage-k edge of the reverse Omega network, where $1 \leq k \leq s-1$. One end-point of e reaches precisely 2^{s-k-1} distinct output nodes while the other end-point of e reaches precisely 2^{k-1} distinct input nodes. Hence, edge e lies on precisely 2^{s-2} input-output paths. Since each input and output contains two nodes of $\mathcal{K}_{n,n}$, edge e lies on precisely 2^s input-output paths—*i.e.*, its congestion is $2^s = n$.

A further look reveals that the congestion of cube-edges of $CCC(n)$, shown as a thin edge in Fig. 6(d), is also $2^s = n$ since from each input, exactly half of the paths will go through the cube-edge of a complex node.   □

Lemma 6 $MSBW(CCC(n)) \geq \frac{1}{2}n$.

Proof: Directly from Lemmas 3, 4, and 5.   □

Finally, Lemma 6 combines with Lemma 2 to yield the desired lower bound, Theorem 1, on the area of layouts of $CCC(n)$.

Theorem 1 Any layout of $CCC(n)$ has area at least $(\frac{1}{2}n - 1)^2$.
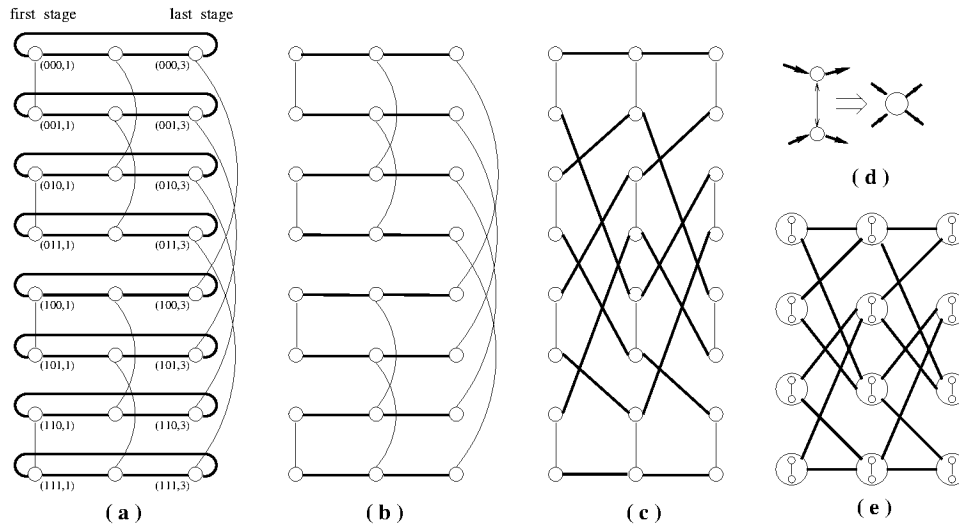
Figure 6: An embedding of $\mathcal{K}_{8,8}$ into $\mathcal{CCC}(n)$.

5 Layout of $\mathcal{CCL}$

In this section, we present the tight layout of the $\mathcal{CCL}$. Avior *et al.* [2] gave a tight layout of the unfolded butterfly network with area $(n + o(n))^2$. We borrow their technique and apply it to the $\mathcal{CCL}$, resulting in the desired tight layout of the $\mathcal{CCL}$ with area $(\frac{1}{2}n + o(n))^2$.

Theorem 2 *There is a layout of $\mathcal{CCL}$ with area $(\frac{1}{2}n + o(n))^2$.*

We prove Theorem 2 via a sequence of reductions.

5.1 First Reduction

We show how to construct the desired layout of $\mathcal{CCL}(4n)$ from four copies of a suitable layout of $\mathcal{CCL}(n)$. In the following, a stage that is placed "along" a side of a grid means that each of the nodes of the stage is either directly on the side or there is no other node that is between it and the side.

Lemma 7 *One can construct a layout L_{4n} of $\mathcal{CCL}(4n)$ with the area indicated in Theorem 2, from four copies of a layout L_n of $\mathcal{CCL}(n)$ that has the following properties.*

- *L_n places $\mathcal{CCL}(n)$ in an $(n + o(n)) \times (n + o(n))$ grid.*

- *L_n places one end stage of $\mathcal{CCL}(n)$ along a vertical side of the grid and the other end stage of $\mathcal{CCL}(n)$ along a horizontal side of the grid.*

Proof: Assume without loss of generality that the given layout L_n places the end stages along the bottom and the right sides; see Fig. 7(a). Flip L_n around horizontally to produce layout L'_n of
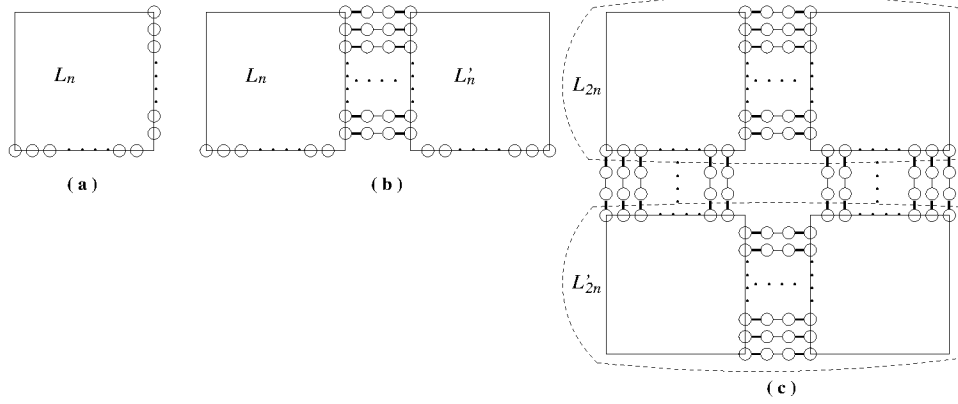
Figure 7: (a) Layout L_n of $\mathcal{CCL}(n)$. (b) Layout L_{2n} of $\mathcal{CCL}(2n)$. (c) Final layout L_{4n} of $\mathcal{CCL}(4n)$.

$\mathcal{CCL}(n)$. To form $\mathcal{CCL}(2n)$, L_n (as well as L'_n) must be extended by an extra stage of nodes. These extra stages of nodes are placed alongside the vertical sides of L_n and L'_n, as shown in Fig. 7(b). Join the two layouts by placing them side by side and then connecting the two newly added stages of nodes, resulting in L_{2n}, a layout of $\mathcal{CCL}(2n)$.

Next, flip layout L_{2n} vertically to produce layout L'_{2n} of $\mathcal{CCL}(2n)$. Add the extra stages of nodes, and then join layouts L_{2n} and L'_{2n} by connecting these stages of nodes, as depicted in Fig. 7(c), to produce layout L_{4n}. Clearly, layout L_{4n} resides in a $(2n + o(n)) \times (2n + o(n))$ grid, and thus satisfies the conditions of Theorem 2. $\qquad\square$

We have thus reduced the layout problem to one of producing L_{2n}.

5.2   Second Reduction

We now show how to construct layout L_n that is needed by Lemma 7. For convenience, we assume that s is even. The case of odd s can be dealt with similarly, and the result will differ in only the lower-order terms.

Lemma 8 *Suppose that we can lay out any $\mathcal{CCL}(n)$ in an $n \times (n-1)$ grid, in such a way that the end stages of $\mathcal{CCL}(n)$ are along the two vertical sides of the grid. Then one can construct layout L_n of $\mathcal{CCL}(n)$ as described in Lemma 7.*

Proof: Let the stages of $\mathcal{CCL}(n)$ be numbered $1, 2, ..., s-1, s$, where $n = 2^s$, and stages 1 and s are the end stages. We create layout L_n of $\mathcal{CCL}(n)$ as follows.

Let $k = s/2$. By cutting the edges between stage k and stage $k+1$ we decompose $\mathcal{CCL}(n)$ into

- $\mathcal{CCL}(n, 1)$: the subgraph of $\mathcal{CCL}(n)$ bounded by stage 1 and stage k (nodes and edges), and

- $\mathcal{CCL}(n, 2)$: the subgraph of $\mathcal{CCL}(n)$ bounded by stage $k+1$ and stage s (nodes and edges).

14
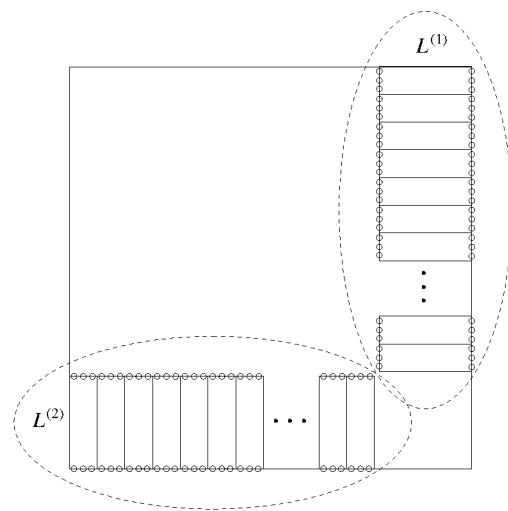
Figure 8: $L^{(1)}$ and $L^{(2)}$ inside L_n.

Let L be a layout of $CCL(2^k)$ in a $2^k \times (2^k - 1)$ grid, in which the two end stages reside along the two vertical sides of the grid. We first construct $L^{(1)}$ of $CCL(n, 1)$ by putting 2^k copies of L, one above another, along the right side of the grid for L_n. Layout $L^{(1)}$ resides in an $n \times (2^k - 1)$ grid. By basic properties of the hypercube, $CCL(n, 2)$ is isomorphic to $CCL(n, 1)$ by a suitable relabeling of the lines; we do the same for $CCL(n, 2)$ as we just did to $CCL(n, 1)$. The result is rotated 90 degrees to produce $L^{(2)}$, a layout of $CCL(n, 2)$ along the bottom side of the grid for L_n.

It can be easily seen that the smallest grid for L_n that can hold $L^{(1)}$ and $L^{(2)}$ is of area $(n + o(n)) \times (n + o(n))$.

Finally, we must connect $L^{(1)}$ and $L^{(2)}$ to re-create $CCL(n)$. This can be accomplished by routing a specific bijection between the two subgraphs of $CCL(n)$ that have been laid out. It is obvious that the unpopulated area that is left behind after placing the layouts $L^{(1)}$ and $L^{(2)}$ is sufficient for any such bijection to be routed. $\qquad\square$

5.3   Third Reduction

Our last task is to construct the layout L of $CCL(n)$ as demanded in Lemma 8. This can be easily done by modifying the Preparata-Vuillemin layout: all cycles become lines, and nodes can be placed at the two ends of a line (refer to Fig. 2(b)). In particular, $CCL(n)$ can be laid out in an $n \times (n-1)$ grid.

Hence, Theorem 2 is proved.

6   Conclusion

We have given a simple, regular and more compact layout scheme for the $\mathcal{CCC}$, which takes less than half of the area of the Preparata-Vuillemin layout. We have also derived a lower bound on the layout area of the $\mathcal{CCC}$. Our layout deviates from the lower bound by a constant factor of 2. The lower bound, however, is not the tightest possible because its construction does not take into account the laying out of the cycles in a $\mathcal{CCC}$. It is, more appropriately, a lower bound for the $\mathcal{CCL}$. On the other hand, our tight layout of the $\mathcal{CCL}$ can give rise to a layout of the $\mathcal{CCC}$, but the area will be four times that of the $\mathcal{CCL}$ (consider Fig. 8, and the width and the height of the grid will need to be doubled to accommodate the cycles of the $\mathcal{CCC}$). We conjecture therefore that the layout of the $\mathcal{CCC}$ as we have proposed in this paper is optimal. Further work will be directed to deriving a lower bound for the $\mathcal{CCC}$ that would take the cycles into account.

Our layout of the $\mathcal{CCL}$ reveals the superiority of the $\mathcal{CCL}$ over the unfolded butterfly network since the former takes only one-fourth of the layout area of the unfolded butterfly network [2]. Another merit of the $\mathcal{CCL}$ is that a $\mathcal{CCC}$ can be embedded into a $\mathcal{CCL}$ with congestion 2 and dilation 2 due to the well-known fact that a cycle can be embedded into a line with congestion 2 and dilation 2. Hence, the $\mathcal{CCL}$ can be a good substitute for the $\mathcal{CCC}$ and can execute the ascend-descend algorithm with a small constant slowdown.

Another important measure of a layout is the maximum wire length [4, 11]. We have recently succeeded in coming up with a layout of the $\mathcal{CCC}$ which has no long wires and yet preserves the asymptotic-optimality of the area [7]. Our next task is to consider the tradeoff [4, 6] between area and maximum wire length for the $\mathcal{CCC}$ layouts.

References

[1] F. Annexstein, M. Baumslag, and A.L. Rosenberg. Group action graphs and parallel architecture. *SIAM J. on Computing*, pages 544–569, June 1990.

[2] A. Avior, T. Calamoneri, S. Even, A. Litman, and A.L. Rosenberg. A tight layout of the butterfly network. Technical report, University of Massachusette, 1996.

[3] K.E. Batcher. The flip network in STARAN. In *Proc. International Conference on Parallel Processing*, pages 65–71, Detroit, MI, 1976.

[4] R. Beigel and C.P. Kruskal. Processor networks and interconnection networks without long wires. *Proc. ACM Symposium on Parallel Algorithm and Architecture*, 1989.

[5] S.N. Bhatt and F.T. Leighton. A framework for solving VLSI graph layout problem. *J. of Computer and System Sciences*, 28:300–343, 1984.

[6] N. Blum. An area-maximum edge length tradeoff for VLSI layout. In *Proc. 16th Ann. ACM Symp. on Theory of Computing*, pages 92–97, 1984.

[7] G. Chen and F.C.M. Lau. Layout of CCC without long wires. Technical report, 1997.

[8] G. Chen and F.C.M. Lau. Shuffle-X and their Cayley variants. In preparation, 1997.

[9] Y. Dinitz. A compact layout of butterfly on the square grid. Technical Report 873, Technion-Israel Institute of Technology, Haifa 32000, Israel, November 1995.

[10] C.P. Kruskal and M. Snir. A unified theory of interconnection network structure. *Theoretical Computer Science*, 48:75–94, 1986.

[11] F.C.M. Lau and G. Chen. Optimal layouts of Midmiew networks. *IEEE Transactions on Parallel and Distributed Systems*, 7(9):954–961, September 1996.

[12] F.T. Leighton. *Complexity Issues in VLSI: Optimal Layout for the Shuffle-Exchange Graph and Other Networks*. The MIT Press, 1983.

[13] F.T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercube*. Morgan Kaufmann Publishers, 1992.

[14] C. Mead and L. Conway. *Introduction to VLSI Systems*. Addison Wesley, 1980.

[15] K. Mehlhorn, F.P. Preparata, and M. Sarrafzadeh. Channel routing in knock-knee mode: Simplified algorithms and proofs. *Algorithmica*, 1:213–221, 1986.

[16] F.P. Preparata and J. Vuillemin. The cube-connected cycles: A versatile network for parallel computation. *CACM*, 24(5):300–309, May 1981.

[17] C.D. Thompson. Area-time complexity for VLSI. In *Proc. 11th Ann. Symp. on Theory of Computing*, pages 81–88, Atlanta, GA, May 1979.

[18] C.D. Thompson. *A Complexity Theory for VLSI*. PhD thesis, CMU Computer Science Department, 1980.

[19] D.S. Wise. Compact layouts of Banyan/FFT networks. In H.T. Kung, R. Sproull, and G. Steele, editors, *VLSI Sytems and Computations*, pages 186–195. Springer-Verlag, 1981.

Unifying Mesh- and Tree-Based Programmable Interconnect

André DeHon, *Member, IEEE*

Abstract—We examine the traditional, symmetric, Manhattan mesh design for field-programmable gate-array (FPGA) routing along with tree-of-meshes (ToM) and mesh-of-trees (MoT) based designs. All three networks can provide general routing for limited bisection designs (Rent's Rule with $p < 1$) and allow locality exploitation. They differ in their detailed topology and use of hierarchy. We show that all three have the same asymptotic wiring requirements. We bound this tightly by providing constructive mappings between routes in one network and routes in another. For example, we show that a (c, p) MoT design can be mapped to a $(2c, p)$ linear population ToM and introduce a corner turn scheme which will make it possible to perform the reverse mapping from any (c, p) linear population ToM to a $(2c, p)$ MoT augmented with a particular set of corner turn switches. One consequence of this latter mapping is a multilayer layout strategy for N-node, linear population ToM designs that requires only $\Theta(N)$ two-dimensional area for any p when given sufficient wiring layers. We further show upper and lower bounds for global mesh routes based on recursive bisection width and show these are within a constant factor of each other and within a constant factor of MoT and ToM layout area. In the process we identify the parameters and characteristics which make the networks different, making it clear there is a unified design continuum in which these networks are simply particular regions.

Index Terms—Butterfly fat tree (BFT), fat pyramid, fat tree, field-programmable gate-array (FPGA), hierarchical, hierarchical synchronous reconfigurable array (HSRA), interconnect, Manhattan, mesh, mesh-of-trees (MoT), multilevel metallization, Rent's rule, tree-of-meshes (ToM).

I. INTRODUCTION

IN THE DESIGN of field-programmable gate-arrays (FPGAs), we have seen mesh based (e.g., [1]–[5]), hierarchical (e.g., [6]–[11]), and hierarchical mesh interconnection networks (e.g., [12], [13]). We have seen numerous studies showing the characteristics of these networks, how they scale, and empirically how they relate on particular designs (e.g., [7], [9], [13]). In this paper, we examine how they relate in a more fundamental manner. We ask if we can provide any guaranteed bounds on the size of one network given a routing solution in another network (e.g., given a mesh-of-trees (MoT) route, how much larger or smaller can the tree-of-meshes (ToM) route be? Mesh route?). We further ask if there are design variables that allow us to tune the design space between two different network types. This allows us to underscore the ways in which these networks are fundamentally similar and the ways in which they are fundamentally different.

A. Why Does This Matter?

Each of these networks has interesting and useful properties. We would like to know which of these properties are mutually exclusive and which properties might be simultaneously achieved in a single network. For example:

- the ToM network can be placed entirely by recursive bisection;
- the MoT network can be laid out in $\Theta(N)$ two-dimensional (2-D) area when given sufficient metal layers for routing;
- since all three networks are ultimately embedded in a mesh, the optimal mesh placement and routing will achieve minimum wire lengths or minimum total wiring.

We have some evidence that certain combinations are not possible. For example, the traditional switch population schemes for meshes require asymptotically more switches than the MoT or ToM. So, we want to know the following.

- Is it possible to achieve the simplified placement and routing of the ToM simultaneously with the MoT layout guarantee?
- How much does the strict recursive bisection placement which the ToM uses cost compared to an optimal mesh placement? Is it an asymptotically larger costs or just a constant larger?
- The MoT achieves asymptotically fewer switches than the mesh; does it require asymptotically more wiring?
- The MoT can exploit 2-D locality better than the ToM; what does this locality exploitation cost us?

By performing these equivalence mappings, we can answer these questions and show when it is necessary to compromise one good network property for another and when it is possible to achieve good properties simultaneously in a single network design.

Whether or not recursive bisection alone is sufficient for placement is an important question in system-level interconnect prediction. This shows up both in questions about the adequacy of recursive bisection in constructive placements [14] and in questions about the relationship between the pre-placement and post-placement Rent parameters [15], [16] [see (1)]. These relations help us provide, at least, an asymptotic answer. We show that the pre-placement partitioning implies a constructive layout and wiring which need never be more than a constant factor greater than the optimal, post-placement wiring. This gives us insight into the validity of using pre-placement partitioning to predict wiring requirements; the constant factor gap
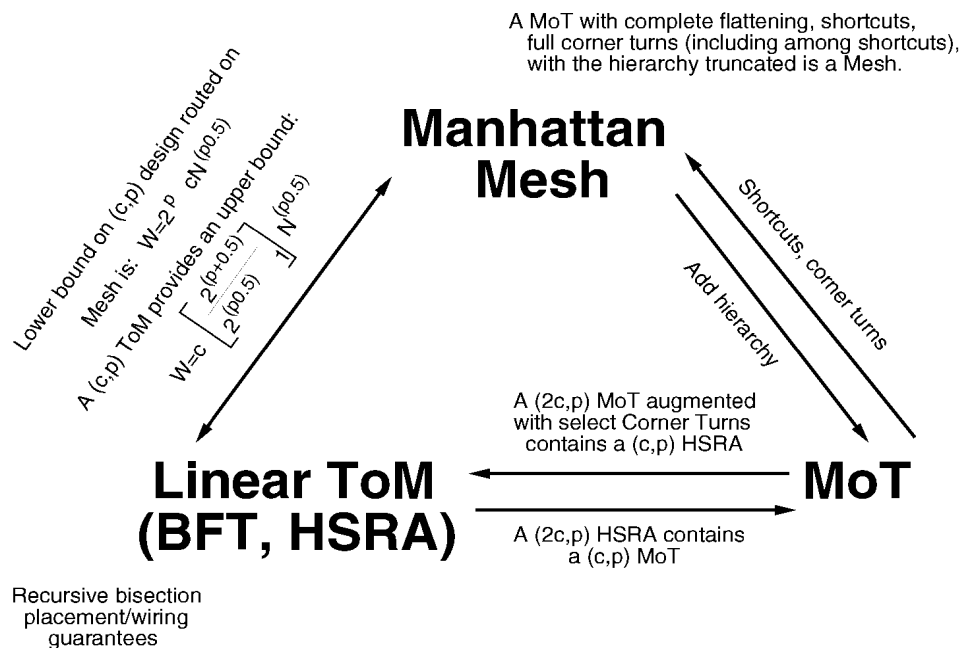
Fig. 1.   Equivalence relation summary.

between the upper and lower bound helps us understand the source of variance in pre- versus post-placement wiring. This further helps us understand the wiring tradeoffs which will be involved when using fast-placement techniques for run-time placement of FPGAs [17], [18].

In earlier work [10], we showed that a properly balanced FPGA-style network may leave some compute elements unused in order to better utilize the expensive wiring. We demonstrated this on a ToM network. The question remained whether or not this result was transferable to more conventional mesh networks. Tessier later showed that it was [19]. When we know the fundamental equivalence relations between networks, we will know when results such as this must necessarily transfer between networks.

These equivalences allow us to establish the asymptotic relationships among the networks. As we consider the implications of exponentially scaling chip and system capacities, it is important to keep an eye on which resource requirements will fundamentally diverge and which are only constant factors apart. For example, the constant factor of additional wire capacity in the MoT or ToM may be a worthwhile expense to avoid paying the asymptotically growing switch requirements in conventional mesh designs.

B. Overview

Fig. 1 summarizes our key results. We start by briefly reviewing Rent's Rule and the three network styles which anchor the edges of our design space (Section II). We then show how the MoT is contained in a linear population ToM (Section III). We observe that a particular corner turn scheme, which only increases the switches in the MoT by a constant factor, gives us a MoT that contains a linear population ToM (Section IV); this shows us that the designs are within a constant factor of each other and identifies exactly the change we need to make to one network to turn it into the other. This also shows us how to layout

a linear population ToM in constant area per endpoint when given sufficient metal layers—giving us a design which can both be placed easily and efficiently. We also use the MoT layout to demonstrate that the MoT and linear switch population ToM networks require, at most, a constant factor more wire channels than the mesh. The MoT/linear-ToM layout shows us an upper bound for mesh channels which is only a constant factor larger than the lower bound. This further shows the cost of the easy placement and good layout for the ToM or augmented-MoT is bounded even in wire costs compared to the mesh. We then identify the set of parameterized differences between a pure MoT and a traditional Mesh showing that there is a continuum design between these two points (Section V). Combining bounds transitively, we see that all three networks require asymptotically the same number of wires. We note that pure ToM designs require only recursive bisection for layout, so these results suggest the wiring benefit of full mesh placement versus recursive bisection placement can be, at most, a constant factor effect.

II. BACKGROUND

All three network types—meshes, MoT, and ToM—are instances of limited-bisection networks. That is, rather than supporting *any* graph connectivity, like a crossbar or Beneš network, these networks are designed to exploit the fact that a typical N-node circuit or computing graph can be bisected (cut in half) by cutting less than $O(N)$ hyperedges. This is significant as the bisection width of a network, BW, directly places a lower bound on the size of the network when implemented in VLSI [20]. With a crossbar or Beneš network, the bisection width is $\Theta(N)$, as is the subsequent bisection of each half of the network. This means the horizontal and vertical width of the design, when implemented in a constant number of metal layers, must be $\Omega(N)$ which implies $\Omega(N^2)$ VLSI layout area. In contrast, a network which only has $BW < O(N)$ bisection width may be implemented in less area as noted below.

A. Limited Bisection Model: Rent's Rule

A common way of summarizing the wiring requirements for circuits is Rent's Rule [21]. Landman and Russo articulate this model for relating the number of gates N and the total number of input and outputs signals, IO

$$\text{IO} = cN^p. \tag{1}$$

This relationship assumes we attempt to maximize locality, i.e., we select the groups of N gates so as to minimize the number of signals (IO) which connect gates in a group to gates in other groups. Rent's observation was that this relation can be tuned to model the IO requirements for all such well chosen sub-groups $N < N_{\max}$. Here c and p are parameters that can be tuned to fit the IO versus N connectivity relationship for a design; c is a constant factor offset which roughly corresponds to the IO size of the primitive elements (gates, look-up tables) in a design, and p defines the growth rate. We can view p as a measure of locality. With $p = 1$, we have a design that has $\Theta(N)$ bisection bandwidth, and hence, has little locality. Note that any group of N gates with bounded fanin, k, will have at most $(k + 1) \times N$ IOs to cut even if all their nets enter and exit the partition. As p decreases, more of the possible signal nets are contained in the partition; the design has more locality and admits to smaller implementations. Landman and Russo, and a large body of subsequent work, observe that typical designs have $0.5 \le p \le 0.75$. Rent's Rule gives us a way of succinctly characterizing the wiring requirements for typical, limited-bi-section designs.

Strictly speaking, (1) captures the dominant asymptotic behavior of the design and the real IO versus group size relationship typically diverges from this at the high and low ends. Landman and Russo called the broad region where (1) held Region I and the top end where it no longer holds Region II. Stroobandt identified the divergence at the low end as Region III [22]. Because of this effect, the c in (1) may be different from the actual primitive element IO in order to better fit the Region I relationship.

Returning to our bisection based area lower bound, we can observe that

$$\text{BW}(\text{chip half}) = \text{IO}(\text{chip half}) = \text{IO}\left(\frac{N}{2}\right). \tag{2}$$

If we place half of the primitive elements in our design on each half of the chip, (2) reminds us that the total number of wires entering each half of the chip is related to the number of primitive elements in each half. To the extent our Rent relation (1) properly captures the IO versus gate relationship, we can use it to determine the number of wires that must cross the bisection of the chip

$$\text{BW}(\text{chip half}) = \text{IO}\left(\frac{N}{2}\right) = c\left(\frac{N}{2}\right)^p. \tag{3}$$

All of these wires must cross a line that runs the width of the chip and divides the chip into two pieces. We can use this relationship to get a lower-bound on the length of this line and hence the side
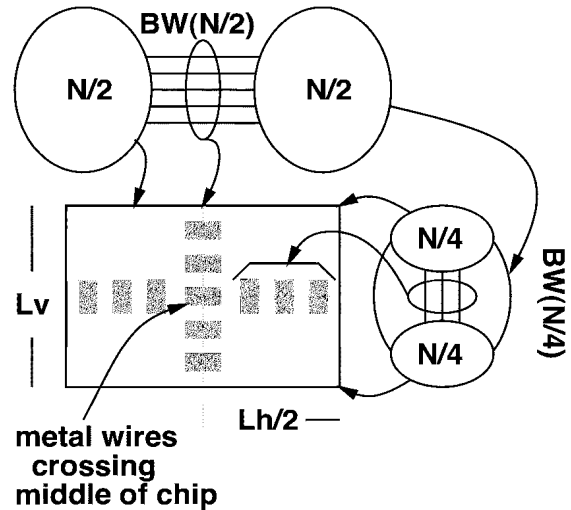
Fig. 2.   Area lower bound based on bisection widths.

length of the chip [see Fig. 2]. Without loss of generality, we assume this line is a vertical cut of the die. Then

$$L_v > \frac{W_{\text{pitch}} \times \text{BW}(\text{chip half})}{N_v}. \tag{4}$$

Here, W_{pitch} is the wire pitch, and N_v is the number of metal layers available for vertical wiring. Equation (4) says the best we can do is pack the wires crossing the bisection densely at the metal pitch into the available wiring layers.

Now that we have a bound on the vertical length of the chip, we can make a similar argument to bound the horizontal length of the chip. We consider cutting each half of the chip with a horizontal line that runs from the edge of the chip to the vertical cut line (see Fig. 2). This line produces two groups of size $N/4$ in each half of this chip half. Our Rent relation tells us the number of wires leaving each of these halves

$$BW(\text{chip quarter}) = \text{IO}\left(\frac{N}{4}\right) = c\left(\frac{N}{4}\right)^p. \tag{5}$$

This allows us to make a similar argument about the length of these horizontal lines

$$\frac{L_h}{2} > \frac{W_{\text{pitch}} \times BW(\text{chip quadrant})}{N_h}. \tag{6}$$

We can then put these two lower bounds together to compute a lower bound on the area of the chip due to wiring

$$\begin{aligned} A_{\text{wire}} &> L_v \times L_h \\ &> \left(\frac{(W_{\text{pitch}})^2}{N_v \times N_h}\right) \times BW\left(\frac{N}{2}\right) \times 2 \times BW\left(\frac{N}{4}\right) \\ &> \left(\frac{(W_{pitch})^2}{N_v \times N_h}\right) \times c\left(\frac{N}{2}\right)^p \times 2c\left(\frac{N}{4}\right)^p \\ &> \left(\frac{(W_{\text{pitch}})^2}{N_v \times N_h}\right) \times \left(\frac{2c^2}{8^p}\right) N^{2p}. \end{aligned} \tag{7}$$

Equation (7) gives us a lower bound on the wiring requirements for *any* layout of a graph with Rent characteristics (c, p). That
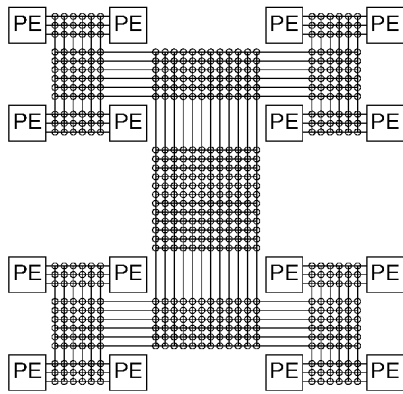
Fig. 3.   ToM topology (shown as $c = 3, p = 0.5$).

BFT

$c = 1$, arity-4

HSRA

$c = 2$, arity-2

Fig. 4.   BFT and HSRA topology.

is, any physical network which supports such a graph must have at least this much wiring area.

To be more precise, we can allow the IOs out of each region to exit on all four sides rather than just the one for the bisection. Note, however, that the regions will be roughly square in order to maximize the perimeter to area ratio, so the total perimeter is always a small constant factor times the edge length. Consequently, such refinements will not change the asymptotic wiring requirements. This wiring area lower bound argument is adapted from Thompson [20].

Equation (7) says our total wiring requirements are growing faster than linearly in gate count when $p > 0.5$. For any fixed number of wiring layers, N_h and N_v, this means wiring area forces the chip area to grow faster linearly for $p > 0.5$. If we allow the number of wiring layers to grow with N, perhaps we can keep A_{wire} down to an area linear in N. Our ability to do this will depend critically on our switch requirements and how the switches and wires are laid out.

B. ToM, BFT, HSRA

Leighton introduced the ToM (see Fig. 3, [23]) as a stylized, limited-bisection network which could be used as a template for the layout of any limited-bisection design and could be the basis of a configurable routing network [24]. Bhatt and Leighton use (α, F) as their parameterization rather than Rent's Rule's (c, p), but they define an equivalent space ($F = c\,(N_{\text{max}})^P$, $\alpha = 2^P$, where N_{max} is the total number of primitive elements in the design). By tuning the child to parent channel width growth of each of the tree stages, the ToM can be parameterized to support the (c, p) wiring requirements for any circuit. Significantly, if we can recursively partition a design so that its IO versus partition size relationship does not exceed the (c, p) of a ToM network, a $(4c, p)$ ToM network will always be able to route it. Using asymptotically the same number of switches, but organizing them differently, the factor of four can be reduced. Using a crossbar type interpretation of the ToM, a $(3c/2, p)$ network supports the (c, p) design [25]. The ToM allows us to do placement only by considering recursive bisections; this is a powerful property that simplifies physical mapping.

Leiserson adapted the ToM into the fat tree [26] and defined a linear switch population version which he called the butterfly fat tree (BFT) [27] (see Fig. 4, left-hand side). Our hierarchical synchronous reconfigurable array (HSRA) [11]
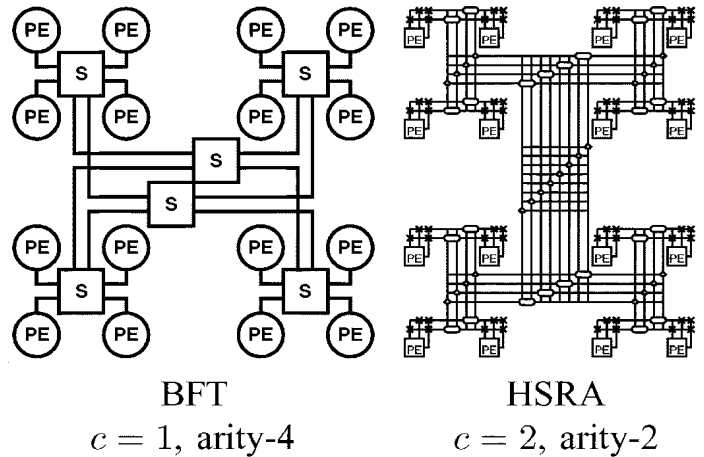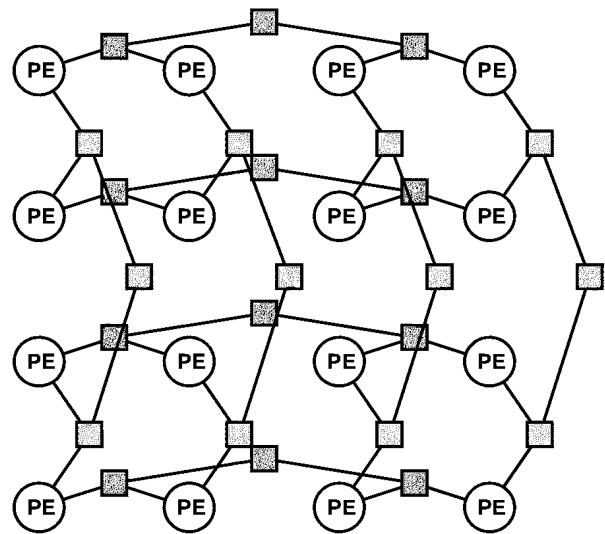
Fig. 5.   Basic MoT topology (shown $c = 1, p = 0.5$).

(see Fig. 4 right) is logically equivalent to a BFT. Both are "linearly populated" in that they have only a linear number of switches (linear in the number of child input channels) in each hierarchical switchbox rather than the quadratic number required by the full ToM (Fig. 3). One consequence of linear population is that the BFT or HSRA requires a total number of switches which is linear in the number of endpoints supported for any $p < 1$. In previous work we have identified resource and routability tradeoffs for this class of networks [25]. In [28], we showed that a $p = 0.5$ BFT could be laid out in $\Theta(N)$ area using $\Theta(\log(N))$ metal layers. However, we left open the question of whether or not a $p > 0.5$ BFT could be laid out in $\Theta(N)$ area given sufficient metal layers. Our ToM to MoT mapping (Section IV) shows that we can provide such a layout for any p, demonstrating that we have a network that can both be placed simply by using recursive bisection and be laid out in asymptotically optimal area using multilevel wiring.

C. MoT

Leighton also introduced the MoT [29], [23] (see Fig. 5). While the ToM–BFT–HSRA style designs have a single tree hierarchy, the MoT starts with a mesh of nodes and builds a

1055
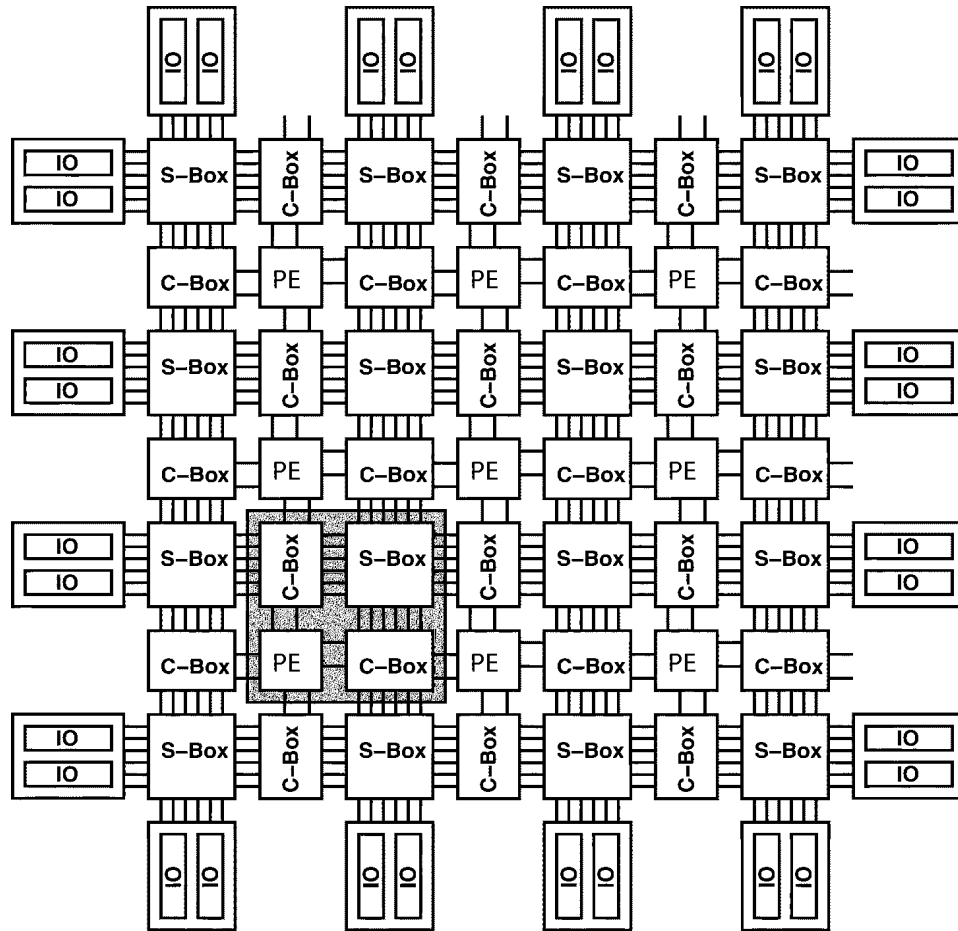
Fig. 6.   Manhattan mesh interconnect model (shown as $W = 6$).

separate hierarchical network along each row and column in the mesh. The resulting network has the asymptotic linear switch property of the BFT/HSRA with greater ability to exploit two-dimensional locality. In recent work [13], we identified how to parameterize the MoT for any Rent-style wiring requirements (c, p), calculated switch and layout asymptotics, and showed that the MoT required fewer switches than conventional Manhattan mesh designs. We further showed a multilayer layout for the MoT that required only $\Theta(N)$ area for any p. [13] was the first to demonstrate constant area, multilevel metallization layouts for any of these limited-bisection networks.

D. Manhattan Mesh

Manhattan Meshes (see Fig. 6) have been most heavily studied as interconnect networks for FPGAs (symmetric [4], island-style [5]). These place a routing channel containing W wire track between every row and column of processing elements. Each node may connect to a subset of the wire tracks adjacent to it through the connection box (C-Box). At the intersection of rows and columns, there is a switch box (S-Box) which allows wires to be linked together into longer signal runs or make Manhattan corner turns between a row and a column. Traditional designs have populated the switch boxes linearly in the number of channels, W.

A mesh arranged as $\sqrt{N} \times \sqrt{N}$ primitive elements has $\sqrt{N} + 1$ horizontal and vertical channels. The total bisection width of the mesh in the horizontal or vertical direction is then

$$\mathrm{BW}_{\mathrm{mesh}} = \left(\sqrt{N} + 1\right) W. \tag{8}$$

To support a design characterized by Rent Parameters (c, p), the Mesh will need

$$\mathrm{BW}_{\mathrm{mesh}} \geq c \left(\frac{N}{2}\right)^p. \tag{9}$$

Equation (9) is the same observation as (3)—the IO out of each half of the chip must cross the bisection. Combining (9) and (8), we can related W to the number of primitive elements, N, and our Rent parameters

$$\left(\sqrt{N} + 1\right) W \geq c \left(\frac{N}{2}\right)^p. \tag{10}$$

For simplicity, we can drop the plus one without affecting the asymptotic implications

$$\left(\sqrt{N}\right) W \geq c \left(\frac{N}{2}\right)^p$$

$$W \geq \left(\frac{c}{2^p}\right) N^{(p-0.5)}. \tag{11}$$
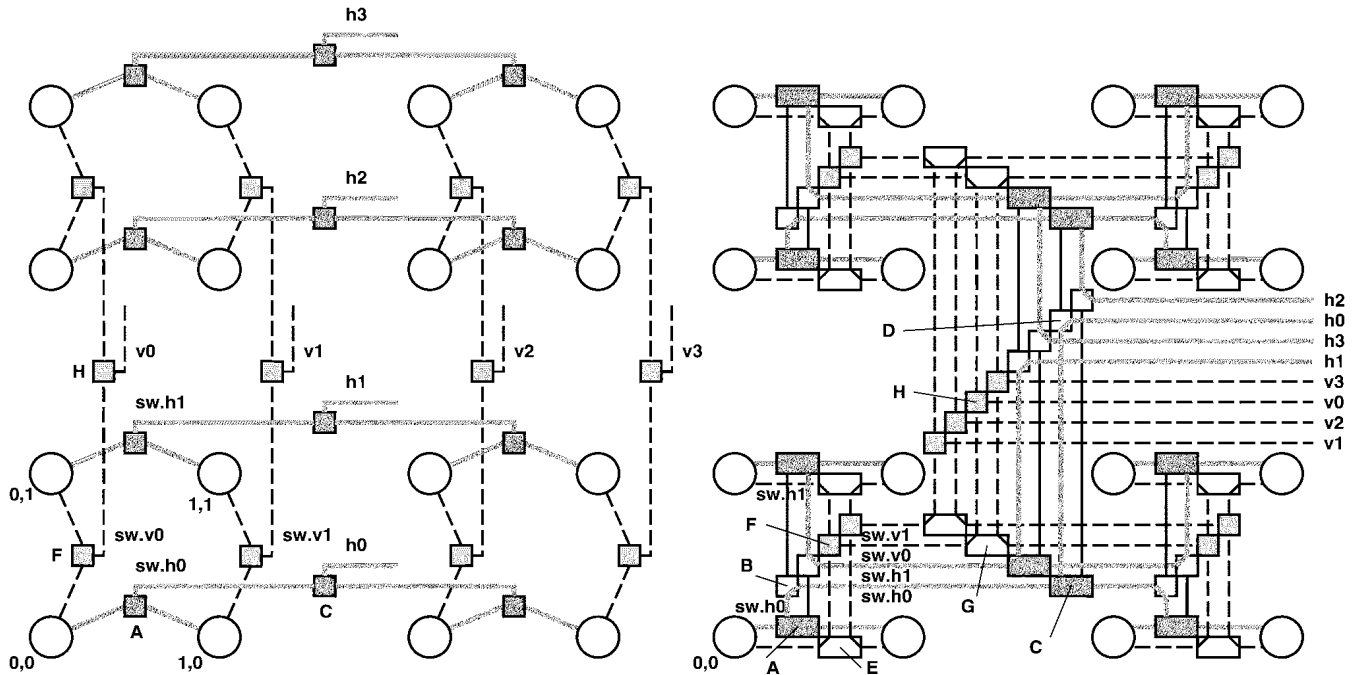
Fig. 7.   Mapping between $p = 0.5$ MoT and $p = 0.5$ HSRA (BFT).

Equation (11) gives us a lower bound on channel width which a mesh will need to support a Rent characterized (c, p) design. The mesh will generally need more wire channels than this because

- this only charges for bisection wires – channels may need to be wider to hold wires in the recursive cuts;
- this assumes optimal wire spreading – it may not be possible to spread wires evenly across all channels without increasing channel widths in the orthogonal channels.

Significantly, note that the mesh channel width, W, grows with N for $p > 0.5$. This is directly related to the same asymptote we saw in (7). The total wiring requirement grows faster than linear for $p > 0.5$. In [13], we further showed that switch and area requirements must grow faster than linear in N when $p > 0.5$ for traditional mesh organizations; since switches occupy space in the substrate, this prevents us from using multilayer metallization in order to fully combat the superlinear wire growth requirements; the density of our meshes will asymptotically decrease with increasing N.

While the mesh is perhaps the most studied and most commercially exploited topology, it has the worst asymptotic growth requirements of the three designs. This is a salient example where it is important to know that the asymptotic savings which the MoT and ToM achieve is at the cost of only a constant factor in wiring compared to the mesh; this says that even if the mesh design is smallest at a particular time, if the Rent relation continues to hold while chip and system capacities continue to grow, the mesh will eventually be the largest design.

III. MAPPING MoT TO LINEAR POPULATION ToM

We start by showing there is a direct mapping between a MoT with a given growth rate p and an HSRA/BFT with the same growth rate. Leighton observed that a MoT could be embedded in a ToM [23] where both are implicitly assumed to have $p = 0.5$. We observe that the mapping holds even if the ToM is linearly populated with switches as in the BFT or HSRA and that the mapping will hold for $0.5 \leq p \leq 1.0$.

A. Mapping

The observation is simply that we can embed each horizontal MoT tree inside a single HSRA tree (See Fig. 7). Note that the horizontal tree connecting the lowest row of the MoT (trace $(0, 0) \rightarrow$ sw.$h0 \rightarrow h.0$) is mapped to a corresponding HSRA tree (marked with same labels). Switches A and C perform the same roles in both trees. HSRA switches B and D are set into a fixed configuration as shown so that switches A and C (and corresponding switches higher in the tree) are connected together to match the MoT topology.

Similarly, we can embed each vertical MoT tree inside a single HSRA tree. Here switches E and G in the HSRA link up switches F and H in the HSRA so they can serve as switches F and H in the MoT.

In both cases, switches in alternate tree stages in the HSRA are simply switched into a static position (e.g., B, D, E, and G, in the called-out example) to match the topology of the MoT, while the other tree switches directly provide the switching needed by the MoT (e.g., A, C, F, and H). The MoT and the HSRA both support arbitrary c values using multiple, disjoint trees—disjoint except at the leaf where they connect to the leaves. Since we use two HSRA trees to support each MoT tree, we see that every $(2c, p)$ HSRA contains within it a (c, p) MoT. Assuming the same arity (number of children links per switchbox; see Section V-A3), a MoT route will traverse twice as many switches when implemented on the HSRA.

B. Implications

This observation says that the number of base trees (c) required for a BFT/HSRA can never be more than a constant factor larger than that for a MoT. The factor of two in leaf channels will manifest themselves as a factor of two in both the horizontal and vertical width of the HSRA, or a factor of 4 total area due to channel width. Both designs require a number of switches which is linear in the number of endpoints nodes and c. This shows that the c's will be linearly related so the total switches will be within a constant factor of each other. One might have expected that the MoTs fuller exploitation of 2-D locality would have given it an asymptotic advantage compared to the ToM; this shows the advantage is, at most, a small constant factor.

The factor of two is an upper bound. The mapped route does not fully use the switches in the HSRA (e.g., B, D, E, G), rather it takes a route which exists in the MoT based on less switching options. As a result, it is likely that any given design will route with a smaller constant factor on the HSRA ($c_{hsra} < 2c_{mot}$).

This shows that if one were to come up with a particularly clever or fast way to place or route a MoT, there would be a direct way to use it for the BFT/HSRA. That is, we simply solve the problem quickly for the MoT, then use the equivalence embedding given in the previous section to identify the switch settings in the ToM necessary to implement the MoT route.

C. Technicalities

1) Leaf Composition: For the mapping to work directly, the HSRA must allow connections between trees in each leaf similar to MoT corner turns. A typical MoT network connection will route through both a horizontal and vertical tree, changing between a horizontal and vertical tree (a corner turn) at a common leaf node. Consequently a MoT route mapped to an HSRA will need to be able to exit one tree route at a leaf, switch to a different tree, and continue routing in that tree.

2) Matching Growth Rates (p's): For the simplest HSRA's and MoTs, we use arity-2 trees, and we approximate a given p by deciding whether each tree stage has single or multiple parents (e.g., in the HSRA shown on the right of Fig. 4, the lowest level tree switches have two parents, while the switches one level up have a single parent). In the single tree HSRA, for arity-2 we repeat base sequences of growths (g_i's)

$$N^p = \left(2^l\right)^p = 2^{lp} = g_0 \times g_1 \times g_2 \times \cdots g_k. \quad (12)$$

So, for $p = 0.5$, we use the sequence (2 1)*, for $p = 0.75$, the sequence (2 2 2 1)*. For the MoT, we have separate trees in every channel contributing to the total bisection bandwidth, and each growth spans both dimensions, so we have

$$N^p = \left(\left(2^l\right)^2\right)^p = 2^{2lp} = 2^l \times g_0 \times g_1 \times g_2 \times \cdots g_l. \quad (13)$$

The sequence (1)* realizes $p = 0.5$, and the sequence (2 1)* realizes $p = 0.75$. Redistributing the 2's

$$2^{2lp} = 2 \times g_0 \times 2 \times g_1 \times 2 \times g_2 \times \cdots \times 2g_l. \quad (14)$$

From this, we see that given a MoT growth sequence $g_{mot} = \{g_{mot_0}, g_{mot_1}, \cdots g_{mot_k}\}^*$, we can create an HSRA growth sequence

$$g_{hsra} = \{2, g_{mot_0}, 2, g_{mot_1}, \cdots 2, g_{mot_l}\}^*. \quad (15)$$

That is, the directly corresponding HSRA sequence includes a two before every growth factor in the MoT sequence. Thus our $p = 0.5$, (1)* MoT sequence yields our (2 1)* HSRA sequence, and our $p = 0.75$, (2 1)* MoT sequence yields our (2 2 2 1)* HSRA sequence. This arises because the MoT always effectively doubles its bandwidth in the nontree dimension simply by aggregating all the tree wires in the orthogonal channels. These are exactly the wires which have fixed switch configurations in the mapping above (see Fig. 7).

One consequence of this is that the directly mapped HSRA growth sequence for a given p corresponds to the HSRA growth sequence derived from the MoT sequence. In many cases this is the same (e.g., for $p = 0.5$: MoT(1)* $\rightarrow$ HSRA(2 1)*, for $p = 0.75$: MoT(2 1)* $\rightarrow$ HSRA(2 2 2 1)*). However, for some sequences there is a simpler growth sequence which one might use on the HSRA. For example, for $p = 2/3$, the simplest MoT sequence is (2 1 1)*. The corresponding mapped HSRA sequence is (2 2 2 1 2 1)*. However, the sequence (2 2 1)* is a simpler growth sequence often used for the HSRA. If we do not use corresponding sequences in the mapping, the embedding may require a larger ratio between c_{hsra} and c_{mot}. Nonetheless, the ratio will remain a constant.

IV. MAPPING LINEAR POPULATION ToM TO MoT

Embedding the MoT in the HSRA made it clear that the MoT has a subset of the connectivity of the HSRA. We want to identify exactly what the difference between these two networks is. *What do we have to add to transform the MoT into the HSRA?*

A. MoT Augmentation

Figs. 8 and 10 show that we need to add a strategic set of orthogonal interchanges to the trees of a single dimension of the MoT in order to achieve HSRA-equivalent connectivity. As shown in Fig. 9, we decompose the MoT into horizontal and vertical channels and concentrate on additions to the horizontal channels. We add vertical links between corresponding switching nodes in different channels (see Fig. 8). Here, "corresponding" means that a switching node at level l is connected to the switching node at the same logical tree point (same logical set of decisions among up links when there is growth) 2^l channels above (below) it.

The additional wires turn the single child per side, single parent switching nodes into 5-way switches instead of 3-way switches [the 3-way switches in Fig. 9 (S_3) turn into 5-way switches like the one shown on the right of Fig. 10 (S_5)], and turn the double parent switches into 6-way switches [see bottom level switches in the MoT on the top right of Fig. 12 (S_6)]. As shown in Fig. 10, we can reorganize the HSRA switching so that it fits inside these augmented MoT switching units while retaining all of the HSRA connectivity. This switch regrouping
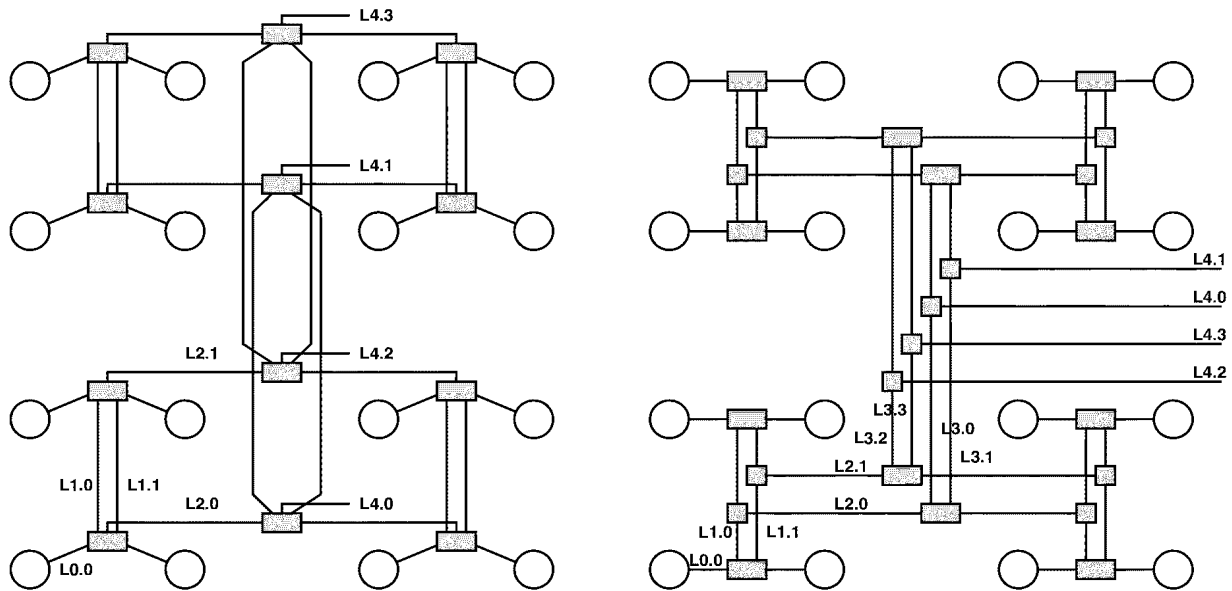
Fig. 8.   Mapping between $p = 0.5$ HSRA and $p = 0.5$ augmented MoT.
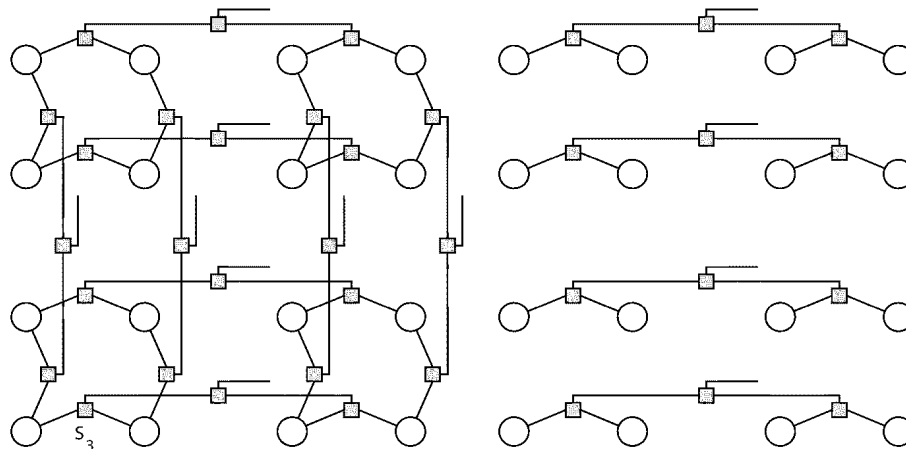
Fig. 9.   Extract horizontal connectivity from $p = 0.5$ MoT.
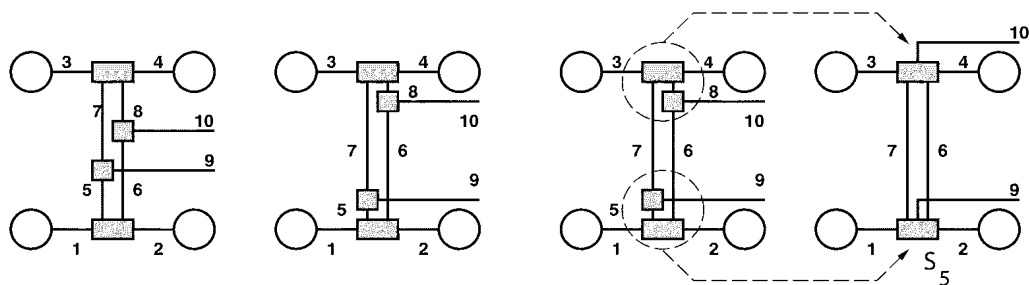
Fig. 10.   Equivalence of augmented MoT switching and HSRA switching.

added to the vertical link topology recovers for us HSRA connectivity for any size HSRA. Fig. 8 marks the resulting wire correspondence.

In this transformation we simply replace every existing switching unit with one which is a constant factor larger. The net effect is to increase the total number of switches by a constant factor. The total number of switches required for this augmented MoT remains linear in the number of endpoints supported.

B. MoT Layout

Fig. 8 shows how the MoT implements the HSRA. What is not immediately demonstrated in such diagrams is how these extra wires will be laid out in the MoT. Most importantly, *when the HSRA-augmenting connections in the MoT are placed, what is the maximum channel width and maximum number of switches per node?* It turns out that we can distribute these augmented connections across the span of a hierarchical MoT segment so that there are a constant number of switches per
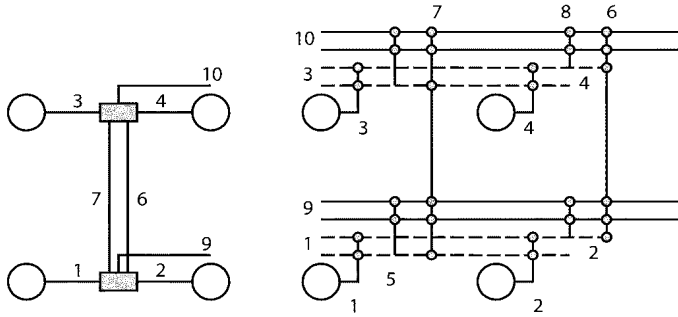
Fig. 11.   Channel layout view and signal mapping for augmented MoT.

endpoint and that the number of wires per channel grows at exactly the same rate as the MoT channel wires (see middle right of Fig. 12). Further, by using the existing wire segments in the orthogonal routing channels, which we have been ignoring so far in our equivalence argument, no new wires are needed—we simply need to apply an additional set of switches which allow us to use the existing wires in this manner (see bottom row of Fig. 12).

To show switch and wire spreading, it is useful to view a more detailed view of the MoT/HSRA layout which shows channel runs and switches. Fig. 11 shows such a view alongside the logical view for a $p = 0.5$ MoT augmented with these HSRA links. Note here that we actually use a pair of MoT wires to implement each single wire into the switchboxes in order to get the full connectivity of the HSRA switching. In particular, this allows a full interchange (e.g., $3 \rightarrow 2$, $4 \rightarrow 1$) which would not be possible if we only used a single wire. The need for two wires arises because the MoT wire is not segmented and switched at the switchbox, as in the HSRA, but rather is a continuous run and we effectively spread out the switches in the HSRA switchbox along the length of the pair of wires.

Fig. 12 shows an augmented MoT network for $p = 0.75$ alongside a $p = 0.75$ HSRA. The $p = 0.75$ case makes clearer the fact that we cannot run all the wires directly in the place where they are shown in the equivalence diagram (see top right of Fig. 12) without filling the channels unevenly. In fact, there will be $O(N^p)$ such connections at the top of the tree, whereas the MoT layout has already spread out the existing $O(N^p)$ **total** wires in its bisection among the $\sqrt{N}$ channels such that there are only $O(N^{(p-0.5)})$ wires per channel and demonstrated that these can be laid out in constant width per channel given $\Theta(N^{(p-0.5)})$ wire layers [13] [reviewed below cumulating in (22)].

Fig. 12 (middle row) shows a channel layout view for the augmented MoT. We use a $c = 2$ MoT to accommodate a $c = 1$ HSRA as suggested above. We note here that the eight wires which had crossed the bisection are now spread out so that there are two wires in each of the four channels. This is accomplished in exactly the same way we guaranteed there were only $O(1)$ switches at each endpoint [13]. Here it is important that we maximally spread out uplinks at a given level so that we do not get multiple links to the same level at the same endpoint; the geometric reduction in uplinks (wire) per endpoint as we ascend the tree guarantees that this is easy to accomplish. In

fact, once we spread out the uplinks properly, we use the placement of the parent–child uplink switches as a guide for where to place these crosslink connections. Every place we have an uplink switch, we place a companion augmenting link to the associated wire in the companion stage (2^l channels above or below as previously identified). In this way, we roughly double the number of switches at each endpoint. Unlike switches, the wires do overlap. That means the number of wires per channel will grow as longer wires overlap shorter ones. The trick is to notice that the wire growth exactly matches the standard wire channel growth so that we can use existing wires for these runs.

Constructively, we note that we have a total of $g_0 \times g_1 \times \cdots g_l$ uplinks at the root of a height l row or column tree. Rearranging (13)

$$N_{up}(l) = g_0 \times g_1 \times \cdots g_l = 2^{2lp-l} = 2^{2l(p-0.5)}. \quad (16)$$

To convert this to a per channel uplink count, we have divided out the 2^l factor which results from combining across the 2^l channels contributing uplinks. Equation (16) is exactly the per channel row or column width at level l necessary to satisfy our Rent relation

$$W_{mot}(l) = \frac{N^p}{2^l} = \frac{2^{2lp}}{2^l} = 2^{2l(p-0.5)}. \quad (17)$$

These uplinks are distributed across the segment span of length 2^l, so each node gets

$$N_{up\_per\_node}(l) = \frac{N_{up}}{2^l} = \frac{2^{2l(p-0.5)}}{2^l} = 2^{2l(p-1)}. \quad (18)$$

The augmenting wires span length 2^l. Wire channel width contribution per level then is

$$W_{mot_{augment}}(l) = 2^l N_{up\_per\_node}(l) = 2^{2l(p-0.5)}. \quad (19)$$

As suggested, this shows the same wire requirements as the MoT needed for this level (17).

We further note that the total width of either channel is

$$W_{mot} = \sum_{l=0}^{l=\log(\sqrt{N})} \left(W_{mot}(l) \right) \quad (20)$$

$$W_{mot} = \sum_{l=0}^{l=\log(\sqrt{N})} \left(2^{2l(p-0.5)} \right) \quad (21)$$

$$W_{mot} = 2^0 + 2^{2(p-0.5)} + 2^{4(p-0.5)} + \cdots$$
$$\qquad + 2^{2\log(\sqrt{N})(p-0.5)}$$
$$= 2^0 + 2^{2(p-0.5)} + 2^{4(p-0.5)} + \cdots + 2^{\log(N)(p-0.5)}$$
$$= 2^0 + 2^{2(p-0.5)} + 2^{4(p-0.5)} + \cdots + N^{(p-0.5)}$$
$$= N^{(p-0.5)} + \left(\frac{N}{2}\right)^{(p-0.5)} + \left(\frac{N}{4}\right)^{(p-0.5)} + \cdots + 1$$
$$= N^{(p-0.5)} \left(1 + \frac{1}{2^{(p-0.5)}} + \frac{1}{2^{2(p-0.5)}} + \cdots \frac{1}{N^{(p-0.5)}} \right)$$
$$< N^{(p-0.5)} \left(\frac{1}{1 - \left(\frac{1}{2}\right)^{(p-0.5)}} \right). \quad (22)$$
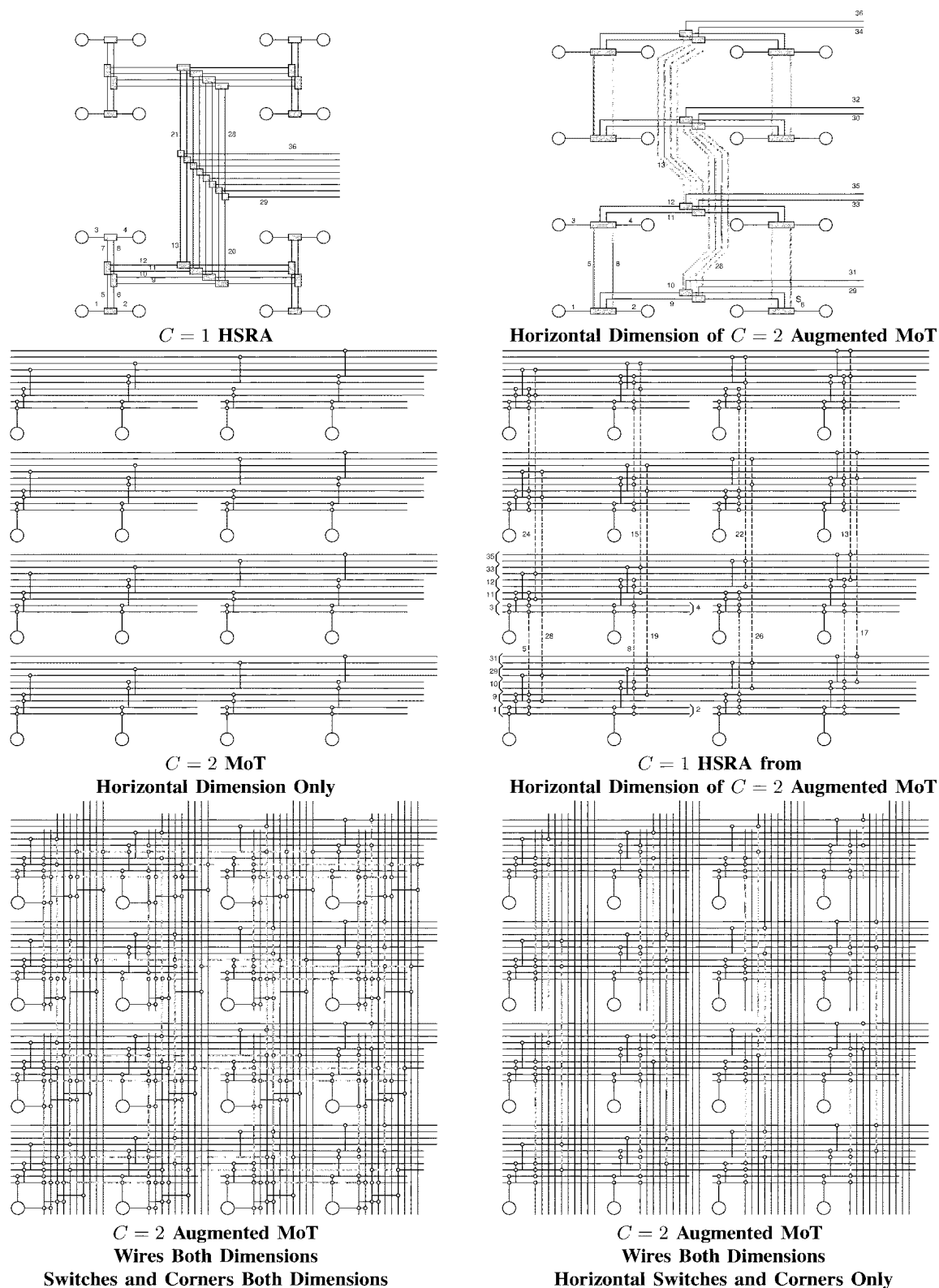
Fig. 12.   Mapping between $c = 1, p = 0.75$ HSRA and $c = 2, p = 0.75$ augmented MoT.

For $p > 0.5$, $(1/2)^{(p-0.5)} < 1$, so the sum converges to a p-dependent constant times $N^{(p-0.5)}$, which is within a constant factor of the mesh channel width lower bound (11).

Since we have noted that the number of wires added for a stage of augmenting links is exactly the same as the number of wires in the parent stage to which they are connecting; and since we are using a $c = 2$ MoT, we can use the wires in the corresponding stage of the orthogonal tree to perform this connection simply by adding the switches necessary to allow them to serve as these augmenting links. The bottom row of Fig. 12 shows the
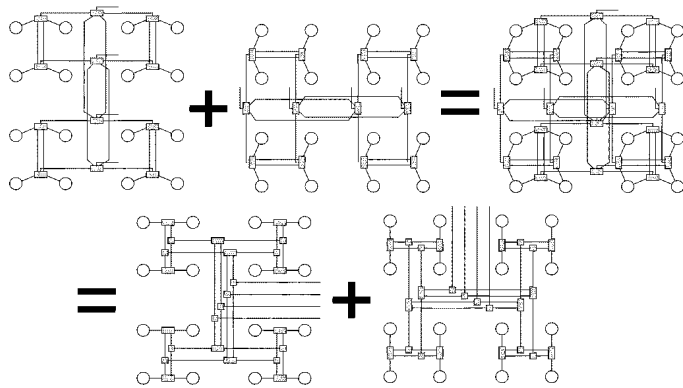
Fig. 13. $c = 4$, $p = 0.5$ augmented MoT in both dimensions corresponds to composing two $c = 1$, $p = 0.5$ HSRAs rotated $90°$ relative to each other.

additional switches and highlights where the augmented paths run over the orthogonal wire runs.

We have shown how to make one-dimension of a $(c_{\mathrm{mot}} = 2c_{\mathrm{hsra}}, p)$ MoT contain a (c_{hsra}, p) HSRA. We can populate the augmenting link switches in both dimensions as shown in Fig. 12 (bottom left). This makes a $c = 4$ MoT contain $c = 1$ HSRA composed with its transpose (see Fig. 13).

C. Implications

1) Design Unification: The augmentation that turns the MoT into the BFT/HSRA allows us to exactly understand the difference between these two networks. We can now see the BFT as a particular corner turn scheme applied to the MoT and unify these two networks into a single, parameterized design.

How we might add links between horizontal and vertical tracks in the MoT (corner turns) was an outstanding question before we discovered this mapping. If we simply allowed every wire to connect to the wires at the same tree level that cross it, we would need an asymptotically growing number of switches per node and would lose the linear switch bound of the MoT. The BFT/HSRA wiring gives us insight into how to formulate a limited corner turn scheme for the MoT. This corner turn scheme does not asymptotically increase the switches or wires in the MoT, but does provide interesting switching character- istics. Since the BFT/HSRA only has to route up and down a single tree, whereas the MoT without augmentation generally has to route up and down two trees, the augmented MoT has half the switches in the worst-case paths between a source and a destination.

A common complaint leveled at the arity–2 HSRA topology is the asymmetry between the horizontal and vertical connec- tions; as shown on the right of Fig. 4, the horizontal nearest- neighbor is one switch away while the vertical nearest-neighbor is two. The equivalence in Fig. 13 makes it clear that the arity-2 HSRA directionality bias can be removed by overlaying the net- work with its transpose. Since all cases where we use MoT and HSRA networks for FPGAs have $c > 1$, we will always have multiple trees and be able to alter the orientation of the trees rel- ative to each other. This equivalence also makes it clearer that the MoT staggering can be applied to the BFT/HSRA.

This mapping further shows us how we can apply any results on fast HSRA mapping (e.g., [30]) to the MoT.

TABLE I
SUMMARY OF RECURRING SYMBOLS USED IN ARTICLE

Symbol	Description	Intro
A_{wire}	Lower bound on chip area	Eq. 7
	due to wiring requirements	
BW	Bisection Width	Sec. II
BW_{mesh}	Wires in Bisection for Mesh Topology	Eq. 8
c	Constant in Rent's Rule	Eq. 1
c_{hsra}	Number of Base Channels in HSRA (Fig. 4)	Sec. III-B
c_{mot}	Number of Base Channels in MoT (Fig. 5)	Sec. III-B
g_i	Growth rate at level i in MoT/ToM Tree	Eq. 12
g_{hsra}	Growth Sequence for HSRA/ToM Trees	Eq. 15
g_{mot}	Growth Sequence for MoT Trees	Eq. 16
IO	Total input/output from a region	Eq. 1
l	Level or Number of levels in MoT or ToM Tree	Eq. 12
L_h	Side length of chip in horizontal direction	Eq. 6
L_v	Side length of chip in vertical direction	Eq. 4
N	Number of Primitive elements in a region	Eq. 1
N_h	Number of horizontal wiring layers	Eq. 6
N_{up}	Number of uplinks at a specified level	Eq. 16
N_v	Number of vertical wiring layers	Eq. 4
p	Growth exponent in Rent's Rule	Eq. 1
W	Mesh Channel Width (Fig. 6)	Eq. 8
W_{mot}	Width of MoT Channel	Eq. 22
W_{pitch}	Wire Pitch	Eq. 4

TABLE II
COMMON PARAMETERIZATION AND ANALOGS

MoT/BFT/HSRA	Manhattan Mesh	Note
Flatness	complete	Standard mesh has flat connectivity up all row and column trees
Arity	L_{seg}	generally a distribution of segment lengths
Base Channels c	W	W per segment length (L_{seg})
Growth p		may be a function of c,p
Shortcuts	Switch Box	Mesh designs typically
Corner Turns	Population	assume all present
Staggering		Same idea
Domains		Similar issues

2) Asymptotics and Layout: We see from (17), (19), (22), and (11), that the $p > 0.5$ MoT, the $p > 0.5$ augmented MoT (or the HSRA), and the mesh have the same asymptotic channel width. None of the networks has more than a constant factor fewer wiring tracks than any of the others.

The equivalence transformation here tells us we can apply what we know about MoT layouts to HSRA layouts. Signifi- cantly, the construction above showed that the HSRA can be laid out in asymptotically the same channel width as the MoT. We previously showed that a $p = 0.5$ BFT/HSRA could be laid out in linear area given $\Theta(\log(N))$ wire layers [28] ; but at that point in time the general question of laying out a BFT/HSRA $(1.0 > p > 0.5)$ in linear area using multilayer metallization remained open. The equivalence above allows us to exploit our prior construction that showed how to layout the MoT for any $p > 0.5$ in linear space using $\Theta(N^{(p-0.5)})$ wire layers [13] in order to also layout any HSRA in linear two-dimensional area using $\Theta(N^{(p-0.5)})$ wire layers. This now gives us two networks that have the $\Theta(N)$ layout area property.

V. MoT to Mesh Parameterization

In a companion article [13], we compared the MoT to a con- ventional, Manhattan mesh. The most fundamental difference

No Flattening ($f = 1$)   **Flattening ($f = 2$)**   **Flattening ($f = 4$)**
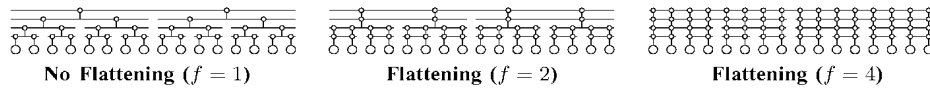
Fig. 14.   Flatness parameter shown on single row (column) channel in $p = 0.5$ MoT.
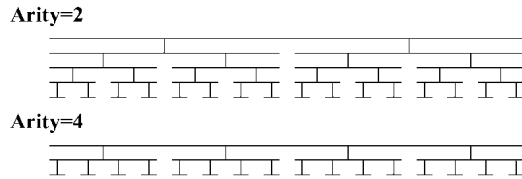
Arity=2

Arity=4

Fig. 15.   Arity parameter on single row (column) channel in $p = 0.5$ MoT.

No Staggering

Staggered

Fig. 16.   Staggering single row (column) channel in $p = 0.5$ MoT.

between the Manhattan mesh and the MoT is the flat endpoint connectivity on the mesh. That is, the mesh C-Box connects the compute element's inputs or outputs to all of (a constant fraction of) the wires in the channel, whereas the MoT only connects to the base level tree channels and uses the tree connections to climb up the tree to reach longer segments. This has the immediate impact that the MoT needs only a linear number of switches, while the Mesh needs $O(W)$ switches per endpoint. Since we have already established that W grows with N for $p > 0.5$ [(11)], we see that the mesh requires asymptotically more switches than the MoT.

We can parameterize this difference and the other traditional differences between the MoT and the mesh in order to define a continuum space between the extremes. Table II summarizes the variables. As noted in Fig. 1, we can view a mesh as a special, degenerate case of the MoT where we tune several of the parameters to their extreme values.

A. Parameterization

1) Flatness: We parameterize flatness by the number of parent tree levels to which we connect each child node. In the MoT, we connect a child at level l to a parent at level $l + 1$. In the Mesh, we connect the leaf child at level 0 to all levels above it. In general, we could provide direct connections among a group of f levels; that is, we connect a child at level $n \cdot f$ to levels $n \cdot f + 1, n \cdot f + 2, \ldots n \cdot f + f = (n+1) \cdot f$ (see Fig. 14).

2) Segment Distribution: The Rent relation (1) can be applied strictly to define a set of segment distributions. We see from the MoT designs, that we get c length 1 segment, $c \times g_{\mathrm{mot}_0}$, length 2 segments, $c \times g_{\mathrm{mot}_0} \times g_{\mathrm{mot}_1}$, length 4 segments, and so on. Recall from (13), that we pick the growth rates to correspond to our target p value. This same idea could be applied to the selection of mesh segment lengths and segment length distributions. As we noted earlier [13], if these lengths are chosen geometrically in this manner, and if corner turns are only allowed at segment ends, the mesh only needs a total number of S-Box switches which is linear in the number of nodes supported by the design.

Conventional mesh designs have often chosen to truncate their hierarchy—stopping after a given segment length or jumping from one segment length to full row/column length lines rather than including all of the geometric wire lengths.

3) Arity: For simplicity, we have, so far, described and shown binary trees for the MoT and HSRA. We showed an
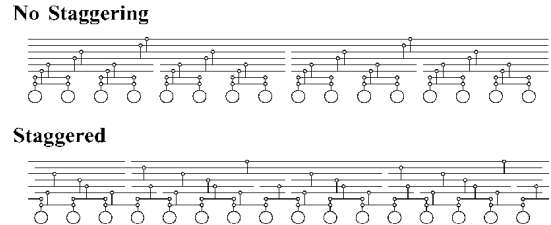
arity-4 BFT in Fig. 4. In general, we can build trees with any number of children levels to a parent level. For example, Fig. 15 shows a MoT row with an arity of 4 as contrasted with an arity-2 MoT row. The arity tunes the rate of segment growth. So an arity-4 MoT has segments of length $1, 4, 16 \ldots$ rather than $1, 2, 4, 8, 16 \ldots$ In this way the combination of arity and p defines segment distribution.

4) Staggering: When we have multiple segments of nonunit length longer than one, it is useful, both for switch placement and for routing, to spread out the switch placements (e.g., [31]–[33]). In the MoT, this is true as well [13]. For the MoT, we stagger the alignment of the trees relative to each other (see Fig. 16). In both cases, staggering minimizes the cases where a route must use a significantly longer (higher) link than it should take to span the distance between the source and sink.

5) Shortcuts: In the strict tree structure of the MoT and BFT/HSRA, there are cases where two nodes are physically close in the layout but logically distant in the tree. This effect is mitigated by staggering. It can be eliminated entirely by adding shortcut connections which allow segments at the same level and in the same channel to be connected to their immediately adjacent neighbors. These shortcuts, which only requires a constant factor more switches than the base MoT, now guarantee that the physical distance one must travel in the MoT or BFT/HSRA is never more than a constant factor larger than the Manhattan distance; this was the key innovation of the Fat Pyramid [34]. Further, the number of switches on the path will be logarithmic in distance, making it asymptotically fewer than any bounded-segment length mesh scheme. These shortcuts perform exactly the same switching as the end-to-end segment switching (E $\leftrightarrow$ W, N $\leftrightarrow$ S) which appears in the switchpoints of standard, Manhattan, switchbox designs (see Fig. 17). That is, in the standard diamond switchbox, the switch which connects a segment to a single segment of the same length in the same channel on the other side of the switchbox, is essentially the same as the shortcut switches which we may or may not include in MoT or ToM designs.

6) Corner Turns: Corner turn parameterization defines where and how routes may turn between orthogonal channels (from horizontal to vertical routing or *vice-versa*). In a standard mesh switchbox, a segment has a corner turning switch to a single orthogonal segment when it crosses that segment or to
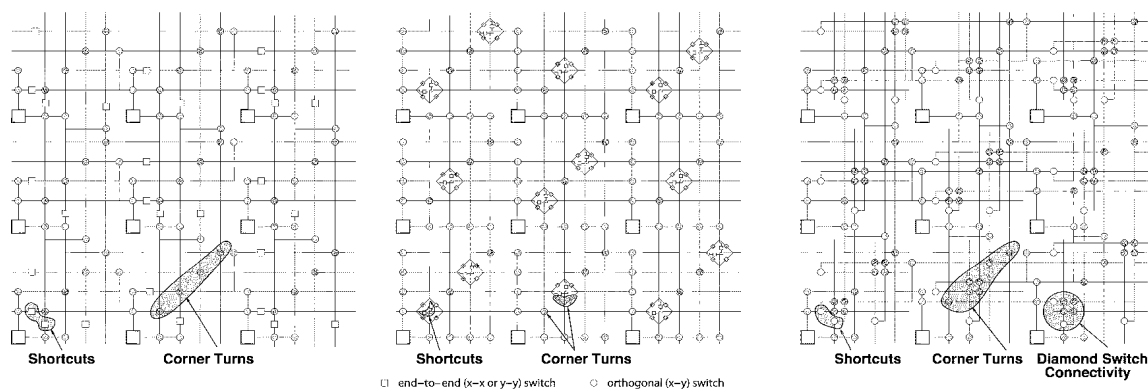
Fig. 17.   Shortcut, corner turn, and switchbox population.

one segment in each direction when it arrives at a switchbox coincident with a segment break in its corresponding orthogonal segment. In a standard diamond switch configuration, these are the NW, NE, SW, and SE switches (see Fig. 17). These corner turns make up the remaining 1–2 switches which are normally attached to the end of each segment (compare $F_s = 3$ in the Toronto Model [3], [4]).

Since W is growing with N for $p > 0.5$ (11), as long as we have at least one corner turn per segment per switchbox, the number of switches per switchbox is growing, so the total number of switches in all the switchboxes are growing faster than linearly. It is asymptotically desirable to avoid this level of corner turn population. The fact that we can lay out MoT and BFT/HSRA designs using asymptotically similar wiring requirements but without such extreme corner turn population certainly suggests there is a viable alternative and it will eventually be beneficial to exploit it. In the MoT, we consider whether corner turns should be limited to the leaves or whether we need some, limited scheme for higher level corner turns (Section IV-C1 and [13]). In general, all the Mesh and MoT corner turn variations make up a rich parameterized design space.

The typical MoT layout, shown on the left of Fig. 17, allows only a single corner turn at each switchbox. On the right of Fig. 17, we show an alternate layout that overlaps adjacent segments in the same channel so that we can use simple switches between orthogonal lines to support corner turns and allow this inclusion of the pair of corner turns (e.g., both NE and NW from the north input to a switchbox) typical in mesh switch populations.

7) Switchbox Population and Domains: It should be clear that the general issues, which the mesh considers in terms of switchbox population can be decomposed into shortcut and corner turn issues above and shown in Fig. 17. In the preceding, we described the most popular mesh and MoT designs where there are a constant number of wires connecting the end (or internal points in the mesh case) of segments. More generally, we might ask about fuller switchbox population (e.g.,[35], [25]) and there are similar questions in both designs. The traditional mesh design has disjoint domains which are only connected at the leaves; similarly, the primary MoTs we have described only allow turns between separate row and column trees at the leaves and typically can only change among corresponding row and

column trees at a corner turn, giving them this domain structure as well.

B. Implications

1) Unified Design Space: Unifying the design space gives us greater insight into how we can tune designs. Reconciling the Mesh with the MoT introduces new design parameters to explore for tuning the Mesh and the MoT. It also sheds some light on the assumptions we tend to make in these designs—assumptions which may merit reexamination, e.g., would the MoT be better off adopting the rich segment endpoint switching of a typical mesh, or would the mesh be better off omitting some of these switches? Is flat-endpoint connectivity a good assumption to keep as we scale up to million compute-node designs? Note that the mesh's flat-endpoint connectivity and full corner-turn population are each, individually, sufficient to force the number of switches in the mesh to grow superlinearly in the number of nodes. The MoT shows us a plausible alternative to avoid the asymptotic growth arising from flat-endpoint connectivity, and the ToM to MoT mapping show us a plausible alternative to full corner-turn population.

2) Shortcuts: Our experience with MoT designs to date suggests that shortcut connections may offer marginal additional value compared to staggering [13]. Shortcuts do reduce the total channel width required to route the design when we do not have staggered segments, but only at a net increase in the total number of switches. Once we add staggering, even the wire reduction benefit is marginal. It may be interesting to consider shortcut depopulation in the Manhattan mesh.

3) Pre- and Post-Layout Rent Characteristics: As shown in the previous section, we can layout a BFT/HSRA with asymptotically the same channel width as a Mesh. We can use the same layout strategy for the ToM, giving us a generalization of Leiserson and Greenberg's fold-and-squash layout [36]. We know the ToM can accommodate layouts simply by recursive bisection. As long as the bisection cuts do not exceed the tree bandwidths, the recursive bisection design will be routable on the ToM, which we now know can be laid out with asymptotically the same channel width as the mesh.

Put together, these observations imply that the *a posteriori* global route Rent exponent for a Manhattan layout should be the same as the *a prior* Rent exponent. That is, while there may be difference in the layout-based partitions (e.g.,[15], [16]), these

should, at most, be placement shuffles to reduce the constant factors associated with tree overlap among tree levels and will not change the asymptotic growth rate. The MoT and ToM layout described above tell us how to take any Rent characterized (c, p) design and lay it out with $O\left(cN^{(p-0.5)}\right)$ Manhattan channel width. This gives us an upper bound on the global channel width required to route a (c, p) design on a mesh; this upper bound is within a constant factor of the lower bound we previously derived on mesh channel width ((11)). Note that the $c_{\text{mot}} = 2c_{\text{hsra}}$ construction above already more than accounts for the downlink conflicts that forced us to use a $(1.5c, p)$ ToM to accommodate a (c, p) design, so the channel width is no higher than $2c$ times the per channel width of the MoT derived in (22)

$$
\begin{aligned}
W &\leq 2cN^{(p-0.5)}\left(\frac{1}{1-\left(\frac{1}{2}\right)^{(p-0.5)}}\right) \\
&\leq \left(\frac{2^{(p+0.5)}}{2^{(p-0.5)}-1}\right) cN^{(p-0.5)}.
\end{aligned}
\tag{23}
$$

This suggests that there is no fundamental reason for the post-placement Rent exponent for a design to be larger than the pre-placement Rent. However, while asymptotically tight, the bounds are loose in absolute terms; for example, the ratio between the lower bound in (11) and this upper bound ((23)) is around 30 for $p = 2/3$. Consequently, this leaves room for large constant factor differences between pre-placement and post-placement IO ratios, and it may take very large designs for the asymptotic effects to dominate. Our construction is unlikely to be the tightest possible, so we leave open the question of how much it is possible to tighten the constant factors in this mapping.

The guarantee above is made in terms of a global route and the full population ToM, rather than the mesh detailed route and the HSRA or BFT, because the mapping ratio for linear population designs remains an important, open question [35], [25]. The result is directly applicable more to custom routing than FPGA routing for this reason. However, if we can establish a constant mapping ratio for some variant of the HSRA/BFT, then these observations would allow us to apply this result to these detailed networks as well.

VI. SUMMARY

To build efficient switching networks for typical circuits, we must use networks which allow us to exploit the locality structure which exists in these circuit graphs. Manhattan meshes, MoT, and ToM style networks are all examples of limited-bisection switching networks which support this locality exploitation. While these networks are different in formulation, we see that they have the same asymptotic wiring requirements—all requiring $O(N^{(p-0.5)})$ wires per channel in 2-D layouts when $p > 0.5$. Using embeddings, we have demonstrated equivalence mappings between the networks (MoT embedded in HSRA, HSRA embedded in augmented MoT with corner turns, and MoT embedded in 2-D mesh); all of these mappings require at most a constant scale factor in wires. The MoT to ToM and ToM to MoT embeddings are made with only a constant scale factor in switches. From these mappings we now see how to layout

linear-population ToM designs of any p (e.g., BFT and HSRA) in constant area using multilayer metallization and how to produce constructive global mesh routes which are known to be within a constant factor of optimal. We can view these networks within a larger, unified design space which helps us understand the tradeoffs which each network makes and aides our search for network parameters which meet design goals.

REFERENCES

[1] W. S. Carter, K. Duong, R. H. Freeman, H.-C Hsieh, J. Y. Ja, J. E. Mahoney, L. T. Ngo, and S. L. Sze, "A user programmable reconfigurable logic array," in *Proc. IEEE 1986 Custom Integrated Circuits Conf.*, May 1986, pp. 233–235.

[2] S. Trimberger, "A reprogrammable gate array and applications," *Proc. IEEE*, vol. 81, pp. 1030–1041, July 1993.

[3] J. Rose and S. Brown, "Flexibility of interconnection structures for field-programmable gate arrays," *IEEE J. Solid-State Circuits*, vol. 26, pp. 277–282, Mar. 1991.

[4] S. D. Brown, R. J. Francis, J. Rose, and Z. G. Vranesic, *Field-Programmable Gate Arrays*. Norwell, MA: Kluwer, 1992.

[5] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*. Norwell, MA: Kluwer, 1999.

[6] *FLEX 8000 Handbook*, Altera Corporation, San Jose, CA, 1994.

[7] A. A. Agarwal and D. Lewis, "Routing architectures for hierarchical field programmable gate arrays," in *Proc. 1994 IEEE Int. Conf. Computer Design*, Oct. 1994, pp. 475–478.

[8] V. C. Chan and D. Lewis, "Area-speed tradeoffs for hierarchical field-programmable gate arrays," in *Proc. Int. Symp. Field-Programmable Gate-Arrays*, Feb. 1996, pp. 51–57.

[9] Y.-T. Lai and P.-T. Wang, "Hierarchical interconnection structures for field programmable gate arrays," *IEEE Trans. VLSI Syst.*, vol. 5, pp. 186–196, June 1997.

[10] A. DeHon, "Balancing interconnect and computation in a reconfigurable computing array (or, why you don't really want 100% LUT utilization)," in *Proc. Int. Symp. Field-Programmable Gate-Arrays*, Feb. 1999, pp. 69–78.

[11] W. Tsu, K. Macy, A. Joshi, R. Huang, N. Walker, T. Tung, O. Rowhani, V. George, J. Wawrzynek, and A. DeHon, "HSRA: High-speed, hierarchical synchronous reconfigurable array," in *Proc. Int. Symp. Field-Programmable Gate-Arrays*, Feb. 1999, pp. 125–134.

[12] R. Sidhu, S. Wadwha, A. Mei, and V. K. Prasanna, "A self-reconfigurable gate array architecture," in *Proc. Int. Conf. Field-Programmable Logic Applications*, Villach, Austria, Aug. 2000, pp. 106–120.

[13] R. Rubin and A. DeHon, "Design of FPGA interconnect for multilevel metallization," *IEEE Trans. VLSI Syst.*, vol. 12, pp. 1038–1050, Oct. 2004.

[14] A. E. Caldwell, A. B. Kahng, and I. L. Markov, "Can recursive bisection produce routable placements?," in *Proc. IEEE Design Automation Conf.*, 2000, pp. 477–482.

[15] P. Verplaetse, J. Dambre, D. Stroobandt, and J. V. Campenhout, "On partitioning vs. placement rent properties," in *Proc. 2001 Int. Workshop on System-Level Interconnect Prediction (SLIP)*, Mar. 2001, pp. 33–40.

[16] X. Yang, E. Bozorgzadeh, and M. Sarrafzadeh, "Wirelength estimation based on rent exponents of partitioning and placement," in *Proc. 2001 Int. Workshop System-Level Interconnect Prediction*, Mar. 2001, pp. 25–31.

[17] A. DeHon and J. Wawrzynek, "Reconfigurable computing: What, why, and design automation requirements?," in *Proc. 1999 Design Automation Conf.*, June 1999, pp. 610–615.

[18] S. W. Gehring and S. H.-M. Ludwig, "Fast integrated tools for circuit design with FPGAs," in *Proc. Int. Symp. Field-Programmable Gate-Arrays*, Feb. 1998, pp. 133–139.

[19] R. Tessier and H. Giza, "Balancing logic utilization and area efficiency in FPGAs," in *Proc. Int. Conf. Field-Programmable Logic and Applications*, R. W. Hartenstein and H. Grübacher, Eds., Aug. 2000, pp. 535–544.

[20] C. Thompson, "Area-time complexity for VLSI," in *Proc. 11th Annual ACM Symp. Theory Computing*, May 1979, pp. 81–88.

[21] B. S. Landman and R. L. Russo, "On pin versus block relationship for partitions of logic circuits," *IEEE Trans. Computers*, vol. 20, pp. 1469–1479, 1971.

[22] D. Stroobandt, "On an efficient method for estimating the interconnection complexity of designs and the existence of region iii in Rent's rule," in *Proc. Great Lakes Symp. VLSI*, Mar. 1999, pp. 330–331.

[23] F. T. Leighton, "New lower bound techniques for VLSI," in *Proc. Annual Symp. Foundations Computer Science*, Nashville, TN, 1981, pp. 1–12.

[24] S. Bhatt and F. T. Leighton, "A framework for solving VLSI graph layout problems," *J. Comput. Syst. Sci.*, vol. 28, pp. 300–343, 1984.

[25] A. DeHon, "Rent's rule based switching requirements," in *Proc. System-Level Interconnect Prediction Workshop* , Mar. 2001, pp. 197–204.

[26] C. E. Leiserson, "Fat-trees: Universal networks for hardware efficient supercomputing," *IEEE Trans. Computers*, vol. C-34, pp. 892–901, Oct. 1985.

[27] R. I. Greenberg and C. E. Leiserson, "Randomized routing on fat-trees," in *Randomness in Computation*, MIT/LCS/TM-307 ed.   Greenwich, CT: JAI Press, 1988, vol. 5, Advances in Computing Research.

[28] A. DeHon, "Compact, multilayer layout for butterfly fat-tree," in *Proc. 12th ACM Symp. Parallel Algorithms Architectures*, July 2000, pp. 206–215.

[29] F. T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*.   San Mateo, CA: Morgan Kaufmann, 1992.

[30] R. Huang, J. Wawrzynek, and A. DeHon, "Stochastic, spatial routing for hypergraphs, trees, and meshes," in *Proc. Int. Symp. Field-Programmable Gate-Arrays*, Feb. 2003, pp. 78–87.

[31] M. Pedram, B. S. Nobandegani, and B. T. Preas, "Design and analysis of segmented routing channels for row-based FPGAs," *IEEE Trans. Computer-Aided Design*, vol. 13, pp. 1470–1479, Dec. 1994.

[32] S. Brown, M. Khellah, and Z. Vranesic, "Minimizing FPGA interconnect delays," *IEEE Des. Test Comput.*, vol. 13, pp. 16–23, Jan. 1996.

[33] K. Compton and S. Hauck, "Track placement: Orchestrating routing structures to maximize routability," in *Proc. Int. Conf. Field-Programmable Logic Applications*, Lisbon, Portugal, Sept. 2003, pp. 121–130.

[34] R. Greenberg, "The fat-pyramid and universal parallel computation independent of wire delay," *IEEE Trans. Computers*, vol. 43, pp. 1358–1365, Dec. 1994.

[35] Y.-L. Wu, S. Tsukiyama, and M. Marek-Sadowska, "Graph based analysis of 2-d FPGA routing," *IEEE Trans. Computer-Aided Design*, vol. 15, pp. 33–44, Jan. 1996.

[36] R. I. Greenberg and C. E. Leiserson, "A compact layout for the three-dimensional tree of meshes," *Appl. Math Lett.*, vol. 1, no. 2, pp. 171–176, 1988.

André DeHon (S'92–M'96) received the S.B., S.M., and Ph.D. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, in 1990, 1993, and 1996, respectively.

From 1996 to 1999, he was Co-Head of the BRASS Group, Computer Science Department, University of California, Berkeley. Since 1999, he has been an Assistant Professor of Computer Science at the California Institute of Technology. He is broadly interested in how we physically implement computations from substrates, including VLSI and molecular electronics, up through architecture, CAD, and programming models. He places special emphasis on spatial programmable architectures (e.g., field-programmable gate-arrays) and interconnect design and optimization.

Balancing Interconnect and Computation in a Reconfigurable Computing Array
(or, why you don't really want 100% LUT utilization)

André DeHon

Berkeley Reconfigurable, Architectures, Software, and Systems
Computer Science Division
University of California at Berkeley
Berkeley, CA 94720-1776
`<andre@acm.org>`

Abstract

FPGA users often view the ability of an FPGA to route designs with high LUT (gate) utilization as a feature, leading them to demand high gate utilization from vendors. We present initial evidence from a hierarchical array design showing that high LUT utilization is **not** directly correlated with efficient silicon usage. Rather, since interconnect resources consume most of the area on these devices (often 80-90%), we can achieve more area efficient designs by allowing some LUTs to go unused—allowing us to use the dominant resource, interconnect, more efficiently. This extends the "Sea-of-gates" philosophy, familiar to mask programmable gate arrays, to FPGAs. Also introduced in this work is an algorithm for "depopulating" the gates in a hierarchical network to match the limited wiring resources.

1   Introduction

The ability of an FPGA to support designs with high LUT usage is regularly touted as a feature. However, high routability across a variety of designs comes at a large expense in interconnect costs. Since interconnect is the dominant area component in FPGA designs, simply adding interconnect to achieve high LUT utilization is not always area efficient. In this paper, we ask:

- *Is an FPGA with higher LUT usage more area efficient than one with lower LUT utilization?*

- That is: *Is LUT usability directly correlated with area efficiency?*

Our results to date suggest that this is often not the case—achieving high LUT utilization can often come at the expense of greater area than alternatives with lower LUT utilization. While additional interconnect allows us to use LUTs more heavily, it often causes us to use the interconnect itself less efficiently.

To answer this question, we proceed as follows:
1. Define an interconnect model which allows us to vary the richness of the interconnect.
2. Define a series of area models on top of the interconnect model to estimate design areas.
3. Develop an algorithm for mapping to the limited wiring resources in a particular instance of the interconnect model.
4. Map circuits to a range of points in the interconnect space, and assess their total area and utilization.
5. Examine relationship between LUT utilization and area.

2   Relation to Prior Work

Most traditional FPGA interconnect assessments have been limited to detailed population effects [1] [15]. In particular, they let the absolute amount of interconnect (*i.e.* number of wiring channels or switches) float while assessing how closely a given population scheme allows detailed routing to approach the limit implied by global routing. They also assume that the target is to fully populate the LUTs in a region of the interconnect.

Instead, we take the viewpoint that a given FPGA family will have to have a fixed interconnect scheme and we must assess the goodness of this scheme. To make maximum use of the fixed interconnect, in regions of higher interconnect requirements where the design is more richly connected than the FPGA, we may have to use the physical LUTs in the device sparsely resulting in a depopulated LUT placement. This represents a "Sea-of-Gates" usage philosophy as first explored for FPGAs in University of Washington's Triptych design [4].

For the sake of illustration, consider a design which has a small, but heavily interconnected controller taking up 20% of the LUTs in the design. The rest of the design is a more regular datapath which does not tax interconnect requirements. If we demanded full population, we would look at the interconnect resources necessary to fully pack the controller, and those requirements would set the requirements for the entire array. However, the datapath portion of the chip would not need all of this interconnect and consequently would end up with much unused interconnect. Alternately, we can spread out the controller, ignoring some LUTs in its region of placement, so that the whole FPGA can be built with less interconnect. Now, the controller may take up 30% of the device resources since it cannot use device LUTs 100% efficiently, but the whole device is smaller since it requires less interconnect.

Recently, NTT argued for more wires and less LUTs [17], and HP argued for rich interconnect which will meet or exceed the requirements of logic netlists [2]. Earlier Triptych showed density advantages over traditional alternatives with partially populated designs [4]. The NTT paper examined two points in the space, while HP and University of Washington each justified a single design point. In this paper, we build a model which allows us to explore the tradeoff space more broadly than a few isolated design points. The model is based on a hierarchical network design and captures the dominant switch and wire effects dictating wire area. This generalization, of course, comes at the cost of modeling the design space more abstractly than a particular, detailed FPGA design.

Figure 1: Tree of Meshes

We will be using a hierarchical interconnect scheme as the basis of our area model. Agarwal and Lewis's HFPGA [1] and Lai and Wang's hierarchical interconnect [11] are the most similar interconnect schemes proposed for FPGA interconnect. As noted above neither of these studies made an attempt to fix the wiring resources independent of the benchmark being studied as we are doing here. To permit a broad study of interconnect richness, our interconnect scheme is also defined in a more stylized manner as detailed in the next section.

3   Interconnect Model

The key requirements for our interconnect model is that it:
- represent interconnect richness in a parameterized way
- allows definition of a reasonable area model

To meet these goals, we start with a hierarchical model based on Leighton's Tree of Meshes [13] or Leiserson's Fat Trees [14]. That is, we build a tree like interconnect where the bandwidth grows toward the root of the tree (See Figure 1). We use two parameters to describe a given interconnect scheme:
1. c = the number of base channels at the leaves of the tree
2. p (α) = the growth rate of interconnect toward the root

Note that we realize p by using one of two kinds of stages in the tree of meshes:

- non-compressing (2:1) stages where the root wires are simply equal to the sum of root wires from the two children so there is no net bandwidth reduction

- compressing (1:1) stages where the root wires are the same as each of the root wires from the children, so that only half of the total children wires can be routed upward

By selecting a progression of these stages we can approach any bandwidth growth rate (See Figure 4).

If we use a repeating pattern of stage growths, we approximate a geometric bandwidth growth rate. That is, a subtree of size $2 \cdot n$ has 2^p times as much bandwidth at its root as a subtree of size n, or every tree level has $\alpha = 2^p$ more wires than its immediate children. This is roughly the model implied by Rent's Rule [12] $(IO = c \cdot N^p)$. More precisely, it represents a bifurcator as defined by Bhatt and Leighton [3] (See Figure 2).

Intuitively, p represents the locality in interconnect requirements. If most connections are purely local and only a few connections come in from outside of a local region p will be small. If every gate in a region had a unique signal coming from outside the region, then $p \rightarrow 1.0$. So think of p as describing how rich our interconnect needs to be. If $p = 1$, we are effectively building a crossbar with no restrictions. If $p = 0$, we are building a 1D systolic array or pure binary tree whose IO bandwidth does not grows as the array grows.

Figure 2: (F,α)-bifurcator

4   Area Effects

For our basic area model, we perform a straightforward layout of the elements shown in Figure 4. That is, we have:
- Logic Block of size A_{pe}
- Switches of size A_{sw}
- Wires of pitch WP

Each subtree is built hierarchically by composing the two children subtrees and the new root channel. Channel widths are determined by either the area required to hold the switches or the width implied by the wire channels, depending on which is greater. We assume a dedicated layer for each of horizontal and vertical interconnect. The result is the "cartoon" VLSI layout as shown in Figure 3.

Figure 3: "Cartoon" Layout of Hierarchical Interconnect

Note that the number of base channels (c) is 3 in all these examples.

Figure 4: Programming Growth for Tree of Meshes

Figure 5: c choose k LUT Input Population ($c = 5$, $k = 3$)

Typical values for an SRAM programmable device:[1]

- $A_{pe} = 40K\lambda^2$ — this would hold 16 memory bits for a 4-LUT ($16 \times 1.2K\lambda^2$/SRAM-bit$\approx$20Kλ^2) plus a the LUT multiplexor and optional output flip-flop (13Kλ^2 in [5], 15Kλ^2 in [8]).
- $A_{sw} = 2.5K\lambda^2$ for a pass transistor switch (including its dedicated SRAM programming bit) — to model mask or antifuse programmable devices, we would use a much smaller size for this parameter.
- $WP = 8\lambda$ for the metal 2 or metal 3 wire trace and spacing

We assume the channels are populated with c choose k input selectors [7] on the input and have a fully populated output connection (See Figure 5). Switch boxes are either fully populated or linearly populated (see Figure 6) with switches.

Figure 7 shows cartoon layouts for 3 different choices of p, highlighting the area implied by each choice. Two things we can observe immediately from this simple model comparison:

- For reasonable parameters, interconnect requirements dominate logic block area; e.g. at $c = 6$, $p = 0.67$, a design with 1024 LUTS has only 5% of its area in LUTs (estimated area per LUT including interconnect is $\approx$750Kλ^2) — while this is a simple area model, the area and ratio are not atypical of real FPGA devices; they are also consistent with prior studies (e.g. 6% for 600 4-LUT design in [5]).

Figure 6:   Linear   Switchbox   Population   for   Hierarchical Interconnect

| $p = 0.50$ | $p = 0.67$ | $p = 0.75$ |
| Rel. Area = 0.25 | Rel. Area = 0.37 | Rel. Area = 1.00 |

Figure 7: Effects of p on Area at 1K LUTs

---

[1] λ = half the minimum feature size for a VLSI process. Assuming linear scaling of all features, λ^2 area should be the same across processes.

3

Figure 8: "Pulling" design up tree to match fixed wire schedule

- Interconnect parameter richness has a large effect on total area.

    To further build intuition, let's assume for a moment that a design can be perfectly characterized by a growth exponent p. If the growth exponent for the interconnect matches the growth of the design ($p_{interconnect} = p_{design}$), then the network will require minimum area. What happens if these two are not perfectly matched? There are two cases:

    - $p_{interconnect} > p_{design}$ — we have more interconnect than necessarily. The design can use all the LUTs in the network, but the network has more wires. As a result, the area per LUT is larger than the matched case—that is, mapping the design on the richer interconnect takes more area than the matched design case.

    - $p_{interconnect} < p_{design}$ — we have less interconnect than necessary. We cannot pack the design into a minimum number of LUTs in order to fit the design. Instead we must pull the design up the tree, effectively depopulating the logic blocks, until the tree provides adequate connectivity for the design (See Figure 8). As a result, we have leaves in the tree which are not fully utilized. As we will see, this also takes more area than the matched design case.

Figure 10 shows the area overhead required to map various designs onto interconnects with various growth factors. As we expect, it shows that the matched interconnect point is the minimum point with no overhead. As we go to greater or lesser interconnect offered by the network, the area overhead grows, often dramatically.

5   Design Requirements

In practice, of course, c and p values are a rough characterization of the interconnect requirements for a real design. With multiple subgraphs of a given size (subtrees at the same height in the tree) we get more than one I/O to subgraph relationship. Further, the growth is seldom perfectly exponential. Finally, even asking if a graph has an (F, α)-bifurcator is an NP-hard problem. So, the bifurcations

Figure 9: I/O versus Partition size graph for `i10`

we construct are heuristic approximations biased by the tools we employ.

    Figure 9 shows the I/O versus subgraph relationship for the one of the IWLS93 benchmark (`i10`).

- Mapped for area with SIS [16] and Flowmap [6]
- Recursively bisected using a single Eigenvalue spectral partitioner

The recursive bisection approximates the natural bandwidth versus subtree sizes which exist in the design. We see the I/O to subgraph relationship is not 1:1. We also see that the max and average contours can be matched well to a geometric growth rate (e.g. Rent's Rule—average $c = 5, p = 0.7$; max $c = 7, p = 0.7$).

    The left of Figure 11 shows the I/O versus subgraph relationship for all the IWLS93 benchmarks area mapped to 2000 or fewer LUTs using SIS, Flowmap, and spectral partitioning as above. On the right it shows the distribution of Rent parameter estimates for these benchmarks. Here we see that while we may be able to pick "typical" c and p values, there is a non-trivial spread in interconnect requirements across this set of designs.

6   Mapping to Fixed Wire Schedule

We have now seen that we can define a parameterized interconnect model with a fixed wire schedule. Designs have their own requirements which do not necessarily match the fixed wire schedule available from a device's interconnect. When the device offers more interconnect than a design needs, mapping is easy, we simply place the design on the interconnect and waste some wires. However, if the design has more interconnect needs than the device provides, how do we map the design to the device?

    As suggested in Figure 8, we can start with the recursively bi-partitioned design and simply pull the whole design up the tree until all the interconnect wires meet or exceed the design requirements. However, keeping the groupings originally implied by the recursive bisection is overly strict. In particular, re-associating the subgraphs based on interconnect availability can achieve tighter packings (See Figure 13). That is, we do not really want a bisection of the LUTs, but a bisection of the total capacity including both interconnect and LUTs. Intuitively, the size of a subgraph is determined by the greater of its LUT requirement and its interconnect requirement relative to the fixed wire schedule of the device.

    To attack the problem of regrouping subtrees to fit into the fixed wire schedule, we introduce a dynamic programming algorithm which determines where to split a given subgraph based on the available wire schedule. That is, we start with a linear ordering of LUTs. Then, we ask where we should cut this linear order-

Figure 10: Theory: effects of mismatched between interconnect network p and design requirements

Figure 11: I/O versus Partition size graph for Benchmark Set

```
// size[start,finish] represents the smallest subtree which will
//        contain the set of LUTs between position start and finish
// uniqueio(o,i,j) returns the number of unique nets which appear both in the subrange i→j,
//        and outside of that range
o = order all LUTs
for i=0 to o.length
        size[i,i] ← size(1,unique(o,i,i)) // base case ≡ single LUT subtrees
for len=2 to o.length
        for start=0 to o.length-len // process all subranges of specified length
                minsize=MAX
                end=start+len-1
                isize=uniqueio(o,start,end)
                for mid=start+1 to end // search for best split point
                        msize=1+max(size[start,mid],size[mid+1,end])
                        size=max(msize,iolevel(isize))
                        minsize=min(size,minsize)
                size[start,end]←minsize
// final result is size[0,o.length-1]
```

Figure 12: Dynamic Programming Algorithm to Map to Fixed Wire Schedule

5

Figure 13: Re-associating Subgraph clusters to match Fixed Wire Schedule

ing of LUTs into two subtrees in order to minimize the total area required—typically, minimizing the heights of the two subtrees. Each of the subtrees are then split in a similar manner. To make the decision of where to cut a subtree, we examine all cut points. As long as we have a single linear ordering for LUTs, this is very similar to the optimal parenthesis matching problem. In a similar manner, we can solve this problem with a dynamic programming algorithm.

The dynamic programming algorithm (Figure 12) finds the optimal subtree decomposition **given** the initial LUT ordering. The trick here, and the source of non-optimality, is picking the order of the LUTs. For this we use the 1D spectral ordering based on the second smallest Eigenvalue which Hall shows is the optimal linear arrangement to minimize squared wire lengths [10].

Figure 14 shows why the single linear ordering is non-optimal. Here we see a LUT B placed to minimize its distances to A, C, and D. The order is such as to keep B, C, and D together for cut 3. However, if we take cut 4, then it would be more appropriate to place B next to A since we have already paid for the wires to C and D to exit the left subgroup. However, as long as we are using a single linear ordering, we do not get to make this movement after each cut is made. In general to take proper account of the existing cut, we should reorder each of the subgraphs ignoring ordering constraints originally imposed by the wires which have already been cut.

To avoid this effect, we would have to reorder each subtree after each cut is made. In addition to increasing the complexity of each cut, this would destroy the structure we exploited to apply dynamic programming—that is, the sub-problems would no longer be identical. Of course, since the spectral partitioning does not even give an optimal cut point for the bisection problem, the ordering effect alone is not the only element of non-optimality here.

There is certainly room for algorithmic improvement here. The

Figure 14: Example showing the limitations of a Single Linear Ordering

results are, nonetheless, good enough to give us interesting depopulations as we will see in the next section.

7   Results from Mapping

Putting it all together:
- Start with the area targeted SIS and Flowmap 4-LUT networks for the IWLS93 benchmarks under 2000 4-LUTs.
- Order using the second smallest Eigenvalue.
- Map to fixed schedule with the dynamic programming algorithm; The results are shown in terms of relative number of LUTs in the top left of Figure 15.
- Apply an area cost model such as shown in top right of Figure 15.
- Result is the relative area map shown at the bottom of Figure 15.

Figure 15 shows that there is a minimum area point across the benchmark set. For our linear switch population model, this occurs at $c = 6, p = 0.6$. As our theory predicts, too much interconnect and too little interconnect both account for area overheads over the minimum. Notice that the only points where the entire benchmark achieves full utilization are $c = 10, p \geq 0.75$ and $p = 0.8, c \geq 7$, all points which are above the minimum area point.

Table 1 examines the effects of picking a particular point in the c-p-design space. For each design in the benchmark set, we can compute the c,p-point which has minimum area. We can then look at the overhead area required between the "best" c, p, picked for the individual design, versus the best c, p for the entire benchmark under certain criteria. For the linear switch population case, we see that average overhead between the benchmark minimum and each benchmark's best area is only 23% and that corresponds to an average LUT utilization of 87%. Similarly, we see that picking the smallest point where we get 100% device utilization results in almost 200% area overhead. We see different absolute numbers, but similar trends with other area models.

Given the range of partition ratios and cut sizes we saw in Figure 11, it is not that surprising that the full utilization point is excessive for many designs and leads to many area inefficient implementations. Figure 16 shows a slice in p-space for the single design i10 whose I/O versus subgraph size curve we showed in Figure 9. Notice that even for this single design, the minimum area point does not correspond to full utilization. In fact, the minimum area point is actually only 50% of the area of the full utilization point. So, even for a single design allowed to pick the network parameters c, p which minimizes device area, full LUT utilization does not always correspond to better area utilization. We see here that the effects of varying wire requirements, which we described in Section 2, do actually occur in designs.

In the previous section, we noted that the fixed wire schedule mapping algorithm in use is not optimal. It is worth considering how a "better" algorithm would affect the results presented here. A "better" algorithm could achieve better LUT utilization for the points where depopulation occurs. For the points on the graph where no depopulation occurs, a better algorithm could offer no improvement. As a result, we expect a better algorithm to magnify these effects—making the depopulated designs tighter and take less

6

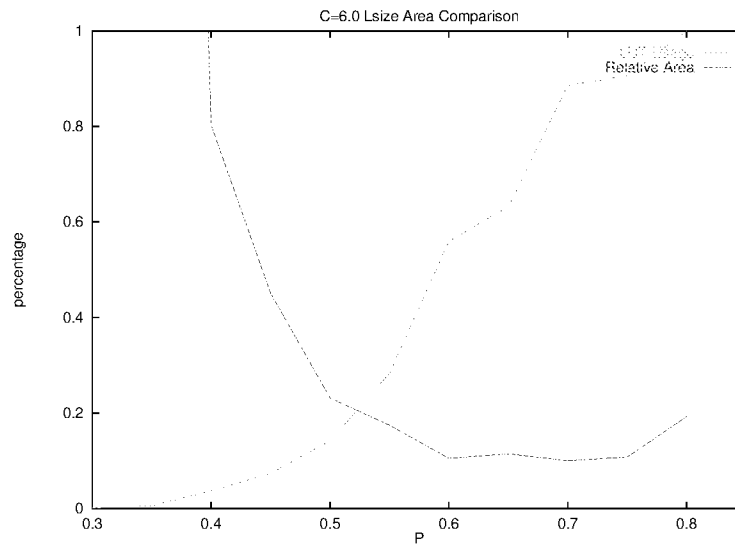Figure 15: Area Utilization Results Mapping Benchmark to Fixed Wire Schedules

Figure 16: p-space slice for i10 showing that area minimization is not directly correlated with high LUT usage

7

Wire Dominated $WP = 8\lambda$, $A_{sw} = 64\lambda^2$

Minimization Objective	params C	P	Sigma relative area	Max relative area	LUT Utilization
relative area	6	0.6	1.30	2.99	0.87
max relative area	6	0.65	1.40	1.91	0.93
area with full utilization	10	0.75	3.23	6.94	1.00

Linear $WP = 8\lambda$, $A_{sw} = 2500\lambda^2$

Minimization Objective	params C	P	Sigma relative area	Max relative area	LUT Utilization
relative area	6	0.6	1.23	2.84	0.87
max relative area	6	0.65	1.24	2.38	0.89
area with full utilization	10	0.75	2.98	4.87	1.00

Switch Dominated (Quadratic)
$WP = 8\lambda$, $A_{sw} = 2500\lambda^2$

Minimization Objective	params C	P	Sigma relative area	Max relative area	LUT Utilization
relative area	6	0.6	1.32	3.50	0.87
max relative area	4	0.65	1.47	2.31	0.49
area with full utilization	10	0.75	4.25	11.5	1.00

Table 1: Compare Effects of Various Network Selection Points

area, while the full utilization designs stay at roughly the same point.

8 Limitations and Future Study

We have only scratched the surface here. As with any CAD effort where we are solving NP-hard problems with heuristic solutions there is a significant tool bias to the results. Flowmap was not attempting to minimize interconnect requirements, and there is a good argument that LUT covering and fixed-wire schedule partitioning should be considered together to get the best results. At the very least, it would be worthwhile to try different LUT mapping strategies to assess how much these results are effected by LUT covering.

The area model used assumes a purely hierarchical, 2-ary interconnect. Two things one would like to explore are (1) the effects of different arity (flattening the tree) and (2) the introduction of shortcut connections (e.g. Fat Pyramid [9]). The shortcut connections will tend to reduce the need for bandwidth in the root channel and may shift the balance in interconnect costs. Further, shortcuts appear essential for delay-mapped designs, which we have also not studied here.

We suspect the hierarchical model captures the high-level requirements of any network, but it will be interesting to study these effects more specifically for mesh-based architectures. The key algorithmic enabler needed for both shortcuts and mesh-based architectures is to identify good heuristics for spreading in two dimensions rather than the one-dimensional approach we exploited here.

An important assumption we have made here is that interconnect growth is geometric (power law). The c, p estimates shown in Figures 9 and 11 support the fact that a geometric growth relationship seems fairly reasonable. Nonetheless, we have not directly explored wire-schedules which deviate from strict geometric growth, and there may be better schedules to be found outside of the strict geometric growth space explored here.

We concentrated on global wiring requirements here and have not focussed on detailed switch population. The robustness of the general trends across different area and population models shown in Table 1 suggests that the major effects identified here are independent of the switch population details. While this does show us the relative merits of a given interconnect richness within a particular population model, we cannot, however, make any conclusions about the relative merits of different population schemes without carefully accounting for detailed population effects in both the area model and routability assessment.

9 Conclusions

We see that wires and interconnect are the dominant area components of FPGA devices. We also see that the amount of interconnect needed per LUT varies both among designs and within a single design. Given that this is the case, we cannot use all of our LUTs and all of our interconnect to their full potential all of the time—we must underutilize one resource in order to fully utilize the other. If we focus on LUT utilization, we waste significant interconnect— our dominant area resource. This suggests, instead, it may be more worthwhile for us to focus on interconnect utilization even if it means letting some LUTs go unused. Answering our opening question, we see that higher LUT usage does not imply lower area and that LUT usability is not always directly correlated with area efficiency.

8

benchmark mapping to LUTs and developed the initial BLIF parser used for these experiments.

Feedback from Randy Huang, Nicholas Weaver, John Wawrzynek, and Eylon Caspi helped cleanup early drafts of this paper.

References

[1] Aditya A. Agarwal and David Lewis. Routing Architectures for Hierarchical Field Programmable Gate Arrays. In *Proceedings 1994 IEEE International Conference on Computer Design*, pages 475–478. IEEE, October 1994.

[2] Rick Amerson, Richard Carter, W. Bruce Culbertson, Phil Kuekes, and Greg Snider. Plasma: An FPGA for Million Gate Systems. In *Proceedings of the International Symposium on Field Programmable Gate Arrays*, pages 10–16, February 1996.

[3] Sandeep Bhatt and Frank Thomson Leighton. A Framework for Solving VLSI Graph Layout Problems. *Journal of Computer System Sciences*, 28:300–343, 1984.

[4] Gaetano Borriello, Carl Ebeling, Scott Hauck, and Steven Burns. The Triptych FPGA Architecture. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 3(4):491–501, December 1995.

[5] Stephen D. Brown, Robert J. Francis, Jonathan Rose, and Zvonko G. Vranesic. *Field-Programmable Gate Arrays*. Kluwer Academic Publishers, 101 Philip Drive, Assinippi Park, Norwell, Massachusetts, 02061 USA, 1992.

[6] Jason Cong and Yuzheng Ding. FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs. *IEEE Transactions on Computer-Aided Design*, 13(1):1–12, January 1994.

[7] André DeHon. Entropy, Counting, and Programmable Interconnect. In *Proceedings of the 1996 International Symposium on Field Programmable Gate Arrays*. ACM/SIGDA, February 1996. Extended version available as Transit Note #128 <http://www.ai.mit.edu/projects/transit/transit-notes/tn128.ps.Z>.

[8] André DeHon. Reconfigurable Architectures for General-Purpose Computing. AI Technical Report 1586, MIT Artificial Intelligence Laboratory, 545 Technology Sq., Cambridge, MA 02139, October 1996. Anonymous FTP publications.ai.mit.edu: ai-publications/1996/AITR-1586.ps.Z.

[9] Ronald Greenberg. The Fat-Pyramid and Universal Parallel Computation Independent of Wire Delay. *IEEE Transactions on Computers*, 43(12):1358–1365, December 1994.

[10] Kenneth M. Hall. An r-dimensional Quadratic Placement Algorithm. *Managment Science*, 17(3):219–229, November 1970.

[11] Yen-Tai Lai and Ping-Tsung Wang. Hierarchical Interconnect Structures for Field Programmable Gate Arrays. *IEEE Transactions on VLSI Systems*, 5(2):186–196, June 1997.

[12] B. S. Landman and R. L. Russo. On Pin Versus Block Relationship for Partitions of Logic Circuits. *IEEE Transactions on Computers*, 20:1469–1479, 1971.

[13] Frank Thomson Leighton. New lower bound techniques for VLSI. In *Twethy-Second Annual Symposium on the Foundations of Computer Science*. IEEE, 1981.

[14] Charles E. Leiserson. Fat-Trees: Universal Networks for Hardware Efficient Supercomputing. *IEEE Transactions on Computers*, C-34(10):892–901, October 1985.

[15] Jonathan Rose and Stephen Brown. Flexibility of Interconnection Structures for Field-Programmable Gate Arrays. *IEEE Journal of Solid-State Circuits*, 26(3):277–282, March 1991.

[16] Ellen M. Sentovich, Kanwar Jit Singh, Luciano Lavagno, Cho Moon, Rajeev Murgai, Alexander Saldanha, Hamid Savoj, Paul R. Stephan, Robert K. Brayton, and Alberto Sangiovanni-Vincentelli. SIS: A System for Sequential Circuit Synthesis. UCB/ERL M92/41, University of California, Berkeley, May 1992.

[17] Atsushi Takahara, Toshiaki Miyazaki, Takahiro Murooka, Masaru Katayama, Kazuhiro Hayashi, Akihiro Tsutsui, Takaki Ichimori, and Ken nosuke Fukami. More Wires and Fewer LUTs: A Design Methodology for FPGAs. In *Proceedings of the 1998 International Symposium on Field-Programmable Gate Arrays*, pages 12–19, 1998.

A Mapped Benchmarks Statistics used for Experiment

Design	Mapped LUTs	Max c	Max p	Avg c	Avg p	Design	Mapped LUTs	Max c	Max p	Avg c	Avg p	Design	Mapped LUTs	Max c	Max p	Avg c	Avg p
5xp1	83	7	0.35	6	0.42	ex2	90	6	0.5	5	0.56	s208	27	6	0.34	5	0.29
9sym	177	6	0.57	6	0.63	ex3	33	6	0.45	5	0.52	s27	6	5	0.20	0	0
9symml	80	6	0.45	5	0.47	ex4	41	7	0.36	6	0.42	s298	40	6	0.35	5	0.41
C1355	74	7	0.64	5	0.67	ex5	27	6	0.36	5	0.44	s344	41	6	0.34	4	0.44
C17	2	0	0	0	0	ex6	71	6	0.51	6	0.4	s349	41	6	0.34	4	0.44
C1908	136	7	0.56	5	0.56	ex7	36	6	0.51	5	0.54	s382	53	6	0.44	5	0.48
C2670	218	6	0.7	4	0.75	example2	139	6	0.69	4	0.74	s386	74	7	0.32	6	0.36
C3540	382	7	0.54	6	0.56	f51m	98	7	0.53	5	0.58	s400	53	6	0.38	5	0.42
C432	79	7	0.53	6	0.57	frg1	282	6	0.62	6	0.51	s420	62	7	0.31	5	0.39
C499	74	7	0.63	5	0.67	frg2	744	8	0.61	5	0.65	s444	53	6	0.37	5	0.39
C5315	590	7	0.66	5	0.69	i1	19	5	0.66	4	0.74	s510	105	7	0.45	6	0.47
C6288	522	8	0.44	6	0.47	i10	906	7	0.68	5	0.71	s526	84	7	0.39	5	0.41
C7552	723	7	0.63	5	0.68	i2	77	6	0.82	5	0.86	s526n	90	7	0.39	5	0.42
C880	116	6	0.65	5	0.65	i3	46	5	0.88	5	0.89	s5378	522	7	0.63	5	0.66
a	1	0	0	0	0	i4	97	5	0.77	4	0.82	s641	79	5	0.6	4	0.68
alu2	279	8	0.49	6	0.54	i5	153	6	0.67	4	0.75	s713	80	6	0.61	5	0.67
apex1	799	8	0.62	7	0.6	i6	144	5	0.76	4	0.78	s8	1	0	0	0	0
apex3	900	9	0.54	6	0.58	i7	215	6	0.71	4	0.77	s820	166	7	0.48	6	0.52
apex6	258	6	0.71	5	0.68	i9	347	7	0.63	5	0.65	s832	169	7	0.48	6	0.52
apex7	108	6	0.61	4	0.64	keyb	209	8	0.43	6	0.48	s838	124	7	0.44	5	0.5
b1	4	4	0.40	3	0.67	kirkman	133	8	0.35	6	0.37	s9234	439	8	0.56	5	0.63
b12	377	7	0.55	7	0.46	lal	67	7	0.56	5	0.6	s953	182	7	0.5	6	0.54
b9	56	6	0.62	4	0.69	ldd	50	7	0.46	5	0.5	sand	406	7	0.57	5	0.61
bbara	34	6	0.43	5	0.44	lion	3	0	0	0	0	sao2	121	7	0.47	6	0.5
bbsse	70	8	0.32	6	0.36	lion9	5	0	0	5	0.1	sbc	332	7	0.57	5	0.6
bbtas	9	5	0.25	4	0.3	majority	4	5	0.13	5	0.17	scf	663	9	0.53	6	0.57
beecount	21	6	0.43	5	0.49	mark1	52	7	0.41	4	0.66	sct	69	7	0.48	5	0.52
c8	87	7	0.54	5	0.58	mc	9	5	0.21	4	0.33	shiftreg	2	0	0	0	0
cc	33	5	0.65	4	0.73	misex1	24	6	0.37	5	0.41	sqrt8	45	7	0.46	6	0.37
cht	68	5	0.71	4	0.69	misex2	54	6	0.57	5	0.59	sqrt8ml	40	6	0.58	5	0.47
clip	243	7	0.53	6	0.44	mm30a	327	6	0.57	6	0.47	squar5	50	8	0.27	6	0.34
clmb	369	2	1	2	1	mm4a	118	9	0.28	7	0.31	sse	70	8	0.32	6	0.36
cm138a	9	5	0.44	5	0.48	mm9a	96	6	0.48	6	0.38	styr	341	7	0.59	7	0.47
cm150a	15	5	0.67	4	0.75	mm9b	120	6	0.51	5	0.53	t481	735	8	0.55	6	0.6
cm151a	8	5	0.53	4	0.63	modulo12	1	0	0	0	0	table3	513	9	0.52	7	0.55
cm152a	11	5	0.41	5	0.43	mult16a	32	5	0.36	4	0.36	table5	522	7	0.66	7	0.53
cm162a	14	6	0.49	5	0.56	mult16b	31	5	0.25	4	0.16	tav	12	5	0.23	4	0.50
cm163a	12	5	0.60	5	0.65	mult32a	64	5	0.39	4	0.38	tbk	616	9	0.5	6	0.54
cm42a	10	5	0.45	5	0.45	mult32b	62	6	0.41	5	0.43	tcon	16	4	0.78	3	0.88
cm82a	4	4	0.5	4	0.5	mux	45	6	0.51	6	0.47	term1	246	8	0.48	6	0.52
cm85a	12	5	0.42	5	0.44	my_adder	32	4	0.61	4	0.61	train11	19	6	0.48	4	0.76
cmb	19	6	0.49	5	0.53	o64	46	5	0.84	4	0.88	train4	3	0	0	0	0
comp	45	6	0.52	5	0.57	opus	50	6	0.51	5	0.57	ttt2	198	9	0.42	6	0.45
con1	6	5	0.31	5	0.33	pair	567	8	0.6	5	0.65	unreg	32	5	0.67	5	0.7
count	37	6	0.58	4	0.64	parity	5	5	0.73	5	0.75	vda	517	8	0.55	6	0.58
cps	821	9	0.62	6	0.67	pcle	23	5	0.5	4	0.59	vg2	277	9	0.42	6	0.47
cse	134	6	0.57	5	0.58	pcler8	33	5	0.6	4	0.68	x1	761	9	0.48	6	0.57
cu	24	6	0.51	5	0.56	planet	410	9	0.45	6	0.5	x2	23	6	0.39	5	0.53
daio	3	0	0	0	0	planet1	410	9	0.45	6	0.5	x3	441	7	0.61	5	0.6
dalu	502	8	0.55	6	0.59	pm1	28	6	0.54	4	0.61	x4	294	7	0.61	5	0.63
decod	18	5	0.49	4	0.52	rd53	36	6	0.38	5	0.4	xor5	21	6	0.38	5	0.46
dk14	51	7	0.35	7	0.24	rd73	190	6	0.56	5	0.57	z4ml	80	7	0.39	5	0.41
dk15	31	6	0.39	6	0.22	rd84	405	7	0.63	5	0.67	alu4	1756	8	0.58	6	0.63
dk16	171	8	0.42	6	0.47	rot	467	7	0.65	5	0.71	apex4	1284	7	0.69	5	0.72
dk17	30	6	0.38	5	0.41	s1	317	8	0.52	6	0.56	apex5	1241	9	0.63	6	0.66
dk27	5	5	0.15	4	0.35	s1196	226	7	0.53	6	0.45	cordic	1381	9	0.62	8	0.52
dk512	25	6	0.17	5	0.22	s1238	253	7	0.52	7	0.46	dsip	1175	7	0.65	6	0.64
donfile	1	0	0	0	0	s1423	162	6	0.51	5	0.40	ex5p	1348	9	0.61	6	0.64
duke2	274	8	0.48	6	0.53	s1488	289	7	0.55	4	0.76	i8	1242	8	0.57	6	0.60
e64	386	7	0.6	5	0.64	s1494	295	7	0.59	6	0.51	k2	1138	8	0.69	6	0.58
ex1	164	6	0.6	5	0.64	s1a	6	1	1	1	1	seq	2004	10	0.53	7	0.58

B Lsize and Level

When mapping to a hierarchical array, or any array for that matter, one problem to address is how we count area used. Do we charge the design for the smallest tree hierarchy used? If so, we only get a logarithmic estimation of size. Designs which are slightly larger than a tree stage are charged the full cost of the next tree level. This could skew measures as ±1 LUT at a power-of-two boundary has a big difference in metric, but elsewhere near factor-of-two differences hardly matter. For the data shown here, we have counted size in terms of the span of LUTs used (*lsize* — See adjacent diagram). That is, if we number the tree LUTs in a linear order; we pack starting at LUT 0 and use the position of the highest placed LUT to account for the capacity used. The LUTs above the last used subtree are all free. Intuitively, if we consume all of a subtree of size 128 and one more subtree of size 64, we still have a subtree of size 64 available for additional logic, so we charge the design to be only of lsize 192. In practice, when we use level as a metric instead of lsize, we see similar trends to those reported here but a larger benchmark-wide mismatch penalty, especially when requiring full population, due to the logarithmic granularity effects.

Level = 3

Lsize = 5

10

Compact, Multilayer Layout for Butterfly Fat-Tree

André DeHon
California Institute of Technology
Department of Computer Science, 256-80
Pasadena, CA 91125
andre@acm.org

ABSTRACT

Modern VLSI processing supports a two-dimensional surface for active devices along with multiple stacked layers of interconnect. With the advent of planarization, the number of layers can be large (6 or 7 in modern designs) and more layers are feasible if the cost is justified. Using a multilayer-wiring VLSI area model, we show how a butterfly fat-tree (or fat-pyramid) with N processors can be laid out in $\Theta(N)$ active device area using $\Theta(\log(N))$ wiring layers. This result may have practical value in laying out efficient, single-chip multiprocessors and FPGAs. It may also provide a theoretical basis for the rate of layer scaling empirically seen in VLSI designs.

Categories and Subject Descriptors

B.7.1 [**Integrated Circuits**]: Types and Design Styles—*VLSI*; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Network Topology*; C.1.4 [**Processor Architectures**]: Parallel Architectures

General Terms

VLSI Layout Theory, Fat-Tree, Fat-Pyramid, Scaling, Universal Network, Multiprocessor, FPGA

1. INTRODUCTION

Traditional VLSI area models (*e.g.* [10]) assume two, or a small fixed number of, wiring layers, which was very appropriate for early VLSI process capabilities. With this model it was possible to identify many interesting cases where wiring limitations determined the size required by chips. Modern VLSI processes, perhaps in response to the empirical recognition of these wiring limitations, now offer many layers of wiring. It is, consequently, interesting to review VLSI wiring restrictions exploiting the new multilayer wiring model.

This paper looks specifically at fat-tree style wiring. The fat-tree was constructed specifically to be efficient for VLSI layouts, and the canonical 2D fat-tree is an example of a structure whose area is wiring limited. Further, the fat-tree can be used as a universal interconnect or wiring substrate. We show that the wiring struc-

ture in the fat-tree is sufficiently regular to permit a layout in $\Theta(N)$ area (the area dictated by the nodes and switches) using $O(\log(N))$ wiring layers. This should be compared to an area of $O(N \log^2(N))$ using a conventional, 2D, bounded wiring layers, layout for a fat-tree.

The paper starts with an abstract of modern, multilayer VLSI layout (Section 2) and a review of the butterfly fat-tree and fat-pyramid (Section 3). In Section 4, we demonstrate the major result that a butterfly fat-tree can be placed and routed efficiently using multiple wiring layers. In Section 5, we look at how this result may relate to VLSI wiring growth. We identify a few, interesting, open questions which this raises in Section 6.

2. MODERN, MULTILAYER VLSI LAYOUT

Contemporary VLSI processes easily offer 6 layers of metalization for wiring. With the advent of Chemical Mechanical Planarization (CMP) [9], it is feasible for process technology to continue stacking additional metal layers as long as the cost of the extra mask steps and processing are justified by the area benefits. This produces an interesting twist on the traditional VLSI models. With current technology, active devices (transistors, gates, buffers) are still largely limited to two-dimensional layout on the silicon substrate. However, wire layers can feasibly be stacked on top of each other creating a three-dimensional structure for interconnect and wiring.

This gives us a model where:

1. Devices which actually compute upon, store, or switch data must be laid out in two dimensions.
2. Wires which interconnect these devices have finite width and spacing.
3. Wires on any two wiring layers can be interconnect with vias and will take up finite space on all intervening layers.

If the active devices for some structure take up total area A, then it is interesting to ask if the active devices can be laid out compactly to fit in $O(A)$ two-dimensional surface area and be supported by the multilayer wiring. Further, we should ask how many wiring layers are required to support this compact active area layout.

3. BUTTERFLY FAT-PYRAMID AND FAT-TREE

The particular structure we are interested in here is Leiserson's Fat-Tree [7] and, by extension, Greenberg's Fat-Pyramid [3]. Results from Leiserson and Greenberg show that an N-node fat-tree (and fat-pyramid) can be laid out in $O(N \log^2(N))$ area [4] using the

fold-and-squash technique of Leighton and Bhatt [1]. Figure 1 shows the Butterfly version of Leiserson's Fat-Tree [5] (the fat-pyramid is similar, adding a constant number of additional wires between physical adjacent switch nodes at the same tree level) along with its compact, fold-and-squash layout.

For this layout it is important to note that each 4-ary tree layer, corresponding to multiplying the number of nodes in the tree by 4, adds:

- a constant number of wire tracks (6 as shown) per "cubie"[1]
- a constant number of switches (1) to *some* cubies

Hence we get the logarithmic growth in the side width of each cubie due to wiring. Since wire width alone in the 2D VLSI model dictates a side growth of $O(\log^2(N))$, it does not (theoretically) matter than some cubies have a switch count which is growing as $O(\log(N))$. The overall result is that the area of the N-node fat-tree grows as $O(N \log^2(N))$.

Active Devices

It is, however, worthwhile to note that the number of active devices in the butterfly fat-tree (and fat-pyramid) converges to a constant independent of the number of tree levels. It should be trivially clear that the number of endpoints nodes is N. It is also true that the number of switching nodes is $\Theta(N)$. For example, if we assume a 4-ary tree with switches with 4 down links and 2 up links, as shown, then the total number of switches is at most $\frac{N}{2}$. To see this, note that each group of 4 leaf nodes needs one switch at the lowest level (labeled 4 in Figure 1). At the next level, we need half as many switches (every 4 switches on the lower level needs 2 switches at the next level). This relationship continues with each succeeding level requiring half as many switches as the level before. Consequently, the number of switches needed per endpoint can be calculated as a classical geometric series:

$$N_{switch} = \frac{N}{4} + \frac{1}{2}\left(\frac{N}{4}\right) + \frac{1}{4}\left(\frac{N}{4}\right) + \frac{1}{8}\left(\frac{N}{4}\right) + \cdots \leq \frac{N}{2}$$

(1)

Since switches and endpoints make up the entire set of active devices, this demonstrates the active device area for a butterfly fat-tree is $\Theta(N)$.

4. LAYOUT

Having established that the active device requirement for a butterfly fat-tree is $\Theta(N)$, the question remains as to whether or not the device can be conveniently laid out in this area and the wiring can all be performed in a reasonable number of wiring layers. We also note from our observations in the previous section that the number of wiring channels per cubie is $O(\log(N))$. Since it is necessary to build a cubie in space $O(1)$ if we are to layout the entire tree in active, two-dimensional area $O(N)$, then that sets a trivial lower bound of $\Omega(\log(N))$ on the number of wire layers required to wire the fat-tree. In fact, it is possible to organize the fat-tree so that it can be laid out in $O(N)$ active area and $O(\log(N))$ wiring layers. Two show this is possible, we demonstrate two things:

1. The switches can be arranged to be placed into cubies so there are at most a constant number (2) of switches in any cubie.

---

[1]Cubies shown here contain 4 processing nodes, but are otherwise similar to the cubies shown in [4].

2. When we account for **both** the wiring per layer and the through vias required between layers, we do not saturate any of the wiring layers.

4.1   Switch Placement

Figure 2 shows the rearrangement of the basic fat-tree and its fold-and-squash layout. The rearranged fat-tree is topologically equivalent to the original fat-tree (Figure 1). However, when this rearrangement is folded up, at most 2 switches end up in the cubie along with 4 processing nodes (Figure 8, provided at the end of the paper, builds the tree one level higher to better show this effect).

In the original fat-tree arrangement, all the switches lie along the same diagonal. In the new arrangement, the diagonals are complementary so that, when folded together, the next level diagonal is always left open. Figure 3 shows the actual folding sequence to display the basic invariant maintained by this arrangement. Each final cubie will contain the 4 leaf processing element, the switch associated with those four processing elements, and, at most, one additional switch. For clarity, the processors and first switch (labeled 4) are not shown in Figure 3 once folding begins.

Notice, at each stage, that, after folding, the lower level(s) manages to leave **both** main diagonals free. One main diagonal is then consumed by the new switches added at the level onto which the lower levels are being folded. This, in turn, leaves one diagonal free in the folded box. As a consequence when this new level is now folded with its peers to create the next tree level, it will also create a structure with both main diagonals free so that the next level of switches can be added and the folding can continue in this manner *ad infinitum*.

4.2   Wires

The basic strategy for wiring is to give each tree layer its own pair of wire layers—one for horizontal wiring and one for vertical wiring. In all likelihood the constants will work out such that more than one tree layer can share the same wiring layer, but for the sake of clear exposition, we will use this generous assumption. As shown, the wiring per tree layer is, at most, 6 wires wide,[2] so we immediately see we have a constant number of wires running through each cubie side on each of the $2 \cdot \log(N)$ wiring layers.

Now, we must also show that we can accommodate all of the through vias in constant area. Since there are at most two switches per cubie, there must be at most $6 \times 2 = 12$ through interconnect vias from the substrate to some routing layer in each cubie.[3] We can allocate a via track for each wire channel in each wiring layer in order to make the connection down to the substrate. Further, the vias in this channel will need to be spaced one wire channel apart to avoid blocking the wires running the orthogonal direction (see Figure 4b). As shown in Figure 4a-b, each of the channels stacked on top of each other on different routing layer can route out to the single via channel and down to the substrate where it needs to connect without creating interference with the other channels in its stack. Note also that we assume the horizontal and vertical layers for a given tree layer are adjacent so that via connections between them can be made without disturbing wiring on any other wiring layers. This composite construction shows that we can wire each cubie in

---

[2]Again, this could almost certainly be done with less wires per channel, but that would only complicate the description.
[3]Actually, since 4 of those connections are to the endpoint nodes, we only have 8 to worry about for the tree wiring layers.
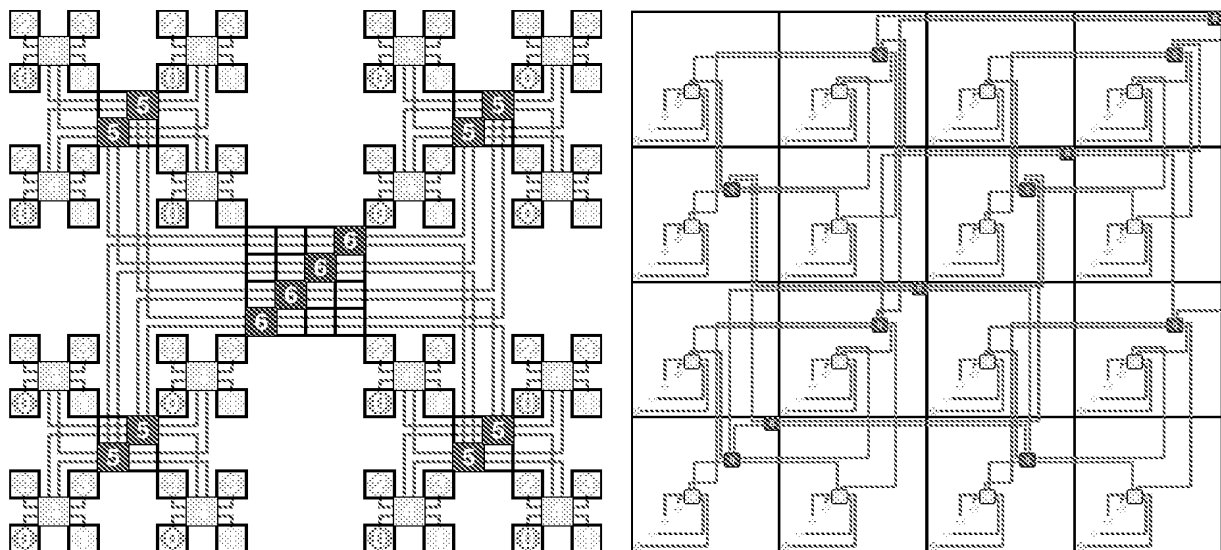
Figure 1: Butterfly Fat-Tree and Compact Layout
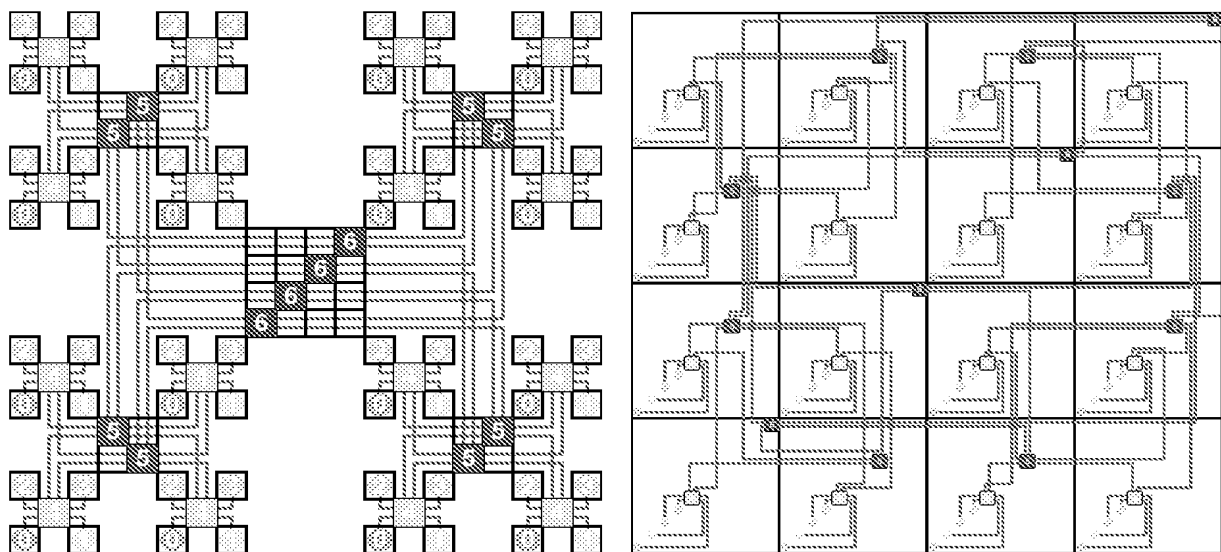
Figure 2: Rearranged Butterfly Fat-Tree and Compact Layout
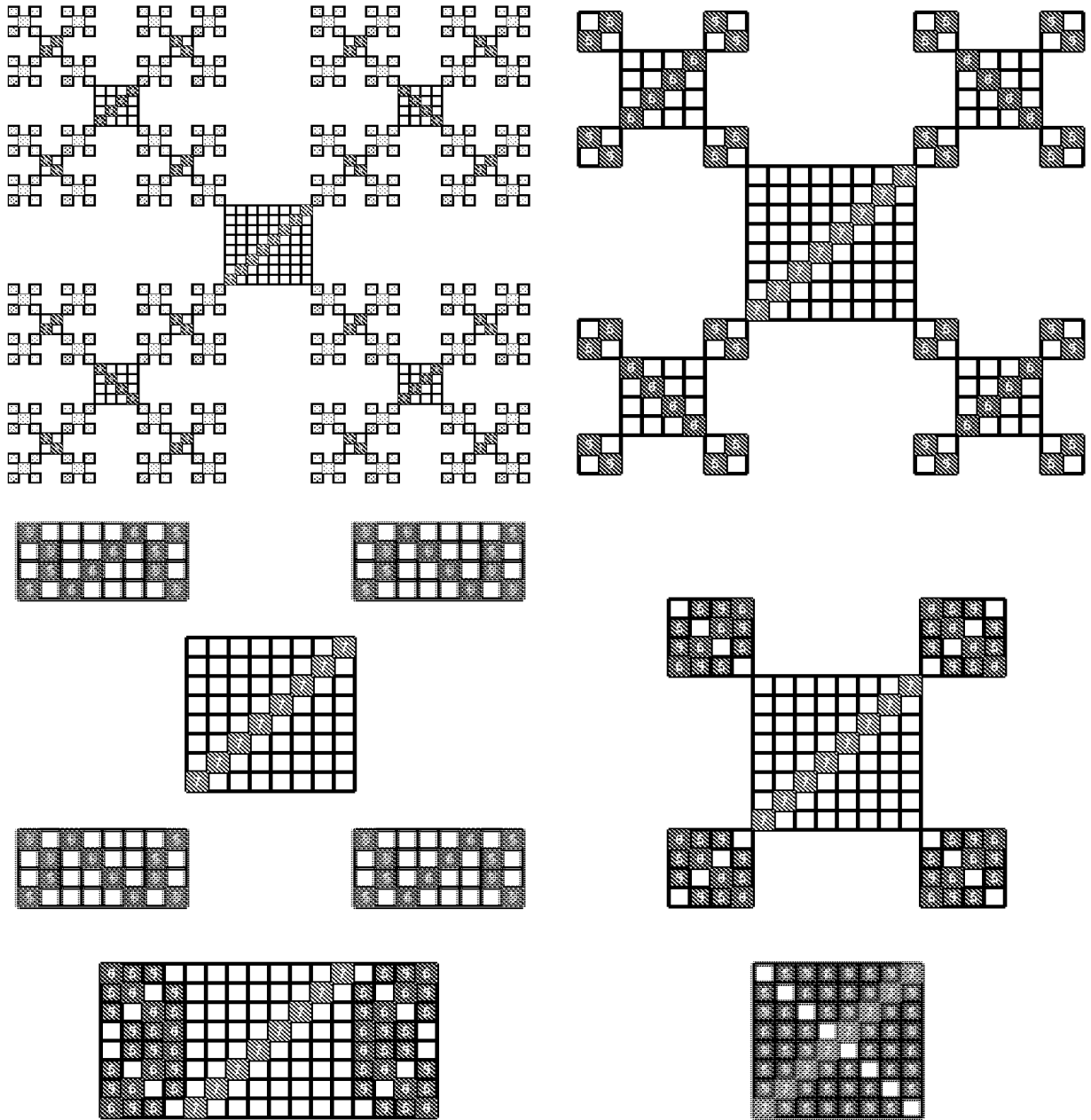
Figure 3: Fold Sequence for Rearranged Butterfly Fat-Tree
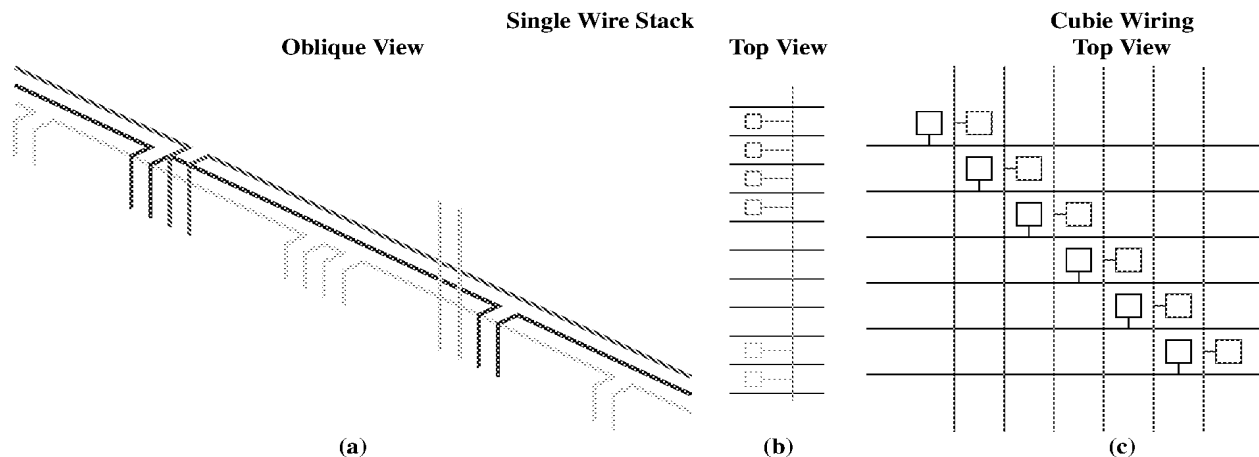
Figure 4: Wiring Pattern

constant, two-dimensional surface area if given $O(\log(N))$ wire layers.

In practice, we would not want to run a pair of wires directly in parallel for long runs due to potential coupling and hence crosstalk effects. Using standard techniques for twisting wires among the channels we can reduce the crosstalk coupling while maintaining our asymptotic bounds. Strictly speaking, adding a shielding layers between wiring runs would also provide such protection without changing the asymptotic bounds, but that should not be necessary.

Together with earlier observations about switch placement, this demonstrates our original claim that the entire butterfly fat-tree can be laid out compactly in $\Theta(N)$ active area and $\Theta(\log(N))$ wiring layers.

5. VLSI IMPLICATIONS
The immediate implication of this result is that we can use the butterfly fat-tree routing topology to compactly layout single-chip multiprocessors and FPGAs (*e.g.* [11]). This says that, given enough metal layers, we can layout an $\alpha = \frac{1}{2}$ bifurcating fat-tree [1], which Leiserson identifies as area universal [8], in area linear in the number of graph nodes. As noted above, this is better than the $O(N \log^2(N))$ area required if we limit the number of metal layers to a constant independent of N.

Empirically, one can observe that the number of metal layers *has been* steadily increasing with the active device capacity of our chips. Bohr observes that the number of metal layers has been increasing at the rate of 0.75 layers per IC generation [2]. Each generation represents a feature size reduction to $0.7\times$ the previous generation. Assuming die sizes stay roughly constant,[4] this means each generation roughly doubles the area for active computing devices. Adding a constant number of metal layers per capacity doubling represents a logarithmic growth, or the same asymptotic bound which we demonstrated above for the fat-tree.

Two-dimensional VLSI layout theory would have predicated that, if our circuits have interconnect as rich as $p \geq 0.5$ (Rent's Rule [6]) or natural bifurcators with $\alpha \geq \frac{1}{2}$, then the number of active

---

[4]Die sizes are not entirely constant, but this is a reasonable approximation for our purposes.

device we can usefully place on a VLSI component will scale sublinearly as devices are pushed out to accommodate the necessary interconnect wiring. This result and Bohr's suggest that processes have evolved to avoid this effect by correspondingly adding metal layers at a logarithmic rate to accommodate the richer interconnect requirements of our designs. Our results demonstrate that wiring layer growth is, in fact, sufficient to allow us to wire up universal routing structures efficiently; that is, with the logarithmic growth in wiring layers, we can place a number of active devices on the die which is linear in the total component area.

6. OPEN QUESTIONS
Can we layout an $\alpha > \frac{1}{2}$ tree in $O(N)$ area with any number of wire layers? with $O(N^{2p-1})$? how? A more general butterfly fat-tree can have a different growth rate in aggregate channel capacity [8] than the area-universal one where the main channel doubles when the subtree quadruples (matching he $\sqrt{A}$ perimeter I/O to area ratio of a two-dimensional layout). For any larger geometric channel growth rate (less than a complete doubling at every stage – *i.e.* $\alpha < 1$), the number of switches in the butterfly fat-tree will still be only $O(N)$. So, the question here, is: is there a similarly clever way to arrange the switches in this more general case? And, can the wiring and through via connections also be arranged to work out?

7. SUMMARY
We have noted that the assumption of a fixed number of wiring layers independent of device capacity does not match technology advances in modern VLSI. Using a multilayer model, we showed that the fat-tree can be arranged and laid out in $\Theta(N)$ area using $\Theta(\log(N))$ wiring layers. Finally, we noted that the growth rate derived here matches empirical observation of the growth rate of wiring layers in VLSI processes, suggesting that general designs have encountered similar wire limitations, encouraging processes to scale wire layers to meet wiring demands.

The primary contributions of this paper are:

- Show how to arrange the switches for folding so there is conveniently a constant number of switches along with each processing node tile (cubie).

- Show that the wiring can be arranged so as not to saturate intervening layers with through via connections.
- Assemble these two results to demonstrate the aforementioned claim for compact fat-tree layout.

8. ACKNOWLEDGMENTS

9. REFERENCES

[1] S. Bhatt and F. T. Leighton. A framework for solving vlsi graph layout problems. *Journal of Computer System Sciences*, 28:300–343, 1984.

[2] M. Bohr. Interconnect scaling – the real limiter to high performance ulsi. In *International Electron Devices Meeting 1995 Technical Digest*, pages 241–244. Electron Devices Society of IEEE, December 1995.

[3] R. Greenberg. The fat-pyramid and universal parallel computation independent of wire delay. *IEEE Transactions on Computers*, 43(12):1358–1365, December 1994.

[4] R. I. Greenberg and C. E. Leiserson. A compact layout for the three-dimensional tree of meshes. *Applied Math Letters*, 1(2):171–176, 1988.

[5] R. I. Greenberg and C. E. Leiserson. *Randomness in Computation*, volume 5 of *Advanes in Computing Research*, chapter Randomized Routing on Fat-Trees. JAI Press, 1988. Earlier version MIT/LCS/TM-307.

[6] B. S. Landman and R. L. Russo. On pin versus block relationship for partitions of logic circuits. *IEEE Transactions on Computers*, 20:1469–1479, 1971.

[7] C. E. Leiserson. Fat-trees: Universal networks for hardware efficient supercomputing. *IEEE Transactions on Computers*, C-34(10):892–901, Oct. 1985.

[8] C. E. Leiserson. Vlsi theory and parallel supercomputing. MIT/LCS/TM 402, MIT, 545 Technology Sq., Cambridge, MA 02139, May 1989. Also appears as an invited presentation at the 1989 Caltech Decennial VLSI Conference.

[9] W. T. Siegle. Interconnection technology for modern logic devices; an exercise in system engineering to assure manufacturability. In *Proceedings of the 1994 Materials Research Society Symposium*, volume 337, pages 3–11. Material Research Society, 1994.

[10] C. Thompson. Area-time complexity for vlsi. In *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*, pages 81–88, May 1979.

[11] W. Tsu, K. Macy, A. Joshi, R. Huang, N. Walker, T. Tung, O. Rowhani, V. George, J. Wawrzynek, and A. DeHon. Hsra: High-speed, hierarchical synchronous reconfigurable array. In *Proceedings of the International Symposium on Field Programmable Gate Arrays*, pages 125–134, February 1999.
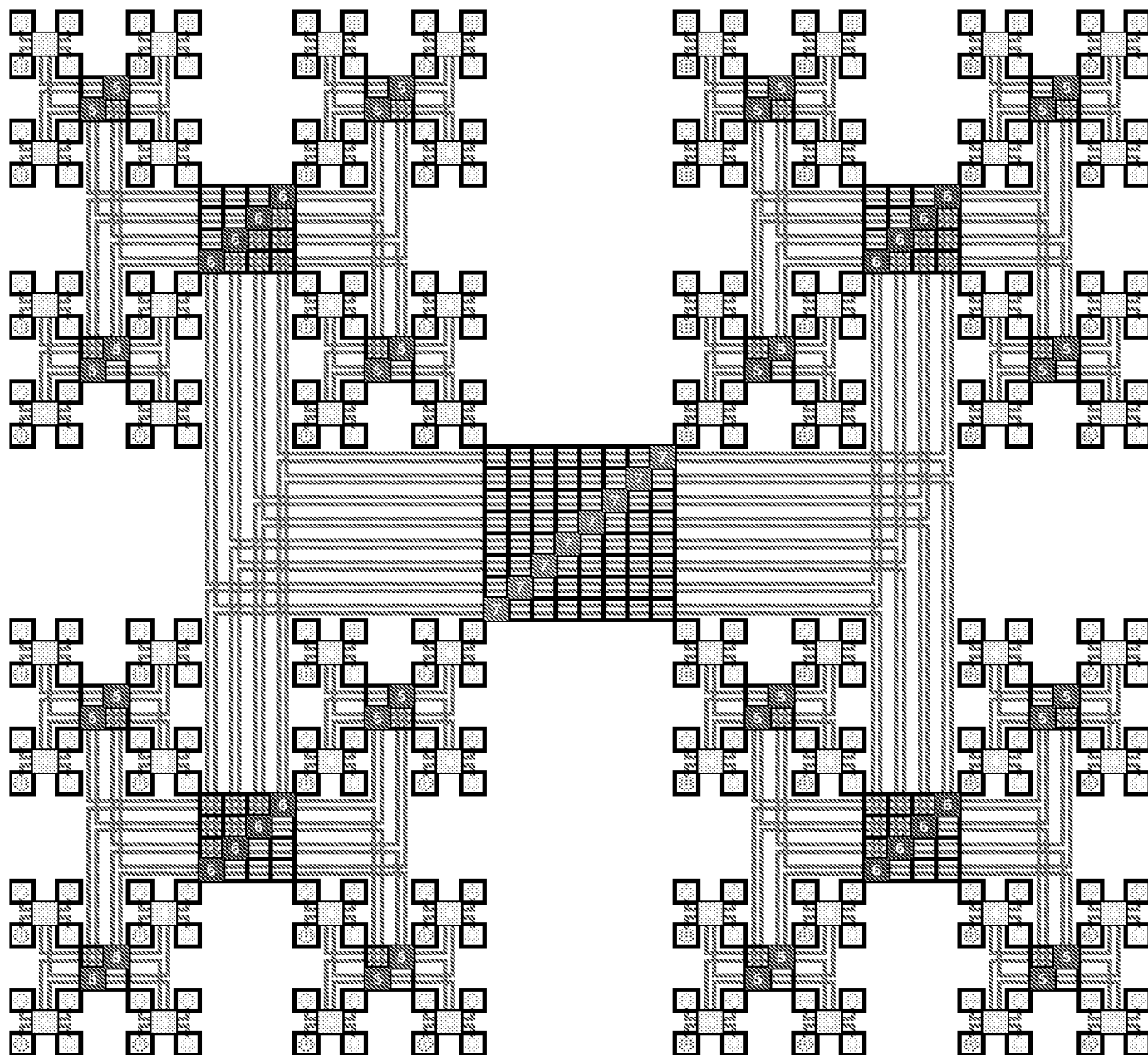
Figure 5: 256-node Butterfly Fat-Tree
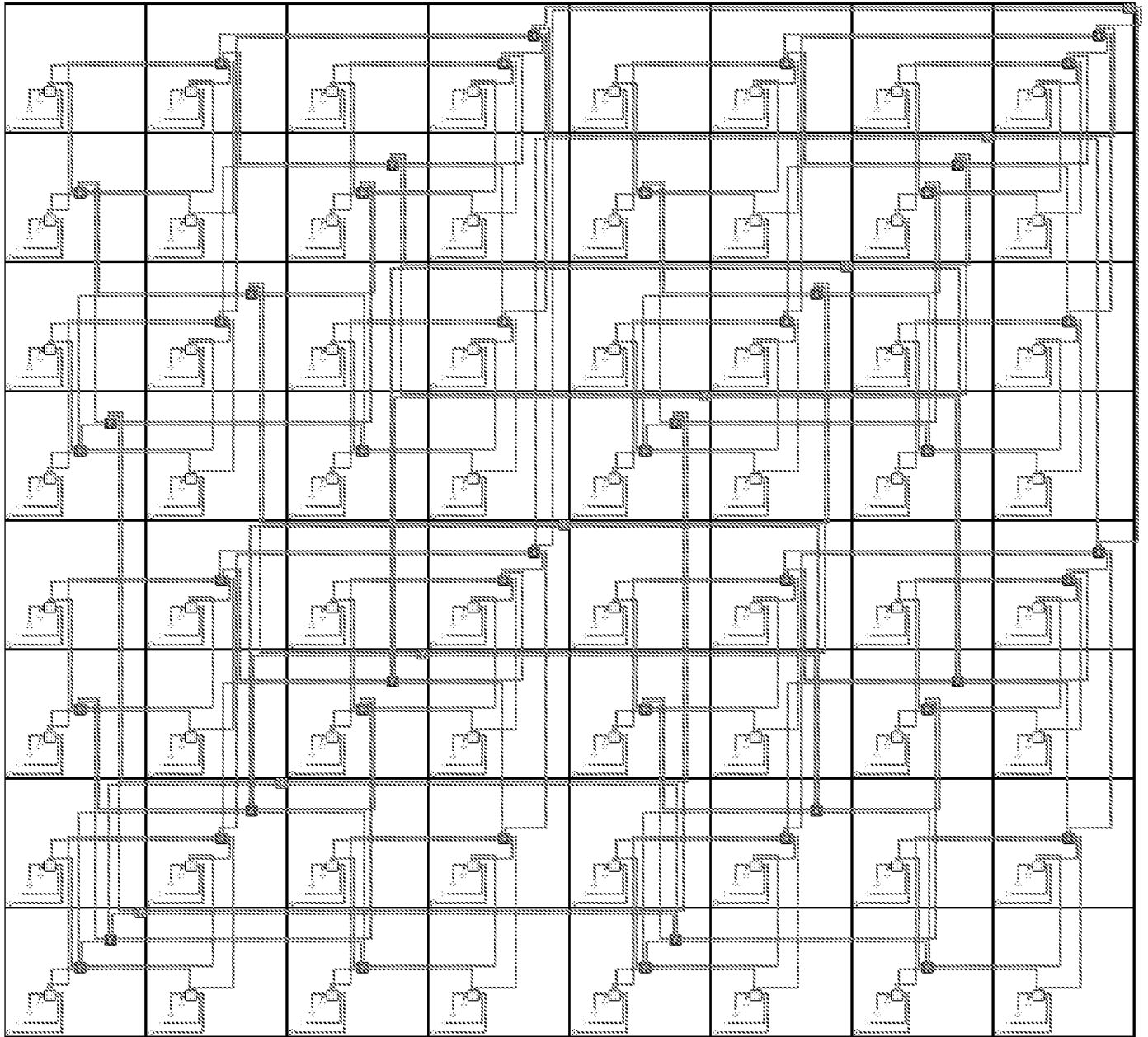
Figure 6: Fold-and-Squash Layout for 256-node Butterfly Fat-Tree
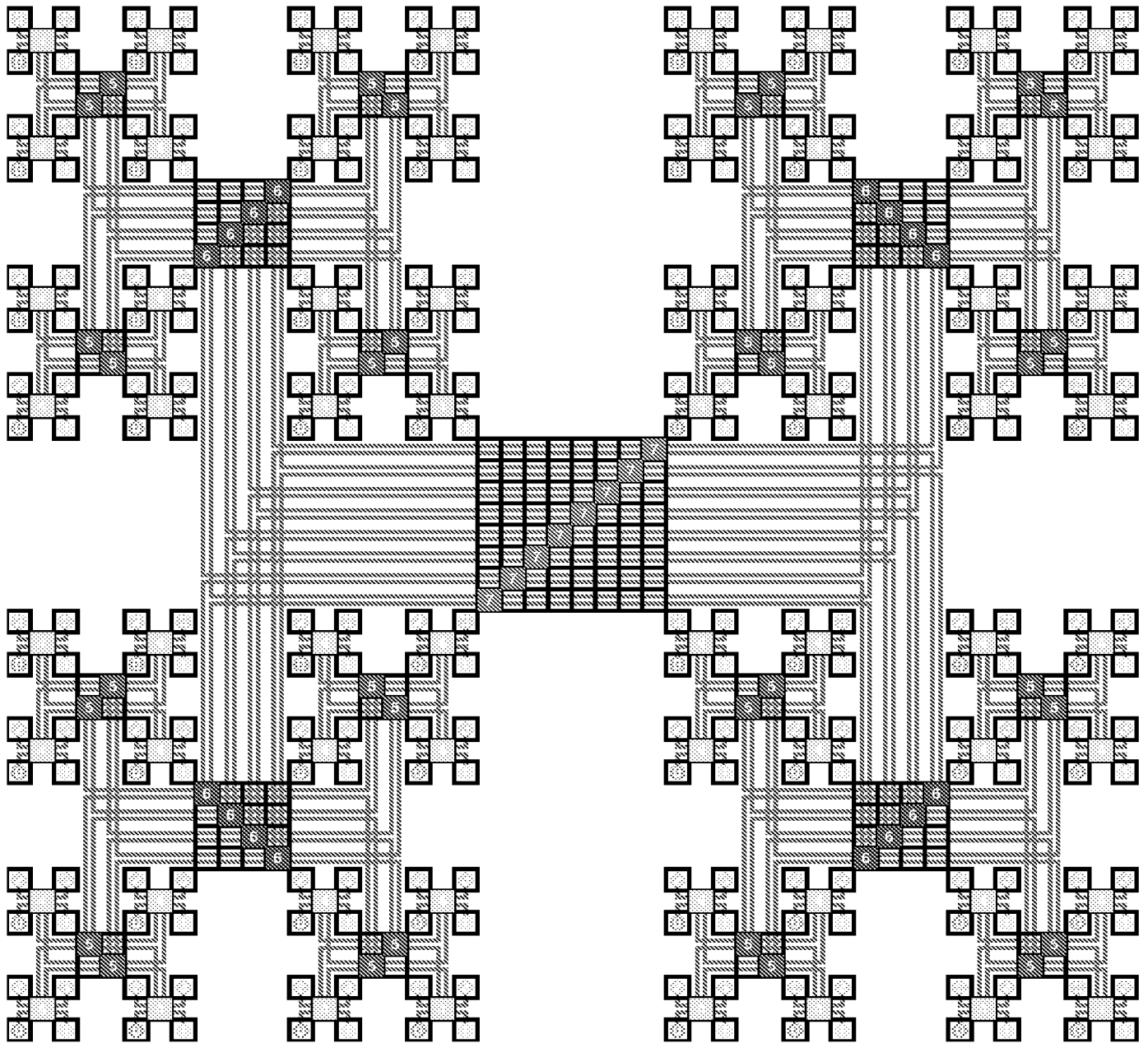
Figure 7: 256-Node Rearranged Butterfly Fat-Tree
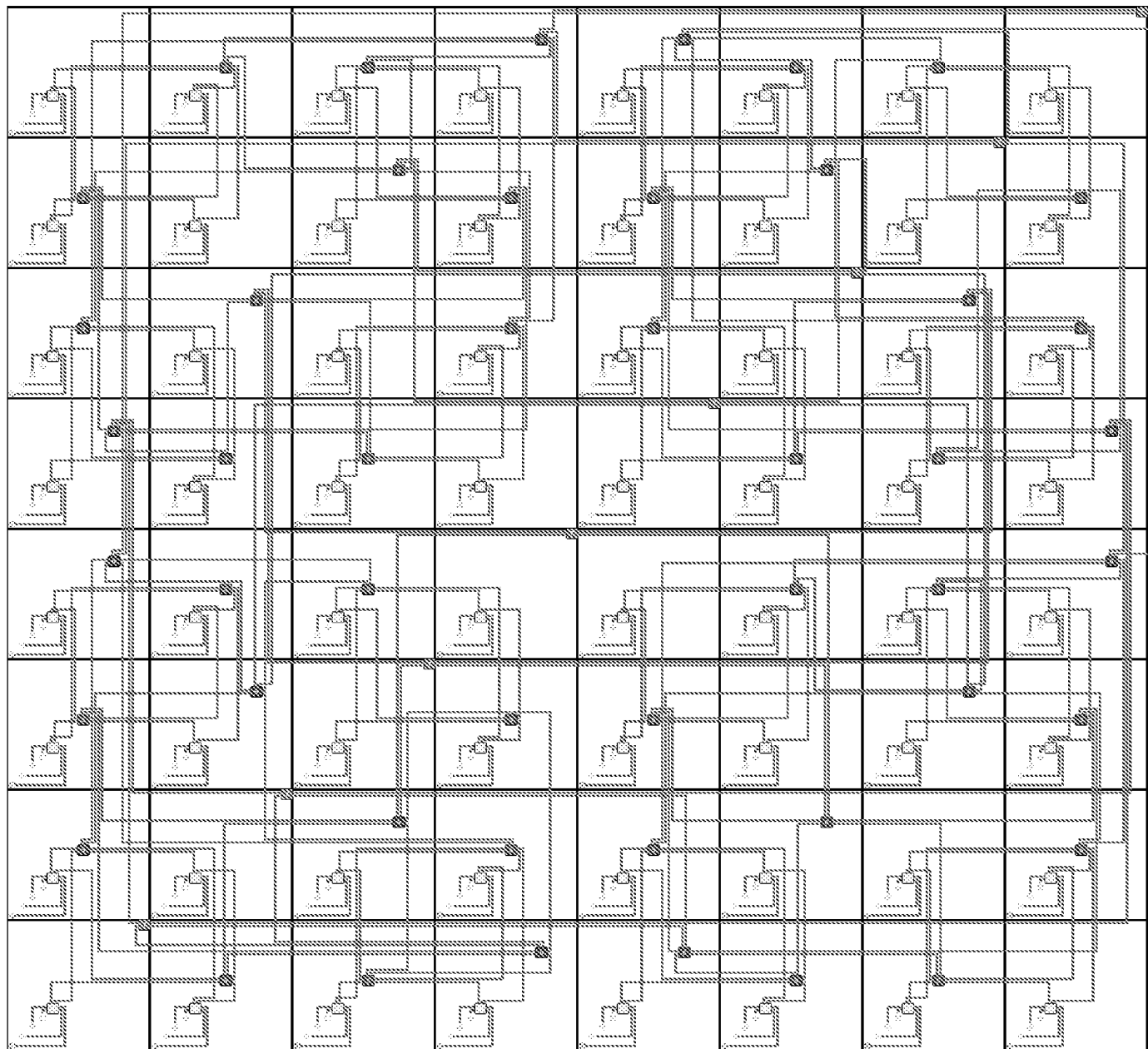
Figure 8: Fold-and-Squash Layout for 256-Node Rearranged Butterfly Fat-Tree

Segmented Channel Routing

Vwani P. Roychowdhury, Jonathan W. Greene, *Member, IEEE*, and Abbas El Gamal, *Senior Member, IEEE*

Abstract—Novel problems concerning routing in a segmented routing channel are introduced. These problems are fundamental to routing and design automation for field programmable gate arrays (FPGA's), a new type of electrically programmable VLSI. The first known theoretical results on the combinatorial complexity and algorithm design for segmented channel routing are presented. It is shown that the segmented channel routing problem is in general NP-complete. Efficient polynomial time algorithms for a number of important special cases are presented.

I. INTRODUCTION

CONVENTIONAL channel routing [1] concerns the assignment of a set of connections to tracks within a rectangular region. The tracks are freely customized by the appropriate mask layers. Even though the channel routing problem is in general NP-complete [4], efficient heuristic algorithms exist and are in common use in many placement and routing systems.

In this paper we investigate the more restricted channel routing problem (see Fig. 3), where the routing is constrained to use fixed wiring segments of predetermined lengths and positions within the channel. Such segmented channels are incorporated in channeled field programmable gate arrays (FPGA's) [3]. In [10], [11] we demonstrated that a well-designed segmented channel needs only a few tracks more than a freely customized channel. This leads us to believe that segmented channel routing is fundamental to routing for FPGA's.

The architecture of channeled FPGA's [3] is similar to that of conventional (mask programmed) gate arrays, comprising rows of logic cells separated by segmented routing channels (Fig. 1). The inputs and outputs of the cells each connect to a dedicated vertical segment. Programmable switches are located at each crossing of vertical and horizontal segments and also between pairs of adjacent horizontal segments in the same track. By programming a switch, a low-resistance path is created between the two crossing or adjoining segments.

A typical example of routing in a channeled FPGA is shown in Fig. 1. The vertical segment connected to the
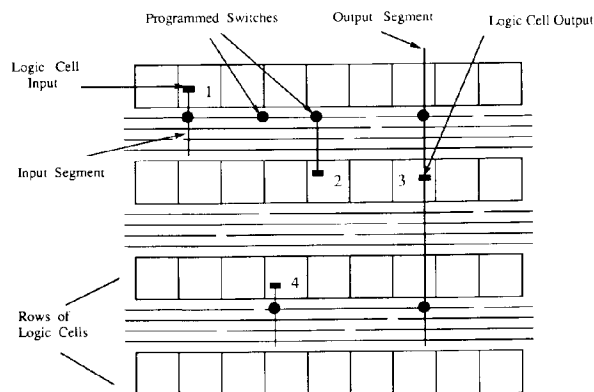
Fig. 1. FPGA routing architecture. ● denotes a programmed switch; unprogrammed switches are omitted for clarity.

output of cell 3 is connected by a programmed switch to a horizontal segment, which, in turn, is connected to the input of cell 4 through another programmed switch. In order to reach the inputs of cells 1 and 2, two adjacent horizontal segments are connected to form a longer one.

The choice of the wiring segment lengths in a segmented channel is driven by tradeoffs involving the number of tracks, the resistance of the switches, and the capacitances of the segments. These tradeoffs are illustrated in Fig. 2.

Fig. 2(a) shows a set of connections to be routed. With the complete freedom to configure the wiring afforded by mask programming, the *left edge* algorithm [5] will always find a routing using a number of tracks equal to the density of the connections (Fig. 2(b)). This is the case since there are no "vertical constraints" in the problems we consider.

In an FPGA, achieving this complete freedom would require switches at every cross point. Furthermore, switches would be needed between each two cross points along a wiring track so that the track could be subdivided into segments of arbitrary length (Fig. 2(c)). Since all present technologies offer switches with significant resistance and capacitance, this would cause unacceptable delays through the routing. Another alternative would be to provide a number of continuous tracks large enough to accommodate all nets (Fig. 2(d)). Though the resistance is limited, the capacitance problem is only compounded, and the area is excessive.

A segmented routing channel offers an intermediate approach. The tracks are divided into segments of varying
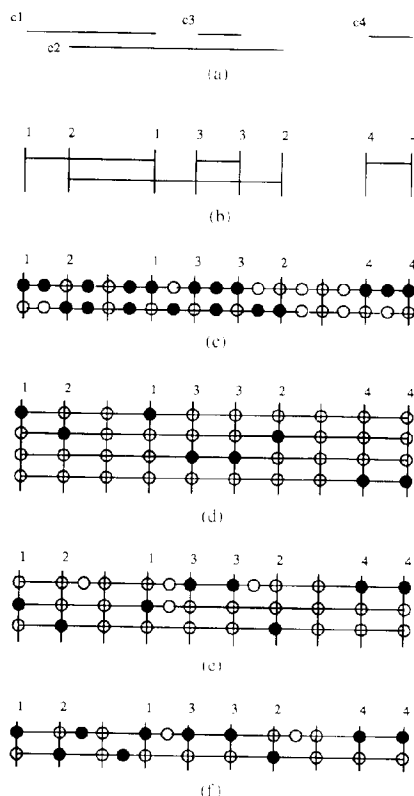
Fig. 2. Examples of channel routing. ○ denotes open switch; ● closed switch. (a) Set of connections to be routed. (b) Routing in unconstrained channels. (c) Routing in fully segmented channels. (d) Routing in an unsegmented channel. (e) Segmented for 1 segment routing. (f) Segmented for 2-segment routing.

lengths (Fig. 2(e)), allowing each connection to be routed using a single segment of the appropriate size. Greater routing flexibility is obtained by allowing limited numbers of adjacent segments in the same track to be joined end-to-end by switches (Fig. 2(f)). Enforcement of simple limits on the number of segments joined, or their total length, guarantees that the delay will not be unduly increased. Our results apply to the models of Fig. 2(e) and 2(f).

In Section II we formally define segmented channel routing and summarize the key results in the paper. Details of the algorithms and the proofs for theorems are given in Sections III–V and in the Appendix.

II. Definitions and Summary of Results

The input to a segmented channel routing problem, as depicted in Fig. 3, is a segmented channel consisting of a set $\mathfrak{I}$ of T tracks, and a set $\mathfrak{C}$ of M connections. The tracks are numbered from 1 to T. Each track extends from column 1 to column N, and is divided into a set of contiguous segments separated by switches. The switches are placed between two consecutive columns.

For each segment s, we define $left(s)$ and $right(s)$ to be

the leftmost and rightmost columns in which the segment is present, $1 \leq left(s) \leq right(s) \leq N$. Each connection c_i, $1 \leq i \leq M$, is characterized by its leftmost and rightmost columns: $left(c_i)$ and $right(c_i)$. Without loss of generality, we assume throughout that the connections have been sorted so that $left(c_i) \leq left(c_j)$ for $i < j$.

A connection c may be *assigned* to a track t, in which case the segments in track t that are present in the columns spanned by the connection are considered *occupied*. More precisely, a segment s in track t is occupied by the connection c if $right(s) \geq left(c)$ and $left(s) \leq right(c)$. In Fig. 3 for example, connection c_3 would occupy segments s_{21} and s_{22} in track 2 or segment s_{31} in track 3.

Definition 1—Routing: A routing, R, of a set of connections is an assignment of each connection to a track such that no segment is occupied by more than one connection.

A *K-segment* routing is a routing that satisfies the additional requirement that each connection occupies at most K segments.

We can now define the following segmented channel routing problems:

Problem 1—Unlimited Segment Routing: Given a set of connections and a segmented channel, find a routing.

To reduce the delay through assigned connections, it may be desirable to limit the number of segments used for each connection.

Problem 2—K-Segment Routing: Given a set of connections and a segmented channel, find a K-segment routing.

If is often desirable to determine a routing that is optimal with respect to some criterion. We may thus specify a weight $w(c, t)$ for the assignment of connection c to track t, and define:

Problem 3—Optimal Routing: Given a set of connections and a segmented channel, find a routing which assigns each connection c_i to a track t_i such that $\Sigma_{i=1}^{M} w(c_i, t_i)$, is minimized.

For example, a reasonable choice for $w(c, t)$ would be the sum of the lengths of the segments occupied when connection c is assigned to track t. Note also that with appropriate choice of $w(c, t)$, Problem 3 subsumes Problem 2.

The problems defined above consider segmented channel routing with the restriction that each connection may be assigned only to a single track. It is easy to see that the routing capacity of a segmented channel may be increased if a connection is assigned to segments in different tracks. For example, consider the segmented channel routing problem in Fig. 4. It can be easily shown that if the assignment of each connection is constrained to a single track, successful routing does not exist. However, by assigning connection c_3 to segments s_{11} and s_{33}, which are located in tracks t_1 and t_3, successful routing may be achieved. We refer to such a routing as generalized routing.

Definition 2—Generalized Routing: A generalized routing R_G, of a *set* of connections consists of an assign-
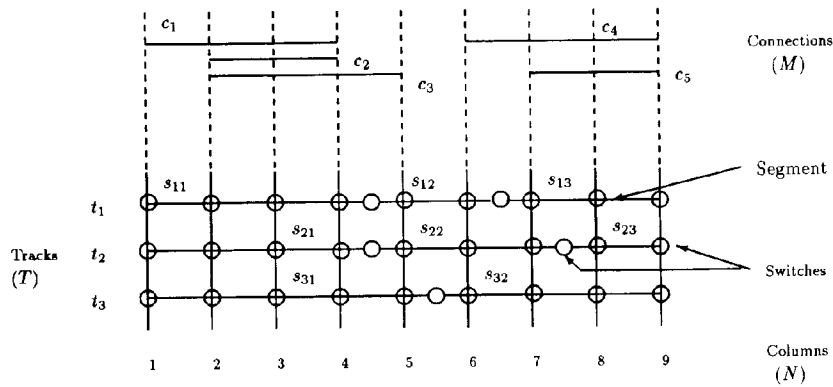
Fig. 3. An example of a segmented channel and a set of connections. $M = 5$, $T = 3$, $N = 9$. Connections: c_1, c_2, c_3, c_4, c_5. Segments: s_{11}, s_{12}, s_{13}, s_{21}, s_{22}, s_{23}, s_{31}, s_{32}.
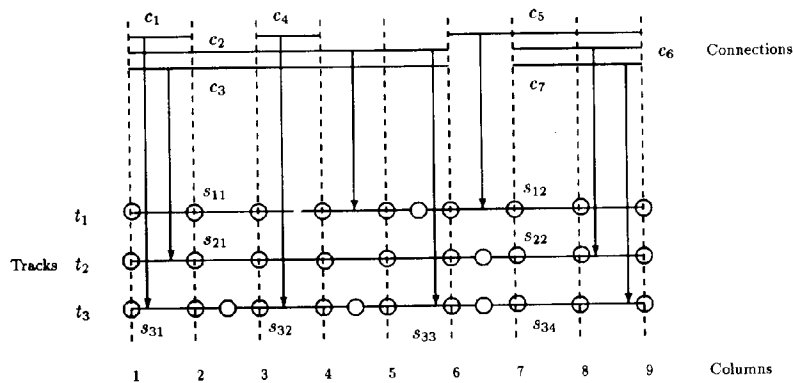
Fig. 4. An example where generalized routing is necessary for successful assignment.

ment of each connection to one or more tracks such that no segment is occupied by more than one connection.

Thus a generalized routing allows each connection $c = (left(c), right(c))$ to be *split* into $p(p \geq 1)$ parts: $(left(c), l_1)$, $(l_1 + 1, l_2)$, $(l_2 + 1, l_3)$, $\cdots$, $(l_{p-1} + 1, right(c))$, such that each part can be assigned to different tracks. A column l_i, where a connection is split, is referred to as a column where the connection c *changes* tracks.

Detailed hardware implementations may be developed to support generalized routing. For example, vertical wire segments may be added to facilitate track changing. In this case if a connection changes tracks, two switches must be programmed compared to only one if the connection is assigned to two contiguous segments in the same track. Thus allowing connections to occupy multiple tracks might lead to increase in area and to greater delays.

Motivated by such penalties, constraints may be imposed on the generalized segmented channel routing problem, leading to the following potentially important special cases.

1) Determine a generalized routing that uses at most k segments for routing any particular connection.

2) Determine a generalized routing that uses at most l different tracks for routing any connection.

3) Determine a generalized routing where connections can switch tracks only at predetermined columns.

We present preliminary results on the unconstrained version of generalized segmented channel routing problem.

Problem 4—Generalized Segmented Channel Routing: Given a set of connections and a segmented channel, find a generalized routing.

In this paper we establish the following results.

Theorem 1: Determining a solution to Problem 1 is strongly NP-complete.

Theorem 2: Determining a solution to Problem 2 is strongly NP-complete even when $K = 2$.

The reductions used to prove these theorems are rather tricky, and may have applications to problems in the area of task-scheduling on nonuniform processors. A proof of Theorem 1 is presented in Section III, and a proof of Theorem 2 is given in the Appendix.

We should note here that proving a given problem as NP-complete might not be enough to indicate its intractability. For example, many NP-complete problems such as Knap Sack have polynomial time solutions if all the input parameters are polynomially bounded in the input size. Strongly NP-complete problems however, remain

NP-complete even if input parameters are polynomially bounded; examples include TSP, Hamiltonian circuit, etc. (see [6] for detailed discussion on such issues). For Problems 1 and 2, the input parameters are indeed polynomially bounded in the number of columns (N) and tracks (T); for example, $M \leq TN$ and the lengths of the connections and the segments are bounded above by N. Hence, a proper approach would be to show that these problems are strongly NP-complete, which is what Theorems 1 and 2 establish.

Although Theorems 1 and 2 show that segmented channel routing is in general NP-complete, several special cases of the problem are tractable. We have developed polynomial-time algorithms for the following special cases:

Identically Segmented Tracks: Two tracks will be defined to be identically segmented if they have switches at the same locations, and hence, segments of the same length. The *left edge* algorithm used for conventional channel routing can be applied to solve Problems 1, 2, and 3.

1-Segment Routing: A routing can be determined by a linear time ($O(MT)$) greedy algorithm that exploits the geometry of the problem. The corresponding optimization problem can be also solved in polynomial time by reducing it to a weighted maximum bipartite matching problem.

At Most 2-Segments Per Track: If each track is segmented into at most two segments then also a greedy linear time algorithm (similar to the one for 1-Segment routing) can be designed to determine a routing.

We have also developed a general $O(T!M)$-time algorithm using dynamic programming for solving Problems 1, 2, and 3. This general algorithm can be adapted to yield more efficient algorithms for the following cases:

Fixed Number of Tracks: If the number of tracks is fixed, then the general algorithm directly yields a polynomial time algorithm.

K-Segment Routing: The general algorithm can be modified to yield an $O((K + 1)^T M)$-time algorithm. Note that for small values of K the modified algorithm performs better than the general one.

Fixed Types of Tracks: If the number of tracks is *unbounded* but the tracks are chosen from a fixed set where T_i is the number of tracks of type i, then an $O((\Pi_i T_i^{K-2})M)$ time (hence, a polynomial-time) algorithm can be designed.

Furthermore, we have developed a heuristic algorithm based on linear programming for solving Problems 1 and 2 that appears to work surprisingly well in practice.

The general algorithm and the above-mentioned special cases are described in Section IV.

In Section V we present preliminary results on the generalized segmented channel routing problem. In particular we show that Problem 4 admits a polynomial time algorithm if the number of tracks is bounded. Determining the exact complexity of the generalized segmented channel routing problem remains an open problem.

III. Complexity of the Segmented Routing Problem

In this section we prove Theorem 1, i.e., determining a solution to Problem 1 is strongly NP-complete. The proof of Theorem 2, i.e., determining a solution to Problem 2 is strongly NP-complete even when $K = 2$, is presented in the Appendix. The NP-completeness reductions for both the theorems is from the *Numerical Matching Problem with Target Sums*, which has been shown to be strongly NP-complete [7].

Numerical Matching with Target Sums [7]: Given a set $S = \{1, \cdots, n\}$, and positive integers $x_1, \cdots, x_n, y_1, y_n, z_1, \cdots, z_n$ with

$$\sum_{i \in S} (x_i + y_i) = \sum_{i \in S} z_i$$

do there exist permutations α and β of S such that $x_{\alpha(i)} + y_{\beta(i)} = z_i$ for all $i \in S$?

We assume without loss of generality that $x_1 < x_2 < \cdots < x_n$, $y_1 < y_2 < \cdots < y_n$, and $z_1 < z_2 < \cdots < z_n$. Furthermore, we assume that for any instance of the problem, we have $x_{i+1} - x_i \geq n$ and $x_1 + y_1 \geq x_n + n$. If these conditions are not met for an instance of the problem then one can define an equivalent problem (i.e., the modified problem has a solution if and only if the original problem has a solution) for which the conditions are met by performing the following transformations:

1) *Scaling:* Define $m = \lceil n / min(x_i - x_{i-1}) \rceil$. If $m > 1$ then set $x_i \leftarrow mx_i$, $y_i \leftarrow my_i$, and $z_i \leftarrow mz_i$.

2) *Translation:* Define $p = x_n + n - (y_1 + x_1)$. If $p > 0$ then set $y_i \leftarrow y_i + p$, and $z_i \leftarrow z_i + p$.

Given an instance of the Numerical matching problem $\mathfrak{N}$, we now show how to construct an instance of Problem 1 in (pseudo)-polynomial time; we shall refer to the segmented channel and the set of connections generated by the reduction procedure as $\mathfrak{Q}$.

The set of connections $\mathcal{C}$ is defined as follows.

1) For each x_i we define a connection a_i such that $left(a_i) = 4$, $right(a_i) = x_i + 3$. Thus, each connection a_i is of length $x_i - 1$, and starts at column number 4.

2) For each y_k, we define n connections $b_{k1}, \cdots, b_{kn}$ (one for each x_j) such that $left(b_{kj}) = x_j + 4 + (n - k)$ and $right(b_{kj}) = (y_k + x_j) + 4$. Note that $right(b_{kj}) - left(a_j) = x_j + y_k$.

3) n connections $d_1, \cdots, d_n$ are defined with $left(d_i) = 1$, and $right(d_i) = 3$.

4) $n^2 - n$ connections $e_1, \cdots, e_{n^2-n}$ are defined with $left(e_i) = 1$, and $right(e_i) = 5$.

5) n^2 connections $f_1, \cdots, f_{n^2}$ are defined with $left(f_i) = x_n + y_n + 5$ and $right(f_i) = x_n + y_n + 7$.

Set the number of columns in the construction to $N = x_n + y_n + 7$.

The set $\mathfrak{I}$ of n^2 tracks are then defined as follows.

1) For the first n tracks $t_1, \cdots, t_n$ each track t_i begins with a segment $(1, 3)$, followed by unit-length segments that span the region from column 4 to column $z_i + 4$ (i.e., there is a switch between every two columns between col-

ROYCHOWDHURY *et al*: SEGMENTED CHANNEL ROUTING   83

umn 4 and column $z_i + 4$), followed by a single segment of the form $(z_i + 5, N)$.

2) The rest of the $n^2 - n$ tracks are best described by dividing them into n blocks, each consisting of $n - 1$ tracks. Each such track comprises 3 segments.

The first block of $n - 1$ tracks, i.e., tracks $t_{n+1}, t_{n+2}, \cdots, t_{2n-1}$, are constructed using the definitions of the connections b_{1j}, $1 \leq j \leq n$. The segments in the track t_{n+1} are $(1, \text{left}(b_{11}) - 1)$, $(\text{left}(b_{11}), \text{right}(b_{12}))$, and $(\text{right}(b_{12}) + 1, N)$. That is, the middle segment in the track t_{n+1} is defined such that the connections b_{11} or b_{12} can be assigned to it. In general, the segments in each track t_{n+j}, $1 \leq j \leq n - 1$, are defined as $(1, \text{left}(b_{1j}) - 1)$, $(\text{left}(b_{1j}), \text{right}(b_{1(j+1)}))$, and $(\text{right}(b_{1(j+1)}) + 1, N)$. That is, the middle segment in the track t_{n+j}, $1 \leq j \leq n - 1$, is designed such that the connections b_{1j} or $b_{1(j+1)}$ can be assigned to it.

The ith block of $n - 1$ tracks (i.e., tracks $t_{n+(i-1)(n-1)+1}, \cdots, t_{n+i(n-1)}$) is constructed using the definitions of the connections b_{ij}, $1 \leq j \leq n$. The segments in the track $t_{(n+(n-1)(i-1))+j}$ (i.e., the jth track in the ith block) are $(1, \text{left}(b_{ij}) - 1)$, $(\text{left}(b_{ij}), \text{right}(b_{i(j+1)}))$, and $(\text{right}(b_{i(j+1)}) + 1, N)$. That is, the middle segment in the track $t_{(n+(n-1)(i-1))+j}$ is designed such that the connections b_{ij} or $b_{i(j+1)}$ can be assigned to it.

The following example illustrates this construction.

Example 1: Consider the unlimited segment routing problem (see Fig. 5) corresponding to the instance of the Numerical matching problem with Target Sums:

$$x_1 = 2, x_2 = 5, x_3 = 8, \quad y_1 = 9, y_2 = 11, y_3 = 12,$$

$$z_1 = 11, z_2 = 17, z_3 = 19. \qquad \square$$

We might note here that our proof of the NP-complete reduction is geometric in nature and it is helpful to use the above example in understanding the statement and the proof of each of the following propositions and lemmas. Before we proceed, however, let us define the following.

Two connections c_1 and c_2 will be said to *overlap* if they are present in the same column(s), i.e., $\text{left}(c_2) \leq \text{left}(c_1) \leq \text{right}(c_2)$ or $\text{left}(c_1) \leq \text{left}(c_2) \leq \text{right}(c_1)$.

A connection c_1 is said to *fit* in a segment S_1 if $\text{left}(c_1) \geq \text{left}(S_1)$ and $\text{right}(c_1) \leq \text{right}(S_1)$.

A segment is said to be *available* for a set of connections if it is unoccupied by the rest of the connections in $\mathcal{C}$.

Proposition 1: In any routing R of $\mathcal{Q}$ the following prevail.

a) The connections f_i, $1 \leq i \leq n^2$, are assigned to n^2 different tracks.

b) The connections d_i, $1 \leq i \leq n$, and a_i, $1 \leq i \leq n$, are assigned to tracks $t_1, \cdots, t_n$, and connections e_i, $1 \leq i \leq n^2 - n$, are assigned to tracks t_{n+1} through t_{n^2}.

Proof: Claim a) follows directly from the construction; i.e., the connections f_i, $1 \leq i \leq n^2$ are all identical and overlapping.

Claim b) follows from the following observations that are based on the above reduction.

1) Each connection e_i, $1 \leq i \leq (n^2 - n)$, overlaps with every other e_j. Each e_i also overlaps with every connection d_j, $1 \leq j \leq n$ and every connection a_k, $1 \leq k \leq n$.

2) In tracks t_1 through t_n a d_i and a_j can be assigned to the same track; such assignment is not possible for tracks t_i, where $i > n$.

3) Finally, it follows from 1), 2) and from the pigeonhole principle that if any e_i is assigned to a track t_j, $j \leq n$, then there would not be a sufficient number of tracks so as to assign all the connections d_i, $1 \leq i \leq n$, a_j, $1 \leq j \leq n$, and e_k, $1 \leq k \leq n^2 - n$. $\qquad \square$

Proposition 2: In any routing R of $\mathcal{Q}$, the segments available for assigning the connections a_i, $1 \leq i \leq n$, and b_{ij}, $1 \leq i, j \leq n$ are as follows.

a) In any track t_i, $1 \leq i \leq n$, the segments in columns 4 through $z_i + 4$ (i.e., the portion that is fully segmented) are available.

b) In any track t_i, $n + 1 \leq i \leq n^2$, only the middle segment is available.

Proof: Follows from Proposition 1: a) in any track t_i, $1 \leq i \leq n$, the first segment is always occupied by a d_j (for some $1 \leq j \leq n$), and the last segment is occupied by an f_k, hence the only available portion is the fully segmented part of the track; b) every track t_i, $n + 1 \leq i \leq n^2$, has only three segments, and from Proposition 1 we know that the left segment is occupied by a connection e_j (for some $1 \leq j \leq n^2 - n$) and the right segment by another connection f_k, $1 \leq k \leq n^2$. $\qquad \square$

The following proposition shows that in any routing R of $\mathcal{Q}$, every track has exactly one b_{ij} assigned to it.

Proposition 3: All connections b_{ij}, $1 \leq i, j \leq n$, overlap; hence, they have to be assigned to different tracks.

Proof: Given the geometry of our construction, it suffices to show that b_{11} and b_{1n} overlap. Now $\text{right}(b_{11}) = x_1 + y_1 + 4$, and $\text{left}(b_{1n}) = x_n + 4 + (n - 1) = x_n + n + 3$. Hence, $\text{right}(b_{11}) - \text{left}(b_{1n}) = x_1 + y_1 - (x_n + n - 1)$, which is strictly greater than 0 by our assumptions. $\qquad \square$

We can now show one direction of the reduction procedure.

Lemma 1: If the given Numerical Matching problem with target sums has a solution, then there exists a routing R for $\mathcal{Q}$.

Proof: Suppose there exist permutations α and β such that $x_{\alpha(i)} + y_{\beta(i)} = z_i$ for all $1 \leq i \leq n$. Then we can define a routing R for $\mathcal{Q}$ as follows.

1) Connections d_i, $1 \leq i \leq n$, e_i, $1 \leq i \leq n^2 - n$, and f_i, $1 \leq i \leq n^2$, are assigned according to Proposition 1.

2) For every i, $1 \leq i \leq n$, connections $a_{\alpha(i)}$ and $b_{\beta(i)\alpha(i)}$ are assigned to track t_i. Since $x_{\alpha(i)} + y_{\beta(i)} = z_i$, one can easily show that the connections can be appropriately assigned in the available segments (see also Proposition 2).

At this stage, for every i, $1 \leq i \leq n$, all except one connection among the connections b_{ij}, $1 \leq j \leq n$, need to be routed.

3) Consider the connections b_{1j}, $1 \leq j \leq n$. Let b_{1k} be

the connection that has been assigned to one of the tracks t_i, $1 \leq i \leq n$. Recall that the tracks $t_{n+1}-t_{n+(n-1)}$ were designed using the definitions of b_{1j}, $1 \leq j \leq n$, and that the middle segment in track t_{n+1} can accommodate either connection b_{11} or connection b_{12}. So assign b_{11} to track t_{n+1} and repeat this procedure by assigning connections $b_{12}-b_{1(k-1)}$ to tracks $t_{n+2}-t_{n+(k-1)}$. Now b_{1k} has already been assigned, hence one has to assign connections $b_{1(k+1)}-b_{1n}$. By construction, however, $b_{1(k+1)}$, can be assigned to track t_{n+k}, and this assignment procedure can be continued by assigning $b_{1(k+2)}$ to track $t_{n+(k+1)}$, and so on.

In general, for any i the unassigned $n - 1$ connections among b_{ij}, $1 \leq j \leq n$ can be assigned to the ith block of tracks (i.e., tracks $t_{n+(i-1)(n-1)+1}, \cdots, t_{n+i(n-1)}$) by following the same procedure as above.

Next we show that if Q has a valid routing then there is a solution for the numerical matching problem $\mathfrak{N}$. The following definitions that capture the geometry of the routing problem Q will be helpful:

It is clear from Propositions 1, 2, and 3 that each track t_l, $1 \leq l \leq n$ has one connection from a_i, $1 \leq i \leq n$ and one connection from b_{kj}, $1 \leq k, j \leq n$ assigned to it. Also, note that since the parts of the first n tracks that are available for the connections a_i, $1 \leq i \leq n$ and b_{kj}, $1 \leq k, j \leq n$ are fully segmented, two connections, a_i and b_{kj}, can be assigned to the same track only if they do not overlap.

We define the *length* or *space* occupied by the connections a_i and b_{kj} assigned to some track t_l ($1 \leq l \leq n$) as equal to right(b_{kj})-left(a_i). That is, the length (or space) occupied by the two connections is the geometrical length from the left end of the connection a_i to the right end of the connection b_{kj}.

Claim 1: It follows from Proposition 2 that the total length (or space) available in the first n tracks for assigning the connections a_i, $1 \leq i \leq n$ and b_{ij}, $1 \leq i, j \leq n$ is $\Sigma_1^n z_i$.

The above claim follows immediately from the observation that the only portion of each track t_l to which a_i and b_{ij} can be assigned is of length z_i.

Proposition 4: Connections a_i and b_{kj} cannot be assigned to the same track if $j < i$.

Proof: left(b_{kj}) $= x_j + 4 + (n - k)$ and right(a_i) $= x_i + 3$. Thus right(a_i) − left(b_{kj}) $= x_i - (x_j + n) + k - 1$. However, $(k - 1) \geq 0$, and by our assumptions $x_i - (x_j + n) \geq 0$ for all $j < i$. Hence, a_i and b_{kj} overlap for $j < i$.

Proposition 5: If a_i and b_{kj} ($j \geq i$) are assigned to the same track t_l ($1 \leq l \leq n$) then the length occupied in the track t_l is $x_j + y_k$ ($\geq x_i + y_k$).

Proof: Left(a_i) $= 4$, and right(b_{kj}) $= x_j + y_k + 4$. Hence, right(b_{kj}) − left(a_i) $= x_j + y_k \geq x_i + y_k$ (because by our assumption $j \geq i$ implies that $x_j \geq x_i$).

The next two propositions use the definitions of the tracks $t_{n+1}, \cdots, t_{n^2}$, and determine the restrictions on possible assignments of the connections b_{ij} to these tracks.

Proposition 6: None of the connections b_{kj} for $k > 1$ can be assigned to tracks $t_{n+1}-t_{n+(n-1)}$.

Proof: Recall that the tracks $t_{n+1}-t_{n+(n-1)}$ were constructed using the connections b_{1j}, $1 \leq j \leq n$. Now consider any track t_{n+l}. From Proposition 1 we know that its end segments are already occupied. Hence, for any b_{kj} to be assigned to this track, it must fit within the middle segment (left(b_{1l}), right $(b_{1(l+1)})$).

First, consider the case where $k > 1$ and $j \leq l$. Recall that left(b_{kj}) $= x_j + (n - k) + 4$; since $j \leq l$, we have $x_j \leq x_l$ and since $k > 1$, we can write left(b_{kj}) $= x_j + (n - k) + 4 < x_l + (n - 1) + 4 = $ left(b_{1l}). Hence, b_{kj} cannot be assigned to track t_{n+l}.

Next, consider the case where $k > 1$ and $j > l$. Recall that right(b_{kj}) $= x_j + y_k + 4$; since $j \geq (l - 1)$, we have $x_j \geq x_{l+1}$; furthermore, $k > 1$ implies that $y_k > y_1$. Hence, right(b_{kj}) $= x_j + y_k + 4 > x_{l+1} + y_1 + 4 = $ right($b_{1(l-1)}$). Therefore, b_{kj} cannot be assigned to track t_{n-l}.

Proposition 7: In general, none of the connections b_{kj} for $k > i$ can be assigned to tracks $t_{n+(n-1)(i-1)+1}-t_{n+(n-1)i}$. Hence, none of the connections b_{kj} for $k > i$ can be assigned to tracks $t_{n+1}-t_{n+(n-1)i}$.

Proof: Recall that the tracks under consideration were constructed using the definitions of b_{ij}, $1 \leq j \leq n$. The proof then follows along the lines of the previous proposition.

Let R be any routing of Q, then we define m_i as follows:

$$m_i = |\{b_{ij} : 1 \leq j \leq n, \text{ and } b_{ij} \text{ is assigned to some track } t_l, 1 \leq l \leq n, \text{ in } R\}|.$$

In other words, m_i is the number of connections from the set $\{b_{i1}, b_{i2}, \cdots, b_{in}\}$ that are assigned to the first n tracks (i.e., $t_1, t_2, \cdots, t_n$). Propositions 8–10, following show that in any valid routing R of Q, $m_i = 1$, for all $1 \leq i \leq n$.

Proposition 8: $\Sigma_1^k m_i \leq k$, $\forall 1 \leq k \leq n$ and $\Sigma_1^n m_i = n$.

Proof: Each track has exactly one connection b_{ij} (for some i and j) assigned to it. Hence, by definition $\Sigma_1^n m_i = n$.

To show that $\Sigma_1^k m_i \leq k$ for every $1 \leq k \leq n$, first consider $k = 1$. Suppose that $m_1 > 1$, then exactly $n - m_1$ connections from among the connections b_{1j}, $1 \leq j \leq n$ are assigned to tracks $t_{n+1} - t_{n^2}$. Even if all of them were assigned to tracks in the first block (i.e., among $t_{n+1}, \cdots, t_{n+n-1}$), there would be $(m_1 - 1) \geq 1$ tracks in the block that are left unassigned. However, by Proposition 6, no connection b_{ij}, when $i > 1$ can be assigned to any track among $t_{n+1}, \cdots, t_{n+n-1}$. Thus, at least $(m_1 - 1)$ tracks among $t_{n+1}, \cdots, t_{n+n-1}$ have no connection b_{ij} assigned to them. This leads to a contradiction (because every track has exactly one b_{ij} assigned to it).

Using Proposition 7, the same arguments can be applied for any $k > 1$. That is for $k = 2$, one can show (using Proposition 7) that if $m_1 + m_2 > 2$, then some

tracks among t_{n+1} through $t_{n+2(n-1)}$ do not have any connection b_{ij} assigned to them. □

Proposition 9: Let $w_1 < w_2 < \cdots w_n$ be a sequence of positive integers and let non-negative integers m_i, $1 \leq i \leq n$, satisfy the following relations: $\Sigma_1^k m_i \leq k$, $\forall 1 \leq k \leq n$, and $\Sigma_1^n m_i = n$. Then $\Sigma_1^n m_i w_i > \Sigma_1^n w_i$ if and only if some of the m_i are 0.

Proof: First we observe that if there exists an $m_j > 1$, then there exists $l = m_j - 1$ distinct variables $m_{j_1}, \cdots, m_{j_l}$, such that all of them are 0 and $j_i < j$. If not, then one can easily show that $\Sigma_1^j m_i > j$, which is a contradiction. Thus, if any of the variables $m_i > 1$, then it always forces some m_k to equal 0 such that $k < i$. Hence, $\Sigma_1^n m_i w_i > \Sigma_1^n w_i$ if and only if some of the m_i are 0. □

Proposition 10: In any routing R, $m_i = 1 \ \forall \ 1 \leq i \leq n$, i.e., in every routing only one connection from the set $\{b_{i1}, \cdots, b_{in}\}$ is assigned to one of the first n tracks.

Proof: If a_i and b_{kj} are assigned to the same track then from Proposition 5 we know that the length occupied is $\geq x_i + y_k$. Now, by definition m_k connections from among b_{kj}, $1 \leq j \leq n$ appear in the first n tracks. Hence, the total length occupied by the connections a_i, $1 \leq i \leq n$, and the connections b_{ij}, $1 \leq i, j \leq n$ that are assigned in the first n tracks is $\geq \Sigma_1^n x_i + \Sigma_1^n m_k y_k$.

If at least one m_k is 0, then Proposition 9 implies $\Sigma_1^n m_k y_k > \Sigma_1^n y_k$ (because $y_1 < y_2 < \cdots y_n$). Hence, the total length occupied by the connections a_i and b_{ij} in the first n tracks is $> \Sigma_1^n x_i + \Sigma_1^n y_k = \Sigma_1^n z_i$. This leads to a contradiction because Proposition 2 and Claim 1 show that the total space available is equal to $\Sigma_1^n z_i$. Hence, $m_i = 1$ $\forall 1 \leq i \leq n$. □

Lemma 2: If there is a routing for $\mathfrak{Q}$, then there exists a solution to $\mathfrak{N}$.

Proof: Proposition 10 shows that $\forall i$ only one connection among $\{b_{i1}, \cdots, b_{in}\}$ is assigned to one of the first n tracks. By Proposition 5, if a_i and b_{kj} ($j \geq i$) are assigned to the same track then the length occupied is $x_j + y_k$ ($\geq x_i + y_k$). Hence, the total length occupied by the connections is $\geq \Sigma_1^n x_i + \Sigma_1^n y_k = \Sigma_1^n z_i$.

Claim a: A connection a_i can only be assigned to the same track with some b_{ki}.

Proposition 4 shows that if a_i and b_{kj} are assigned to the same track, then $j \geq i$. Now if a_i is matched with some b_{kj} and $j > i$, then the length occupied is $x_j + y_k > x_i + y_k$. Hence, the total length occupied by all the connections in the first n tracks is greater than $\Sigma_1^n x_i + \Sigma_1^n y_k = \Sigma_1^n z_i$. However, this leads to a contradiction since the total space available in the first n tracks is $\Sigma_1^n z_i$ (Claim 1). Hence, a_i can be assigned only to the same track as some b_{ki}.

It follows then that if we define the connections assigned to track t_i, $1 \leq i \leq n$, as $a_{\alpha(i)}$ and $b_{\beta(i)\alpha(i)}$, then α and β are permutations of the set $\{1, \cdots, n\}$. Also, by our convention the total length occupied in track t_i by $a_{\alpha(i)}$ and $b_{\beta(i)\alpha(i)}$ is $= x_{\alpha(i)} + y_{\beta(i)}$.

Claim b: $x_{\alpha(i)} + y_{\beta(i)} = z_i$.

Suppose this is not the case for some i, $1 \leq i \leq n$.

Then it implies that in the track t_i, the length occupied by the connections $a_{\alpha(i)}$ and $b_{\beta(i)\alpha(i)}$ is $< z_i$. Now by Proposition 2, in any track t_k ($1 \leq j \leq n$) the space available for assigning the connections a_i and b_{ij} is z_k. Hence, the length occupied by the connections a_i and the connections b_{ij} in the first n tracks is $< \Sigma_1^n z_i$. However, this leads to a contradiction because we showed that the length occupied is $\geq \Sigma_1^n z_i$.

Thus, the assignment of connections to the first n tracks defines permutations α and β such that $\forall i$, $x_{\alpha(i)} + y_{\beta(i)} = z_i$. □

Theorem 1: Determining a solution to Problem 1 is strongly NP-complete.

Proof: Follows from Lemmas 1 and 2: □

IV. Algorithms for Segmented Routing

In this section we present algorithms for various special cases of Problems 1–3. We first discuss algorithms that exploit the geometry of the segmented channels. We then discuss a general algorithm based on dynamic programming. Finally, we discuss a heuristic algorithm (based on linear programming) that appears to work surprisingly well in practice.

A. Geometrical Algorithms

Identically Segmented Tracks: If all tracks are identically segmented (i.e., the locations of the switches are the same in every track), then Problems 1 and 2 can be solved by the left-edge algorithm [5] in time $O(MT)$. Assign the connections in order of increasing left ends as follows: assign each connection to the first track in which none of the segments it would occupy are yet occupied.

Note that the density of the connections does not provide an upper bound on the number of tracks required for routing (as is the case for conventional routing when the left-edge algorithm is used in the absence of vertical constraints). However, if prior to computing the density the ends of each connection are extended until a column adjacent to a switch is reached, then the density would be a valid upper bound.

1-Segment Routing: If we restrict consideration to 1-segment routings, Problem 2 can be solved by the following greedy algorithm.

The connections are assigned in order of increasing left ends as follows. For each connection, find the set of tracks in which the connection would occupy one segment. Eliminate any tracks where this segment is already occupied. From among the remaining tracks, choose one where the unoccupied segment's right end is closest to the left (i.e., the right end coordinate of the segment in the chosen track is the smallest), and assign the connection to it. If there is a tie, then it is broken arbitrarily. In the example of Fig. 3, the algorithm assigns c_1 to s_{11}, c_2 to s_{21}, c_3 to s_{31}, c_4 to s_{32}, and c_5 to s_{13}. The time required is $O(MT)$.
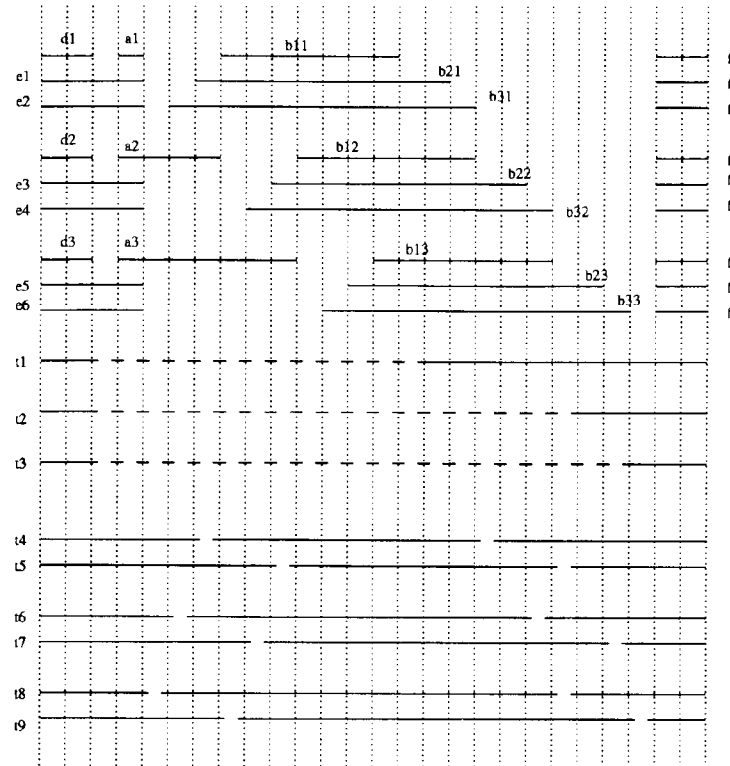
Fig. 5. The segmented channel and connections for Example 1.

Next, we show that if some connection cannot be assigned to any track, then no complete routing is possible.

Theorem 3: The above algorithm solves Problem 2 if $K = 1$.

Proof: It suffices to show that if there is a routing for Problem 2 with $K = 1$, then it can always be modified to obtain an assignment that the above algorithm would generate.

Let R be a routing with $K = 1$. Consider the leftmost connection c_1. Let F_1 be the set of segments that c_1 can be assigned to, and let S_1 be the set of segments in F_1 with the minimum right edge. There now are three possible cases.

1) In R, c_1 has been assigned to one of the segments in S_1. In such a case, no modification is necessary; the assignment of c_1 is according to the above algorithm.

2) In R, c_1 has been assigned to some $s \notin S_1$, and that there is at least one unoccupied segment in S_1. Then assign c_1 to one of the unoccupied segments in S_1.

3) In R, c_1 has been assigned to some $s \notin S_1$, and every segment in S_1 is occupied. In that case choose some c_i that occupies a segment $s_1 \in S_1$. We can now always interchange the assignments, i.e., assign c_i to s and assign c_1 to $s_1 \in S_1$. Thus a new assignment is obtained where c_1 is assigned according to the above algorithm.

The justification for swapping is as follows (see Fig. 6). Since $s_1 \in S_1$, c_1 can always be assigned to it. More-

over, the left edge of s is at or to the left of c_i (because left edge of c_i is at or to the right of c_1) and the right edge of s is to the right of c_i (because by definition of S_1 the right edge of s is to the right of s_1). Hence, c_i can be assigned to s.

The above procedure can be continued for c_2 and other connections until a modified routing R' is obtained that satisfies the conditions of the above algorithm.

For 1-segment routing, Problem 3 may be solved efficiently by reducing it to a bipartite matching problem. Fig. 7 shows the graph corresponding to the routing problem in Fig. 3. The left side has a node for each connection and the right side a node for each segment. An edge is present between a connection and a segment if the connection can be assigned to the segment's track. The weight $w(c, t)$ is assigned to the edge between connection c and a segment in track t. A minimum-weight matching indicates an optimal routing. The time required using the best known matching algorithm (see [6]) is $O(V^3)$, where $V \leq M + NT$ is the number of nodes.

At Most 2-Segments Per Track: In a track with two-segments, the first segment from the left will be referred to as the *initial* segment and the next one will be referred to as the *end* segment. If the track is unsegmented, i.e., it has only one segment, then for our purposes we will refer to the only segment as an end segment.

The following greedy algorithm, which is similar to the

ROYCHOWDHURY *et al.* SEGMENTED CHANNEL ROUTING

87
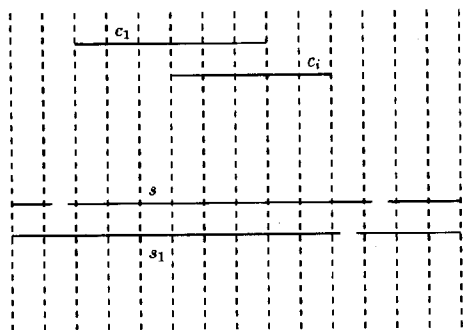
Fig. 6. An example of assignment modification in a 1-segment routing. If connection c_1 is assigned to s and c_i to s_1 such that $right(s) > right(s_i)$, then the assignment can always be swapped.

Fig. 7. Bipartite graph for 1-segment routing of the problem in Fig. 3.

one for 1-segment routing, can be used to determine a solution to Problem 1:

The connections are assigned in order of increasing left ends (ties are resolved arbitrarily). During the execution of the algorithm a track will be considered *unoccupied* if no connection has been assigned to it.

Now for each connection, determine the set of tracks in which the connection would occupy a single segment. Eliminate any track where this segment is already *occupied*. Now consider the following two cases:

Case 1: If no track is available (i.e., after the above-mentioned elimination of tracks), then append the connection to the pool, P, of unassigned (but already examined) connections.

Case2: If tracks are available, then assign the connection to a track where the unoccupied segment's right end is closest to the left (i.e., the right-end coordinate of the segment in the chosen track is the smallest). If more than one track qualifies, then the tie is broken arbitrarily.

Next, if $|P|$ (i.e., the number of unassigned, but already examined, connections) equals the number of tracks unoccupied by any connection, then assign the connections in P to these unoccupied tracks in any order; mark these tracks as occupied, and remove the assigned connections from P. Else, if $|P|$ is greater than the number of such unoccupied tracks, then stop and signal that no valid routing is possible.

Continue with the next connection.

When all the connections are examined and pool P is nonempty, then assign the connections in P to unoccupied tracks.

In the example shown in Fig. 8, the above algorithm would assign c_1 to track t_1 and append c_2 to the pool P. For c_3, both tracks t_2 and t_3 are eligible, and let the tie be broken by assigning c_3 to track t_3. At this point, there is one unoccupied track (i.e., t_2) and there is one connection (i.e., c_2) in pool P. Hence, the number of unoccupied tracks equals the number of connections in P, and the algorithm would assign connection c_2 to the unoccupied track t_2. Next, the algorithm assigns c_4 to track t_1.
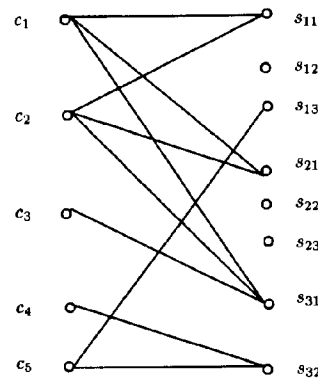
Theorem 4: The above-mentioned algorithm determines a routing, if one exists, for the case where every track has at most two segments.

Proof: We shall provide an outline of the proof. Since every track has at most two segments, it follows that if a connection cannot be assigned to a single segment, then it has to occupy a whole track. Note also that the above-mentioned algorithm follows the greedy algorithm developed for 1-segment routing, and if a connection cannot be assigned to a single segment (by following the 1-segment routing algorithm) only then it is appended to the pool P.

The basic idea of the proof relies on the following observations. Since the algorithm developed for 1-segment routing was proved to be optimal, the connections in the pool P represent each of those connections that require a whole track. Moreover, these connections (i.e., which require whole tracks) are not assigned until a) all other connections are assigned to single segments and there are enough unoccupied tracks left to accommodate the connections in P; or b) during execution there are exactly as many unoccupied tracks as the number of connections in P (i.e., since the connections in P must require whole tracks, these unoccupied tracks must be assigned to these connections). Thus the routing algorithm maximizes the connections that can be assigned to single segments and minimizes the connections that have to be assigned whole tracks.

A more rigorous proof, similar to the one for Theorem 3, can also be developed. More precisely, we can show that given any routing, one can always modify it such that the modified routing will be the same as one that the above-mentioned algorithm would generate. The details, however, get more involved; moreover, one may lose the intuitive appeal of the above explanations.   □

B. A General Algorithm for Determining Routing

Although the problem of determining a routing for a given segmented channel and a set of connections is in general NP-complete, we describe below an algorithm that finds a routing in time linear in M (the number of connec-
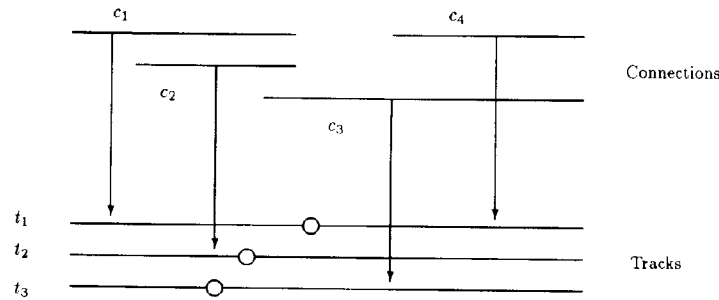
Fig. 8. An example of a routing with at most two segments per track.
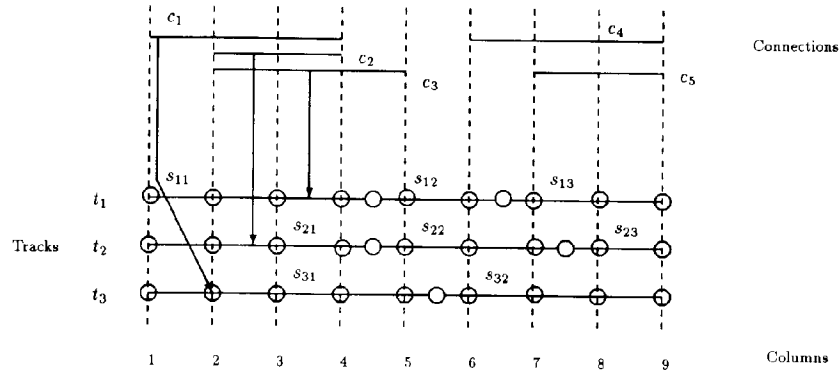
Fig. 9. A frontier for the example of Fig. 3. Connections c_1, c_2, and c_3 are assigned to segments s_{11}, s_{21}, and $\{s_{31}, s_{12}\}$, respectively. The frontier is $x = [7, 6, 6]$.

tions) when T (the number of tracks) is fixed. This is of interest since T is often substantially less than M. The algorithm may also be quite efficient when there are many tracks, but they are segmented in a limited number of ways (see Theorem 7). The algorithm first constructs a data structure called an *assignment graph* and then reads a valid routing from it. The same algorithm applies to both Problems 1 and 2, though with different time and memory bounds. It can also be extended to Problem 3.

Frontiers and the assignment graph: Given a valid routing for connections c_1 through c_i, it is possible to define a *frontier* which constitutes sufficient information to determine how the routing of $c_1 \cdots c_i$ may be extended to include an assignment of c_{i+1} to a track such that no segment occupied by any of c_1 through c_i will also be occupied by c_{i+1}. Fig. 9 shows an example of a frontier. It will be apparent that c_{i+1} may be assigned to any track t in which the frontier has not advanced past the left end of c_{i+1}. For example, in Fig. 9 connection c_4 can be assigned to track t_2 but not to track t_1.

More precisely, given a valid routing of $c_1, \cdots, c_i$, $1 \le i < M$, define the frontier x to be a T-tuple $(x[1], x[2], \cdots, x[T])$ where $x[j]$ is the leftmost unoccupied column in track t_j at or to the right of column $left(c_{i+1})$. (A column in track t_j is considered unoccupied if the segment present in the column is not occupied.) The frontier is thus a function $x = F_i(t_{c_1}, \cdots, t_{c_i})$ of the tracks t_{c_1},

$\cdots, t_{c_i}$ to which $c_1 \cdots c_i$ are, respectively, assigned. For $i = 0$, let $x = F_0$, where $F_0[t] = left(c_1)$ for all t. For $i = M$, let $x = F_M$, where $F_M[t] = N + 1$ for all t.

Next, we describe a graph called the assignment graph, which is used to keep track of partial routings and the corresponding frontiers. A node at level i, $1 \le i < M$, of the assignment graph corresponds to a frontier resulting from some valid routing of c_1–c_i; see Fig. 10 for an illustration of the structure of an assignment graph. Level 0 of the graph contains the root node, which corresponds to F_0. If a complete valid routing for $c_1, \cdots, c_M$ exists, then level M of the graph contains a single node corresponding to F_M. Otherwise, level M is empty.

The assignment graph is constructed inductively. Given level $i \ge 0$ of the graph, construct level $i + 1$ as follows. (For convenience, we identify the node by the corresponding frontier.)

For each node x_i in level i {
  For each track t_j, $1 \le j \le T$ {
    If $x_i[j] = left(c_{i+1})$ {
      /* c_{i+1} can be assigned to track t_j.*/
      Let x_{i+1} be the new frontier after c_{i+1} is assigned to track t.
      If x_{i+1} is not yet in level $i + 1$ {
        Add node x_{i+1} to level $i + 1$.
        Add an edge from node x_i to node x_{i+1}. Label it with t_j.

Edges represent valid assignment of
connection c_{i+1} to some track t_j

Frontiers after assignment of   Frontiers after assignment of
connections $c_1, \cdots, c_i$ connections $c_1, \cdots, c_{i+1}$
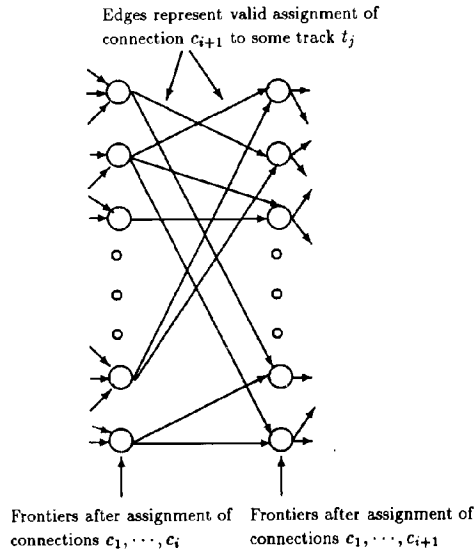
Fig. 10. An illustration of the structure of an assignment graph.

```
        }
      }
  Else {
      /* x_i[j] > left(c_{i+1}) so c_{i+1} cannot be assigned
      to track t_j. */
      Continue to next track t_{j+1}.
      }
    }
  }
```

If there are no nodes added at level $i + 1$, then there is not valid assignment of c_1–c_{i+1}.

Searching for the node x_{i+1} in level $i + 1$ can be done in $O(T)$ time, using a hash table. Insertion of a new node in the table likewise requires time $O(T)$.

If there is a maximum of L nodes at each level, then construction of the entire assignment graph requires time $O(MLT^2)$. Once the assignment graph has been constructed, a valid routing may be found by tracing a path from the node at level M back to the root, reading the track assignment from the edge labels. (If there is no node at level M, then no complete valid assignment exists.) This takes only $O(M)$ time, so the overall time for the algorithm is $O(MLT^2)$. The memory required to store the assignment graph is $O(MLT)$.

A minor change allows us to solve the optimization problem as well. Each edge is labeled with the weight $w(c, t_c)$ of the corresponding assignment. Each node is labeled with the weight of its parent node plus the weight of the incoming edge. The algorithm is modified as follows. If a search in level $i + 1$ finds that the new node x_{i+1} already exists, we examine its weight relative to the weight of node x_i plus $w(c_{i+1}, t_{c_{i+1}})$. If the latter is smaller, we replace the edge entering x_{i+1} with one from x_i and update the weights accordingly. Thus the path

traced back from the node at level M will correspond to a minimal weight routing. The order of growth of the algorithm's time remains the same, as does that of its memory.

Analysis for unlimited segment routing: The following theorem shows that for unlimited segment routing, $L \leq 2T!$, so that the time to construct the assignment graph and find an optimal routing is $O(MT^2 T!)$ and the memory required is $O(MTT!)$.

Theorem 5: For unlimited segment routing, the number of distinct frontiers that may occur for some valid assignment of c_1–c_i is at most $2T!$.

Proof: Let $l = left(c_{i+1})$. Let d be the number of connections among c_1 through c_i that are present in column l. Since the assignment of c_1 through c_i determining the frontier must be a valid one, we know that $d \leq T$. The d connections can be assigned to d of the T tracks in $T!/(T - d)!$ ways. Once we have assigned a connection to a track t_j, the value of $x[j]$ in that track is determined. For each of the remaining $(T - d)$ tracks, there are only two alternatives.

1) the track t_j may be unoccupied in column l, in which case, $x[j] = l$.

2) the track t_j may be occupied in column l by some connection c with $right(c) < l$, up to the first switch to the right of column l. In this case, $x[j]$ is the column just to the right of this switch, regardless of which such connection c is involved.

Thus the number of possible frontiers is at most $2^{(T-d)} T!/(T - d)! \leq 2T!$. ☐

Analysis for K-segment routing: The following theorem shows that for K-segment routing, $L \leq (K + 1)^T$, so that the time to construct the assignment graph and find an optimal routing is $O(MT^2(K + 1)^T)$ and the memory required is $O(MT(K + 1)^T)$.

Theorem 6: For K-segment routing, the number of distinct frontiers that may occur for some valid routing of c_1–c_i is at most $(K + 1)^T$.

Proof: Let $l = left(c_{i+1})$ and consider track t_i. Since the connections are sorted by increasing left edge, at most one connection from among $c_1 \cdots c_i$ may occupy track t_i in columns at or to the right of column l. Such a connection may occupy track t rightward through the segment appearing in column l, or through that segment plus the next one, or possibly as far as the Kth segment at or to the right of column l. Of course it is also possible that no connection from among c_1–c_i occupies the segment in column l of track t_i. Thus there are only $K + 1$ possible locations for the frontier $x[i]$ in track t_i, and at most $(K + 1)^T$ possible values for the frontier x overall. ☐

Case of many tracks of a few types: Suppose the T tracks fall into two types, with all tracks of each type segmented identically. Then two frontiers that differ only by a permutation among the tracks of each type may be considered equivalent for our purposes in that one frontier can be a precursor of a complete routing if and only if the other can. Thus we can restrict consideration to only one

of each set of equivalent frontiers and strengthen the result of Theorem 6 as follows.

Theorem 7: Suppose there are T_1 tracks segmented in one way and $T_2 = T - T_2$ segmented another way. The number of distinct frontiers x that may occur for some valid K-segment routing of c_1–c_i, and that satisfy $x[i] \leq x[j]$ for all $i < j$ with tracks t_i and t_j of the same type, is $O((T_1 T_2)^K)$.

Proof: As in Theorem 6, there are at most $K + 1$ possible values for $x[i]$. Due to the inequality restriction (which eliminates all but one member of each set of equivalent frontiers), the number of possible frontiers is at most

$$\binom{T_1 + K}{T_1} \times \binom{T_2 + K}{T_2}$$

which for large T_1 and T_2 is $O((T_1 T_2)^K)$.

It follows that a K-segment routing may be found in time $O(M(T_1 T_2)^K T^2)$, and memory $O(M(T_1 T_2)^K T)$.

The result of Theorem 7 may easily be generalized to the case of l types of tracks, in which case the time is $O(M(\Pi_1^l T_i^K))$, and the memory is $O(M(\Pi_1^l T_i^K)T)$.

C. A Linear Programming Approach

Problems 1 and 2 can be reduced to $0 - 1$ linear programming (LP) problems via a straightforward reduction procedure. The $0 - 1$ LP is in general NP-complete. For our purposes, however, such a reduction is interesting because our simulations showed (see [12]) that for almost all cases the corresponding $0 - 1$ LP problems could be solved by viewing them as *ordinary LP problems* for which efficient algorithms are known. In particular, our simulation results indicated that whenever a randomly generated instance of Problem 1 had a feasible solution, one could always find $0 - 1$ feasible solutions for the corresponding integer LP problem by solving it as an ordinary LP. The simulations were carried out for fairly large-sized instances, e.g., $M = 60$ and $T = 25$.

We now describe briefly the reduction procedure for Problem 1. The corresponding reduction for Problem 2 follows after minor modifications. Let us define binary variables x_{ij}, for $1 \leq i \leq M$, and $1 \leq j \leq T$ as follows: if $x_{ij} = 1$, then connection c_i is assigned to track t_j, else if $x_{ij} = 0$, then connection c_i is not assigned to track t_j. Since in a routing each connection is assigned to at most one track, one has the following constraints:

$$\sum_{j=1}^{l} x_{ij} \leq 1, \quad \forall \, 1 \leq i \leq M.$$

One also has to make sure that in any routing two connections assigned to the same track must not share a segment. Consider a track t_j; one can then easily determine sets of connections $P_{j1}, \cdots, P_{jl_j}$ (not necessarily disjoint) such that at most one from each set can be assigned

to the track t_j. Hence for each such set P_{jk}, one must satisfy

$$\sum_{c_i \in P_{jk}} x_{ij} \leq 1.$$

Finally, one must make sure that all the connections are routed; this can be ensured by maximizing the following objective function:

$$\sum_{i=1}^{M} \sum_{j=1}^{l} x_{ij}.$$

One can now easily verify the following facts about the above $0 - 1$ LP.

1) The objective function achieves the maximum value of M if there is a solution to Problem 1. This is because in a feasible routing each c_i is assigned to some track (thus there exists only one j such that $x_{ij} = 1$ for every i) and the constraints are never violated.

2) If the objective function achieves the value of M, then there is a solution to Problem 1. This follows directly from our construction of the $0 - 1$ LP.

Note that one can derive a $0 - 1$ LP for solving Problem 2 if one assigns $x_{ij} = 0$ whenever a connection c_i cannot be assigned to track t_j because it would require more than K segments.

V. An Algorithm for Determining Generalized Routing

We present here an algorithm for solving Problem 4. The algorithm has a time complexity of $O(T^{T+3}M)$, and is derived by modifying the construction of assignment graphs introduced in the last section. Thus, for a constant number of tracks the generalized segmented routing problem can be solved in time linear in M (the number of connections). We should note here that efficient algorithms for various special cases of the generalized segmented routing problem and results on their computational complexity remain as open problems.

Given an instance of the generalized segmented routing problem with a set of connections $\mathcal{C}$ (with M connections) and a set of tracks $\mathcal{T}$, the first step in our algorithm involves defining a new set of connections $\mathcal{C}'$ as follows:
For every connection $c_i = (left(c_i), right(c_i))$ in $\mathcal{C}$ we will define $p = right(c_i) - left(c_i) + 1$ connections in $\mathcal{C}'$, each spanning a single column. That is, the corresponding p connections in $\mathcal{C}'$ are: $(left(c_i), left(c_i))$, $(left(c_i) + 1, left(c_i) - 1)$, $\cdots$, $(right(c_i), right(c_i))$. Note that every connection in $\mathcal{C}'$ spans only a single column and the total number of connections in $\mathcal{C}'$ is at most MN (because each connection in $\mathcal{C}$ can generate at most N connections in $\mathcal{C}'$).

Proposition 11: A generalized segmented routing (as defined in Definition 2) for a set of connections $\mathcal{C}$ and a set of tracks $\mathcal{T}$ can be determined by finding a usual segmented routing (as defined in Definition 1) for the set of

ROYCHOWDHURY *et al:* SEGMENTED CHANNEL ROUTING

connections $\mathcal{C}'$ and the set of tracks $\mathfrak{I}$ if two connections in $\mathcal{C}'$ that are derived from the same parent connection in $\mathcal{C}$ are allowed to share the same segment.

Proof: The proof follows directly from the above construction. □

In order to determine a routing where certain connections from $\mathcal{C}'$ are allowed to occupy the same segment, we shall modify the construction of the assignment graph that was introduced in the previous section. As before, given a valid routing for connections $c_1, \cdots, c_i$ (in $\mathcal{C}'$), it is possible to define a *frontier* which constitutes sufficient information to determine how the routing of $c_1, \cdots, c_i$ may be extended to include an assignment of c_{i+1}. However, since connections in $\mathcal{C}'$ that originate from the same connection in $\mathcal{C}$ are allowed to occupy the same segment, it is not sufficient to just keep track of the occupancy of the segments (this is what is done in Section IV-B). In other words, one has to keep the additional information that would indicate whether connection c_{i+1} can be assigned to an already occupied segment. This can be done by storing the information that if a segment at a frontier is occupied, then which connection from $\mathcal{C}$ occupies it; *a segment will be said to be occupied by a connection c_j in $\mathcal{C}$ if a connection in $\mathcal{C}'$ that is derived from c_j occupies the given segment.*

More precisely, given a valid routing of $c_1, \cdots, c_i$ in $\mathcal{C}'$, define the frontier x to be a T-tuple $(x[1], x[2], \cdots, x[T])$ where $x[j] = (x_1[j], x_2[j])$. $x_1[j]$ is defined as before, i.e., it is the leftmost unoccupied column in track t_j at or to the right of column $left(c_{i+1})$. (Recall that a column in track t_j is considered unoccupied if the segment present in the column is not occupied.) On the other hand, $x_2[j]$ indicates that if the column $left(c_{i+1})$ is occupied (i.e., $x_1[j] > left(c_{i+1})$) then which connection in $\mathcal{C}$ occupies it. $x_2[j]$ can take two types of values:

1) $1 \leq x_2[j] \leq M$: In this case the value of $x_2[j]$ gives the connection in $\mathcal{C}$ that occupies the segment of the frontier (i.e., the segment present in column $left(c_{i+1})$) in track t_j. Thus, if c_{i+1} is derived from connection $c_{x_2[j]}$ in $\mathcal{C}$ then c_{i+1} can be assigned to track t_j *irrespective* of the value of $x_1[j]$.

2) $x_2[j] = \phi$: This case would imply that whether c_{i+1} can be assigned to track t_j is determined *only* by the value of $x_1[j]$. Thus if $x_2[j] = \phi$ then c_{i+1} can be assigned to track t_j only if $x_1[j] = left(c_{i+1})$.

Thus a frontier is a function $x = F_i(t_{c_1}, \cdots, t_{c_i})$ of the tracks $t_{c_1}, \cdots, t_{c_i}$ to which $c_1, \cdots, c_i$ are respectively assigned. For $i = 0$, let $x = F_0$, where $F_0[t] = (left(c_1), \phi)$ for all t. For $i = M$, let $x = F_M$, where $F_M[t] = (N + 1, \phi)$ for all t.

As in Section IV-B, an assignment graph can now be used to keep track of the partial routings and the corresponding frontiers. A node at level i, $1 \leq i < M$, of the assignment graph corresponds to a frontier resulting from some valid routing of c_1 through c_i. Level 0 of the graph contains the root node, which corresponds to F_0. If a complete valid routing for $c_1, \cdots, c_M$ exists, then level M

of the graph contains a single node corresponding to F_M. Otherwise, level M is empty.

As in Section IV-B, the assignment graph is constructed inductively, and a modified algorithm for its construction can be stated as described below.

Given level $i \geq 0$ of the graph, construct level $i + 1$ as follows. (For convenience, we identify the node by the corresponding frontier.)

For each node x^i in level i {
    For each track t_j, $1 \leq j \leq T$ {
        If $(x_1^i[j] = left(c_{i+1}))$ or $(c_{i+1}$ is derived from $c_{x_2^i[j]}$ in $\mathcal{C}$) {
        /* c_{i+1} can be assigned to track t_j. */
        Let x^{i+1} be the new frontier after c_{i+1} is assigned to track t.
        If x^{i+1} is not yet in level $i + 1$ {
            Add note x^{i+1} to level $i + 1$.
            Add an edge from node x^i to node x^{i+1}. Label it with t_j.
        }
        }
        Else {
        /* c_{i+1} cannot be assigned to track t. */
        Continue to next track t_{j+1}.
        }
    }
}

If there are no nodes added at level $i + 1$, then there is no valid assignment of c_1 through c_{i+1}.

Theorem 8: There is an $O(T^{T-3}M)$ time algorithm for solving Problem 4.

Proof: Recall from Section IV-B that if L is the maximum number of nodes at any level of the assignment graph then the time complexity of the above algorithm is $O(MLT^2)$. We will show here that $L = O(T^{T+1})$.

Let $l = left(c_{i+1})$, and consider a frontier $x = (x[1], x[2], \cdots, x[T])$ after a valid routing for connections $c_1, \cdots, c_i$. Recall that *every connection in $\mathcal{C}'$ spans a single column.* Hence in any track t_j, *only* the segment present in column l can either be occupied or unoccupied by connections $c_1, \cdots, c_i$; in other words, any segment to the right of the segment present in column l cannot be occupied by $c_1, \cdots, c_i$. Hence, $x_1[j]$ can assume only *two values*, namely, $x_1[j] = l$ or $x_1[j]$ equals the column where the segment present in column l ends.

Let us next consider the possible values of $x_2[j]$. We claim that, given a frontier, in order to correctly assign connection c_{i+1}, it is sufficient to know whether a segment at column l is occupied by connections (in $\mathcal{C}$) present *only* in column $l - 1$. This claim follows easily from the geometry of our segmented routing problem. In other words, connections in $\mathcal{C}$ were broken up into disjoint but contiguous units to generate connections of $\mathcal{C}'$. Hence, if c_{i+1} shares a segment with another connection then that connection must be derived from a connection in $\mathcal{C}$ that occupies column $l - 1$. Thus, in the frontier if a segment

in t_j (spanning column l) is occupied by a connection present in column $l - 1$, then the value of this connection is stored in $x_2[j]$; otherwise, $x_2[j]$ is set to ϕ.

Let d be the connections present in column $l - 1$, then $x_2[j]$ can take at most $d - 1$ values. We have already shown that $x_1[j]$ can take at most two values. Hence, the maximum number of distinct frontiers possible is $2^T T^{d-1}$. However, since connections present at the same column has to be assigned to different tracks, $d \leq T$. Hence, $L \leq 2^T T^{T-1}$; in other words, $L = O(T^{T-1})$.

The above algorithm could be easily modified to solve the following restricted versions of the generalized segmented routing problem.

1) Each connection can switch tracks only at prespecified columns.

2) If a connection c switches from track t_1 to track t_2 at column l then the segments in the two different tracks (to which parts of c are assigned to) must be include l. It is easy to see that the algorithm described in this section might assign connections such that the segments in t_1 and t_2 to which parts of c, are assigned are separated by one column; this might not be desirable in certain hardware models.

We will not go into the details of the modifications, however, the general idea is as follows: the assignment graph as described above enumerates all possible routings, and restricted routings can be easily obtained by disallowing assignments that violate the premises.

VI. Concluding Remarks

We have introduced novel problems concerning the design and routing for segmented channels. We also have presented the first known theoretical results on the algorithm design, and combinatorial complexity of the routing problem for segmented channels. In particular, we showed that 1) the problem of determining a routing for a given segmented channels and connections is in general NP-complete; 2) efficient polynomial time algorithms can be designed for several special cases; and 3) efficient algorithms can be designed for some cases of a generalized segmented routing problem, where connections can occupy segments in different tracks.

There are several open issues in this new area of routing. For example, although we have developed efficient algorithms for many special cases of the routing problem (as listed in Section II), several other interesting cases are yet to be solved; following are some relevant ones: 1) channel length (N) is bounded, 2) connection lengths are bounded, and 3) connections are nonoverlapping. Also, efficient algorithms for the generalized routing problems are not known.

The routing scheme using segmented channels may also be considered as a model for a communication network in a multiprocessor architecture. The logic modules in Fig. 1 can be replaced by processing elements (PE's); the segmented routing network can then be used for dynamically reconfiguring interconnections among the PE's (by programming the appropriate switches as described for the FPGA's). In [8] a preliminary network model that uses specially segmented channels (referred to as express channels) has already been proposed. Tradeoffs similar to those discussed in Section I also appear to hold for such multiprocessor communication networks; however, this area needs further investigation.

Appendix

We showed in Section III that the unlimited segment routing problem is strongly NP-complete. We shall now use the instance of the unlimited segment routing problem that was used in proving Theorem 1 and reduce it to a 2-segment routing problem, thereby showing that the latter problem is also strongly NP-complete.

Let us briefly recall the construction of the unlimited segment routing problem Q.

Given: Integers $x_1 < x_2 < \cdots x_n$, $y_1 < y_2 < \cdots < y_n$, and $z_1 < z_2 < \cdots < z_n$, such that 1) $\Sigma_{i \in S}(x_i + y_i) = \Sigma_{i \in S} z_i$ and 2) $x_{i+1} - x_i \geq n$ for every $1 \leq i \leq n - 1$ and $x_1 + y_1 \geq x_n + n$. For this section, without loss of generality, we shall further assume that $z_1 \geq x_n + n$.

The set of connections, C, is then defined as follows.

1) For each x_i we defined a connection a_i such that left$(a_i) = 4$, right$(a_i) = x_i + 3$.

2) For each y_k we defined n connections $b_{k1}, \cdots, b_{kn}$ (one for each x_j) such that left$(b_{kj}) = x_j - 4 + (n - k)$ and right$(b_{kj}) = (y_k + x_j) + 4$.

3) n connections $d_1, \cdots, d_n$ are defined with left$(d_i) = 1$, and right$(d_i) = 3$.

4) $n^2 - n$ connections $e_1, \cdots, e_{n^2-n}$ are defined with left$(e_i) = 1$, and right$(e_i) = 5$.

5) n^2 connections $f_1, \cdots, f_{n^2}$ are defined with left$(f_i) = x_n + y_n + 5$ and right$(f_i) = x_n + y_n - 7$.

The number of columns is set to $N = x_n + y_n + 7$.

The set $\mathfrak{I}$ of n^2 tracks is then defined as follows:

1) For the first n tracks $t_1, \cdots, t_n$ each track t_i begins with a segment $(1, 3)$ followed by unit length segments that span the region from column 4 to column $z_i + 4$, followed by a single segment of the form $(z_i + 5, N)$.

2) The rest of the $n^2 - n$ tracks are best described by dividing them into n blocks, each consisting of $n - 1$ tracks. Each such track comprises three segments.

The first block of $n - 1$ tracks, i.e., tracks $t_{n+1}, t_{n+2}, \cdots, t_{2n-1}$, are constructed using the definitions of the connections b_{1j}, $1 \leq j \leq n$. The segments in each track t_{n+j}, $1 \leq j \leq n - 1$, are defined as $(1, \text{left}(b_{1j}) - 1)$, $(\text{left}(b_{1j}), \text{right}(b_{1(j+1)}))$, and $(\text{right}(b_{1(j+1)}) + 1, N)$. That is, the middle segment in the track t_{n+j} is defined such that the connections b_{1j} or $b_{1(j-1)}$ can be assigned to it.

The ith block of $n - 1$ tracks (i.e., tracks $t_{n+(i-1)(n-1)+1}, \cdots, t_{n+i(n-1)}$) is constructed using the definitions of the connections b_{ij}, $1 \leq j \leq n$. The segments in the track $t_{(n+(i-1)(n-1))+j}$ (i.e., the jth track in

the ith block) are (1, left(b_{ij}) − 1), (left(b_{ij}), right($b_{i(j+1)}$)), and (right($b_{i(j+1)}$) + 1, N).

Given the above instance $\mathcal{Q}$ of unlimited segment routing, one can generate an instance $\mathcal{Q}_2$ of a 2-segment routing problem as described below.

The number of columns is set to the same value as in $\mathcal{Q}$, i.e., $N = x_n + y_n + 7$. The set of connections in the 2-segment problem $\mathcal{Q}_2$ is defined as follows:

1) The connections a_i, $1 \le i \le n$, e_i, $1 \le i \le n^2 - n$, and b_{ij}, $1 \le i, j \le n$ are defined as in the unlimited segment routing problem $\mathcal{Q}$.

The connections f_i are again defined as in $\mathcal{Q}$, except that there are now $2n^2 - n$ of them, i.e., $1 \le i \le 2n^2 - n$.

The connections d_i, defined in problem instance $\mathcal{Q}$, are omitted in $\mathcal{Q}_2$.

2) $n^2 - n$ new connections g_{ij}, where $1 \le i \le n$, and $1 \le j \le (n-1)$, are added such that left(g_{ij}) = 4 and right(g_{ij}) = $z_i + 4$. Note that for a fixed value of i, all the $n-1$ connections, g_{ij}, where $1 \le j \le (n-1)$, are identical and have the same left and right end points.

The set of tracks (comprising $2n^2 - n$ tracks) is defined as follows:

1) Each track t_i, $1 \le i \le n$ in the construction of the unlimited segment routing problem $\mathcal{Q}$, is replaced by a set of n tracks that we label as t_{ij}, $1 \le j \le n$. Each such track comprises five segments. Let us first describe the five segments in the tracks, t_{1j}, $1 \le j \le n$: they are (1, 2), (3, 3), (4, right(a_j)), (right(a_j) + 1, $z_1 + 4$), and ($z_1 + 5$, N).

In general, for any i ($1 \le i \le n$), the segments in the tracks, t_{ij}, $1 \le j \le n$, are defined as follows: (1, 2), (3, 3), (4, right(a_j)), (right(a_j) + 1, $z_i + 4$), and ($z_i + 5$, N).

2) The last $n^2 - n$ tracks, i.e., $t_{(n+1)} \cdots , t_{n^2}$, in the unlimited segment routing is kept the same in the 2-segment routing problem, $\mathcal{Q}_2$.

Before we proceed, let us review the properties that routings must satisfy in the unlimited segment problem, which we proved in Section III.

1) In every track, t_i, $1 \le i \le n$, a connection a_i can only be assigned to the same track with some b_{ki} (see Lemma 2).

2) In every track, t_i, $1 \le i \le n$, the length occupied by connections $a_{\alpha(i)}$ and $b_{\alpha(i)\beta(i)}$ that are assigned to it is z_i. Note that the length occupied is defined as right($b_{\alpha(i)\beta(i)}$) − left($a_{\alpha(i)}$); (see Lemma 2).

Proposition 12: In any routing of $\mathcal{Q}_2$:

a) the connections e_i, $1 \le i \le n^2 - n$, are assigned to tracks $t_{n+1}, \cdots , t_{n^2}$, i.e., the last $n^2 - n$ tracks.

b) the connections f_i, $1 \le i \le 2n^2 - n$, occupy the last segment in every track.

c) the connections a_i, $1 \le i \le n$, are assigned to tracks t_{ij}, $1 \le i, j \le n$.

d) the connections g_{ij}, $1 \le i \le n$, $1 \le j \le n - 1$ cannot be assigned to tracks $t_{n+1}, \cdots , t_{n^2}$.

e) only n connections from b_{ij}, $1 \le i, j \le n$, can be assigned to tracks t_{ij}, $1 \le i, j \le n$, and the rest of the n^2

− n connections are assigned to tracks $t_{n+1}, \cdots , t_{n^2}$. Thus, each track among $t_{n+1}, \cdots , t_{n^2}$, has one connection from b_{ij}, $1 \le i, j \le n$, assigned to it.

Proof: In Proposition 12, a) follows directly from the construction: if an e_i is assigned to any track among t_{ij}, $1 \le i, j \le n$, then it would occupy three segments: this is not permitted in 2-segment routings. Hence, the first segment in each of the last $n^2 - n$ tracks must be occupied by an e_i connection.

b) follows again from the construction: all the connections f_i are overlapping and there are as many of these connections as the total number of tracks. Hence, they occupy the last segment in each track.

c) follows from a): the connections a_i and e_j overlap for every i and j; hence, none of the a_i connections can be assigned to tracks $t_{n+1}, \cdots , t_{n^2}$.

d) again follows from a): every connection g_{ij} overlaps with every connection e_k. Since the connections e_k are assigned to the last $n^2 - n$ tracks, the connections g_{ij} must be assigned to the tracks t_{ij}, $1 \le i, j \le n$.

e) follows from d): every b_{ij} overlaps with every g_{ij} (because by assumption $z_1 \ge x_n + n$); since all the g_{ij} ($n^2 - n$ of them) are assigned to the top t_{ij}, $1 \le i, j \le n$ tracks, there are only n tracks left that connections b_{ij} can be assigned to. This also implies that each track among t_{n+1}, $\cdots , t_{n^2}$, has one connection from b_{ij}, $1 \le i, j \le n$ assigned to it.   □

Proposition 13: The total length required by the connections among a_i and b_{ij} that are assigned to n tracks among t_{ij}, $1 \le i, j \le n$, is $\ge \Sigma_{i=1}^{n} z_i$.

Proof: Since the connections a_i, b_{ij} and the tracks $t_{n+1}, \cdots , t_{n^2}$ are defined identically in problems $\mathcal{Q}$ and $\mathcal{Q}_2$, Propositions 4, 5, 6, and 7 (proved in Section III) are also true for $\mathcal{Q}_2$.

If R_2 is any routing for $\mathcal{Q}_2$, then we can define a quantity l_i (similar to m_i defined in Section III) as follows:

$$l_i = |\{b_{ij}: 1 \le j \le n, \text{ and } b_{ij} \text{ is assigned to some track } t_{kl}, 1 \le k, l \le n, \text{ in } R_2\}|.$$

In other words, l_i is the number of connections from the set $\{b_{i1}, b_{i2}, \cdots , b_{in}\}$ that are assigned to tracks t_{kl}. We can now exactly follow the arguments of Propositions 8–10 and show that a) $\Sigma_{i=1}^{k} l_i \le k$ for all $1 \le k \le n$, and $\Sigma_{i=1}^{n} l_i = n$; b) the total length occupied by the connections a_i and b_{ij} in the tracks t_{ij}, $1 \le i, j \le n$, is $> \Sigma_1^n x_i + \Sigma_1^n l_k y_k$, and c) finally, (using Proposition 9 and the arguments in Proposition 10)

$$\sum_1^n x_i + \sum_1^n l_k y_k \ge \sum_1^n x_i + \sum_1^n y_k = \sum_{i=1}^{n} z_i$$

where equality is met if and only if $l_k = 1$ for all k.   □

The next proposition shows that among n tracks t_{ij}, $1 \le j \le n$ (i is fixed), there is exactly one track that can be occupied by connections a_k and b_{lm}; the rest are occupied by $n - 1$ connections g_{ij}, $1 \le j \le n - 1$.

Proposition 14: In any routing of Q_2, for any fixed i, the $(n - 1)$ connections g_{ij}, $1 \leq j \leq (n - 1)$, can only be assigned to tracks in the set of t_{ik}, $1 \leq k \leq n$.

Proof: Let us define h_i to be the number of tracks among t_{ij}, $1 \leq j \leq n$, that are unoccupied by the connections g_{kl}, $1 \leq k \leq n$, $1 \leq l \leq (n - 1)$. These are also the tracks available for connections a_i and b_{ij}: hence, $\Sigma_{i=1}^{n} h_i = n$. Moreover, the total space available in these tracks (for connections a_i and b_{ij}) is $\Sigma_{i=1}^{n} h_i z_i$ (because the total length provided by any track t_{ij} for connections a_i and b_{ij} is z_i).

Next, let us observe that any connections g_{ij} can never be assigned to a track t_{kl} where $k < i$. This is because the connection g_{ij} is defined as $(4, z_i + 4)$, and one can easily verify that if it is assigned to track t_{kl}, $k < i$, then it would occupy three segments which is not permitted in a 2-segment routing. Using the above property we can show that

$$\sum_{j=1}^{i} h_{(n+1)-j} \leq i, \qquad \forall 1 \leq i \leq n.$$

For example, for $i = 1$ the above relationship follows easily: none of the connections g_{nj} can be assigned to tracks t_{kl} where $k \leq n$, hence, all of them have to be assigned to tracks t_{nl} (see Proposition 12); thus, the maximum number of tracks unoccupied by g_{nj}, $1 \leq j \leq n - 1$, among t_{nk}, $1 \leq k \leq n$, is at most 1, or equivalently, $h_n \leq 1$. For other values of i, the above relationship can be showed by induction.

Now, we know that $z_n > z_{n-1} > \cdots > z_1$; using this property and the fact that 1) $\Sigma_{j=1}^{i} h_{(n-1)-j} \leq i$, $\forall 1 \leq i \leq n$; b) $\Sigma_{j}^{n} h_{(n-1)-j} = n$, we can easily derive (applying arguments analogous to those in Propositions 9 and 10) that $\Sigma_{i=1}^{n} z_i h_i \leq \Sigma_{i=1}^{n} z_i$ and the equality results if and only if $h_i = 1$ for all $1 \leq i \leq n$.

Thus if $h_i \neq 1$ for all i then it leads to a contradiction with Proposition 13.

Note that we already showed that all the connections g_{nj}, $1 \leq j \leq n - 1$, have to be assigned to tracks t_{nk}, $1 \leq k \leq n$. Now, $h_n = 1$, hence, for connections $g_{(n-1)j}$, $1 \leq j \leq n - 1$, the only available tracks are $t_{(n-1)k}$, $1 \leq k \leq n$. Since $h_{n-1} = 1$, the same arguments can be continued to show that the $(n - 1)$ connections g_{ij}, $1 \leq j \leq (n - 1)$, can only be assigned to tracks in the set of t_{ij}, $1 \leq k \leq n$. □

Theorem 2: Determining a solution to Problem 2 is strongly NP-complete even when $K = 2$.

Proof: First let us show that if there is a solution to the unlimited segment routing problem for Q, then there is a solution to the 2-segment routing problem for Q_2. The assignments for Q_2 are as follows:

1) The connections e_i, $1 \leq i \leq n^2 - n$ are assigned to tracks t_{n-1}, $\cdots$, t_{n^2}. Since, the last $n^2 - n$ tracks are identical in both instances and e_i gets assigned to single segments in every track, this is a valid step.

The connections f_i, $1 \leq i \leq 2n^2$ are assigned the last segment in every track.

2) Since, the last $n^2 - n$ tracks are identical in both Q and Q_2 (and so are the connections b_{ij}), the connections b_{ij} assigned to the these tracks in Q are also assigned to the same tracks in Q_2. This leaves n connections among b_{ij}, $1 \leq i, j \leq n$ to be routed (precisely those which are assigned to tracks $t_1, \cdots, t_n$ in the routing for Q).

3) Next consider the connections a_i and b_{ij} that are assigned to the first n tracks $t_1, \cdots, t_n$ in Q. First consider, t_1, and let the connections assigned to it be $a_{\alpha(1)}$ and $b_{\beta(1)\alpha(1)}$ (recall from Section III that a connection a_i can only be assigned to the same track with some b_{ki}). Now consider the track $t_{1\alpha(1)}$ in Q_2, it has a segment $(4,$ right $(a_{\alpha(1)}))$ to which the connection $a_{\alpha(1)}$ can be assigned and a segment (right $(a_{\alpha(1)}) + 1, z_i + 4)$ to which the connection $b_{\beta(1)\alpha(1)}$ can be assigned. Next, the $n - 1$ connections g_{1j}, $1 \leq j \leq n - 1$ can be assigned to the $n - 1$ tracks among t_{1j}, $1 \leq j \leq n$ that are not occupied by the connections $a_{\alpha(1)}$ and $b_{\beta(1)\alpha(1)}$.

This procedure can be continued, i.e., consider track t_i in Q and let $a_{\alpha(i)}$ and $b_{\beta(i)\alpha(i)}$ be the connections assigned to it. Then for a routing of Q_2, assign the connections $a_{\alpha(i)}$ and $b_{\beta(i)\alpha(i)}$ to track $t_{i\alpha(i)}$. To the rest of the $(n - 1)$ tracks among t_{ij} assign the connections g_{ij}, $1 \leq j \leq n - 1$.

One can easily verify that after following the above three steps, all the connections of Q_2 are appropriately routed.

We now state how to get a routing for Q given a routing for Q_2.

1) Assign d_i, $1 \leq i \leq n$, f_i, $1 \leq i \leq n^2$ and e_i, $1 \leq i \leq n^2 - n$ according to Proposition 1.

2) The $n^2 - n$ connections among b_{ij}, $1 \leq i, j \leq n$ that are assigned to tracks t_{n-1}, $\cdots$, t_{n^2} in Q_2 are assigned to the identical tracks in Q.

3) After the above steps, one is left with the connections a_i, $1 \leq i \leq n$, and n connections among b_{ij} (precisely those that are assigned to tracks t_{ij} in Q_2) that need to be assigned.
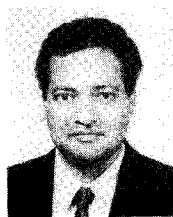
Consider the particular track among t_{ij}, $1 \leq j \leq n$ in Q_2 (note that by Proposition 14 there always exists such a track), that has one connection each from a_i and b_{ij} assigned to it and let these connections be $a_{\alpha(1)}$ and $b_{\alpha(1)\beta(1)}$. Then in Q assign $a_{\alpha(1)}$ and $b_{\alpha(1)\beta(1)}$ to track t_1 (the validity of this assignment follows immediately from the construction of the track t_1).

In general, let $a_{\alpha(i)}$ and $b_{\alpha(i)\beta(i)}$ be the connections assigned to one track among the n tracks t_{ij} $1 \leq j \leq n$. Then assign $a_{\alpha(i)}$ and $b_{\alpha(i)\beta(i)}$ to track t_i in Q. □

REFERENCES

[1] M. Lorenzetti and D. S. Baeder, "Routing," *Physical Design Automation of VLSI Systems*, vol. B. Preas and M. Lorenzetti, Eds. Menlo Park, CA: Benjamin/Cummings, 1988, chap. 5.

[2] H. Hsieh *et al.*, "A second generation user programmable gate array," in *Proc. Custom Integrated Circuits Conf.*, May 1987, pp. 515-521.

[3] A. El Gamal, J. Greene, J. Reyneri, E. Rogoyski, K. El-Ayat, and A. Mohsen, "An architecture for electrically configurable gate arrays," *IEEE J. Solid-State Circuits*, vol. 24, pp. 394–398, Apr. 1989.

[4] T. Szymanski, "Dogleg channel routing is NP-complete," *IEEE Trans. Computer-Aided Design*, vol. CAD-4, pp. 31–41, Jan. 1985.

[5] A. Hashimoto and J. Stevens, "Wire routing by optimizing channel assignment within large apertures," in *Proc. 8th IEEE Design Automation Workshop*, 1971.

[6] C. H. Papadimitrou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Englewood Cliffs, NJ: Prentice-Hall, 1982.

[7] M. R. Garey and D. S. Johnson, *Computers and Intractability—A Guide to the Theory of NP-Completeness*. New York: Freeman, 1979.

[8] W. J. Dally, "Express cubes: Improving the performance of the *k*-ary *n*-cube interconnection networks," MIT, Cambridge, MA, VLSI Memo 89-564, Oct. 1989.

[9] A. El Gamal, "Two dimensional stochastic model for interconnections in master slice integrated circuits," *IEEE Trans. Circuits Syst.*, vol. CAS-28, pp. 127–128, Feb. 1981.

[10] A. El Gamal, J. Greene, and V. P. Roychowdhury, "Segmented channel routing is as efficient as conventional routing (and just as hard)," in *Proc. 13th Conf. Advanced Research VLSI*, Santa Cruz, CA, Mar. 1991, pp. 192–211.

[11] J. Greene, V. P. Roychowdhury, S. Kaptanoglu, and A. El Gamal, "Segmented channel routing," in *Proc. 27th ACM/IEEE Design Automation Conf.*, June 1990, pp. 567–572.

[12] T. Varvarigou, "A linear programming approach to segmented channel routing," Dept. Elect. Eng., Stanford Univ., Stanford, CA, Project Rep. EE391.

gorithms and architectures, design and analysis of neural networks, special purpose computing arrays and VLSI design, and fault-tolerant computation.

Jonathan W. Greene (S'80-M'84) received the Sc.B. degree in biology from Brown University, Providence, RI, in 1979, and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, in 1983.

He is currently with BioCAD Corp., Mountain View, CA, applying computer-aided design and information theoretic techniques to the process of drug discovery. He has worked on various aspects of computer-aided IC design at Hewlett-Packard Laboratories and the LSI Logic Systems Research Lab. From 1986 to 1990 he was with Actel Corporation, where he helped develop the architecture and software for their field-programmable gate arrays and became Director of System Architecture. He has also been a Visiting Scholar in the Information Systems Laboratory at Stanford University, and at Columbia University, NY, Chemistry Department.

Vwani P. Roychowdhury received the B.Tech. degree from the Indian Institute of Technology, Kanpur, India and the Ph.D. degree from Stanford University, Stanford, CA, in 1982 and 1989, respectively, all in electrical engineering.

From September 1989 to August 1991 he was a research associate in the Department of Electrical Engineering, Stanford University. He is currently an Assistant Professor in the Electrical Engineering Department, Purdue University, West Lafayette, IN. His research interests include parallel algorithms and architectures, design and analysis of neural networks, special purpose computing arrays and VLSI design, and fault-tolerant computation.

Abbas El Gamal (S'71-M'73-SM'83) received the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, in 1978.

He is currently an Associate Professor of Electrical Engineering at Stanford University. From 1978–1980 was an assistant professor of electrical engineering at the University of Southern California, Los Angeles, CA. From 1981–1984, he was an assistant professor of electrical engineering at Stanford. He was on leave from Stanford from 1984 to 1987, first as director of the LSI Logic Research Lab, then as founder and Chief Scientist of Actel Corp. His research interests include semi-custom VLSI architectures, design automation and synthesis for VLSI, configurable VLSI, complexity theory, and information theory.

Analytical Framework for Switch Block Design

Guy G. Lemieux and David M. Lewis

Edward S. Rogers Sr. Dept. of Electrical and Computer Engineering
University of Toronto, Toronto, Ontario, Canada
{Lemieux, Lewis}@eecg.utoronto.ca

Abstract. One popular FPGA interconnection network is based on the island-style model, where rows and columns of logic blocks are separated by channels containing routing wires. Switch blocks are placed at the intersections of the horizontal and vertical channels to allow the wires to be connected together. Previous switch block design has focused on the analysis of individual switch blocks or the use of ad hoc design with experimental evaluation. This paper presents an analytical framework which considers the design of a continuous fabric of switch blocks containing wire segments of any length. The framework is used to design new switch blocks which are experimentally shown to be as effective as the best ones known to date. With this framework, we hope to inspire new ways of looking at switch block design.

1   Introduction

Over the past several years, a number of different switch block designs have been proposed such as those shown in Figure 1. FPGAs such as the Xilinx XC4000-series [1] use a switch block style known as *disjoint*. Some alternatives to this style, known as *universal* [2] and *Wilton* [3], require fewer routing tracks and use less transistor area with interconnect of single-length wires. However, with longer wire segments they use more switches per track and often require more transistor area overall [4]. The *Imran* block [5] addresses this overhead by modifying the *Wilton* pattern to use the same number of switches as the *disjoint* pattern.

These switch blocks are designed using different methodologies. The *universal* switch block is analytically designed to be independently routable for all two-point nets. Recently, the *hyperuniversal* switch block [6] extends this for multi-point nets. These blocks rely on reordering nets at every switch block, so their local optimality does not extend to the entire routing fabric. In comparison, the *Wilton* and *Imran* switch blocks are examples of ad hoc design with experimental validation. The *Wilton* block changes the track number assigned to a net as it turns. This way, two different global routes may reach two different tracks at the same destination channel. This forms two disjoint paths, a feature we call the *diversity* of a network. The *Wilton* and *Imran* designs introduce the notion that a switch block must consider its role as part of a larger switching fabric.

The above methods have produced switch blocks that perform well, but there is no formal method to design a switch block while considering the overall routing fabric. In pursuit of this goal, this paper introduces an analytical framework which considers
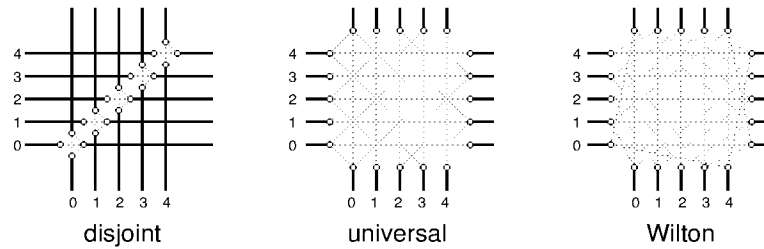
Fig. 1. Different switch block styles.

both long wire segments and the interaction of many switch blocks connected together. This framework includes a restricted switch block model which allows us to analyse the diversity of the network. The framework is used to design an ad hoc switch block named *shifty* and two analytic ones named *diverse* and *diverse-clique*. These new switch blocks are very diverse, and routing experiments show they are as effective as the others.

2   Design Framework

This section describes the switch block framework being composed of a switch block model, permutation mapping functions, and simplifying assumptions and properties.

2.1   Switch Block Model

The traditional model of a switch block draws a large box around the intersection of a horizontal and vertical routing channel. Within the box, switches connect a wire on one side to any wires on the other three sides. Long wire segments pass straight across the switch block, but some track shifting is necessary to implement fixed length wires with one layout tile. Figure 2a) presents this model in a new way by partitioning the switch block into three subblocks: *endpoint* (f_e), *midpoint* (f_m), and *midpoint-endpoint* (f_{me}) subblocks. The endpoint (midpoint) subblock is the region where the ends (midpoints) of wire segments connect to the ends (midpoints) of other wire segments. The f_{me} subblock connects the middle regions of some wires to the ends of others. A switch placed between two sides always falls into one of these subblocks.

   The traditional model in Figure 2a) is too general for simple diversity analysis, so we propose restricting the permissible switch locations. One restriction is to prohibit f_{me} switches; this was done in the *Imran* block [5]. We propose to further constrain the f_m switch locations to lie within smaller subblocks called $f_{m,i}$, as shown in Figure 2b) for length-four wires. This *track group model* is a key component to the framework.

   The track group model partitions wires into *track groups* according to their wire length and starting points. The midpoint subblocks are labeled $f_{m,i}$, where i is a position between 1 and $L - 1$ along a wire of length L. This model is somewhat restrictive, but it can still represent many switch blocks, *e.g.*, *Imran*, and we will show that it performs well. As well, early experiments we conducted without the $f_{m,i}$ subblock restric-
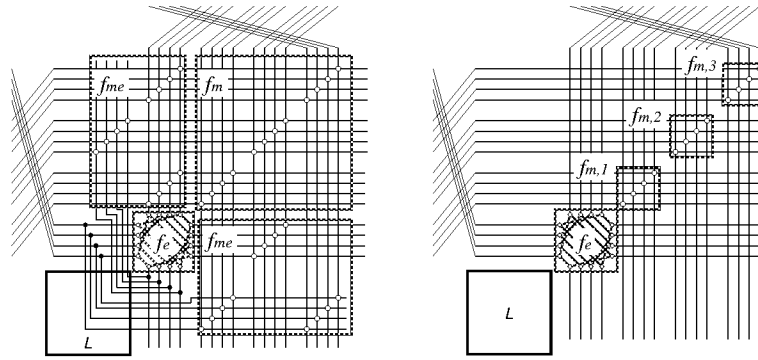
Fig. 2. Switch block models with subblocks, a) traditional and b) track group model.

tions did not produce better results. However, the $f_{m,i}$ subblocks explicitly force track groups to be in separate routing domains so each group can be treated independently.

2.2   Permutation Mapping Functions

Previous work suggests only a small number of switches need to be placed within a switch block. Early work [7] defined switch block flexibility, F_s, as the number of other wires connecting to each wire in this block. They found that $F_s = 3$ is the lowest that is routable with single-length wire segments. Other work [4, 5] has used $F_s = 3$ at wire endpoints and $F_s = 1$ at wire midpoints when long wire segments are used. As well, our experience with $F_s < 3$ is that a few more tracks but less transistor area is needed [8]. This suggests $6W$ and W are reasonable upper bounds for the number of switches in endpoint and midpoint subblocks, respectively.

Given these upper bounds, switch locations can be represented by a *permutation mapping function* between each pair of sides. The different mapping functions and their implied forward direction are shown in Figure 3. In this figure, $f_{e,i}(t)$, or simply $f_{e,i}$, represents an endpoint turn of type i. A switch connects the wire originating at track t to track $f_{e,i}(t)$ on the destination side. Turns in the reverse direction to those indicated are represented as $f_{e,i}^{-1}$ such that $f^{-1}(f(t)) = t$.

Similarly, $f_{m,i}$ is a mapping function for a midpoint turn at position i along the length of a wire, with the most South/West endpoint being the origin at position $i = 0$. Figure 3b) illustrates the different midpoint subblocks in a fabric of 2×4 logic blocks (L) for a single track group. The other three track groups are independent, but they would be similar and have staggered starting locations. There are no connections between the track groups.

Examples of mapping functions for various switch blocks are shown in Table 1. Each of these functions are modulo W, where W is the track group width. Also, note that it is common for connections straight across a switch block (E–W or N–S) to stay in the same track, so it is usually assumed that $f_{e,5} = f_{e,6} = t$.
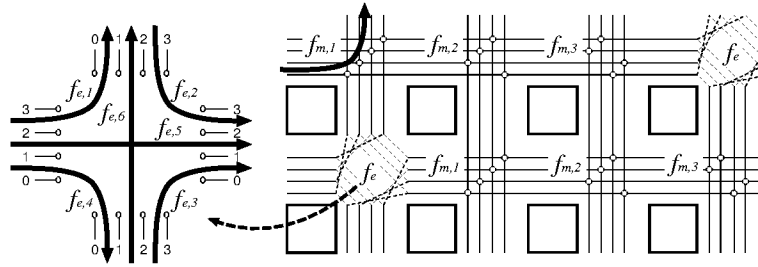
Fig. 3. Mapping functions for a) endpoint and b) midpoint subblock turns.

2.3 Additional Assumptions

In addition to the explicit assumptions above, there are a few implicit ones being made as well. It is assumed that the subblocks are square with W tracks on each side and there is a one-to-one correspondence between the originating track and the destination track. Since $f^{-1}(f(t)) = t$, it is also presumed that each switch is bidirectional. Additionally, we assume a track group contains only one wire length and switch type.

2.4 Commutative Switch Blocks

The mapping functions of the *universal* and *Imran* switch blocks involve *twists* where the function is of the form $f(t) = W - t + c$. Unfortunately, these functions are difficult to analyse because the twist is not commutative. Using commutative functions simplifies the model because the order in which turns are made becomes unimportant. Paths with an arbitrary number or sequence of turns can be reduced to a canonical permutation which uniquely determines the destination track. Later in Section 3.2, this will allow us to significantly reduce the search space. We define a switch block to be *commutative* if all of its mapping functions are commutative.

   Consider the example shown in Figure 4, where two paths are compared in two different architectures. The left architecture uses commutative switch blocks, but the right one does not. The destination track of the upper path is $f_{e,2}(f_{e,4}(f_{e,3}(f_{e,1}(t))))$, while the lower path is $f_{e,3}(f_{e,1}(f_{e,2}(f_{e,4}(t))))$. In a commutative architecture, both paths can be rewritten as $f_{e,1}(f_{e,2}(f_{e,3}(f_{e,4}(t))))$. These necessarily reach the same track. In a non-commutative architecture, the operations cannot be reordered and the paths may reach different tracks. This example suggests that commutative architectures are less diverse. However, results will demonstrate that commutative switch blocks are very diverse and as routable as the non-commutative *Imran* block.

3   Framework Applications

To illustrate the use of the new framework, two approaches will be used to determine a set of permutation mapping functions. The first, named *shifty*, is an ad hoc commutative switch block. The second creates two switch blocks, named *diverse* and
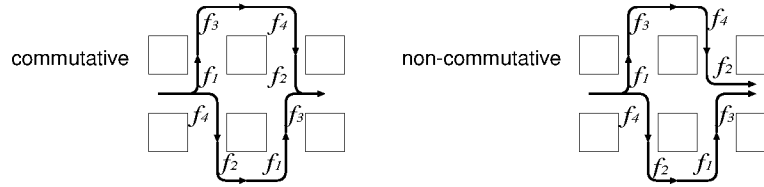
Fig. 4. Turn order is not important in commutative switch blocks.

diverse-clique, by optimizing diversity. Both of these approaches assume length-four interconnect wires. As well, they will assume that two separate layout tiles can be used in a checkered fashion to further increase diversity.

3.1   Application: *shifty* and *universal-TG* Designs

The first application of the new framework is the design of a commutative switch block similar to *Imran* but without the non-commutative twists. The following mapping functions describe the new switch block: $f_{e,1} = t-1$, $f_{e,2} = t-3$, $f_{e,3} = t-2$, $f_{e,4} = t-4$, and $f_{m,i} = t$ (mod W). This block is named *shifty* because each turn involves a shift from one track number to another by a constant amount. The constant values are chosen to be small because the arithmetic is always done modulo W. This avoids $f_{e,1}$ from being equivalent to $f_{e,4}$, for example, except with certain small W values.

Other switch blocks can also be adopted within this framework. For example, the *disjoint* and *Imran* switch blocks naturally conform to the track group model already. As well, suppose the *universal* pattern is applied only at endpoint subblocks and the identity mapping $f_{m,i} = t$ is used at midpoint subblocks. This new pattern, *universal-TG*, is similar to the original in that *each subblock* can connect any set of two-point nets that obey basic bandwidth constraints. It also requires less transistor area with long wire segments in the same way that *Imran* improves *Wilton* by reducing the number of switches per track.

To create additional diversity, it is possible to use two different switch block designs arranged in a checkerboard pattern. If the above switch blocks are assigned to the white square locations, a modified one can be used on the black square locations. These black switch blocks are characterized by their own mapping functions, g. Ad hoc designs for various g switch blocks, which are chosen to be slightly different from their f counterparts, are shown in Table 1. In choosing the specific g_e functions for the *disjoint* and *universal-TG* blocks, care is taken to preserve their layout structures by merely re-ordering the horizontal tracks.

3.2   Application: *diverse* and *diverse-clique* Designs

This section will use the design framework to develop commutative switch blocks that are maximally diverse for all possible two-turn paths. Two different switch blocks will be designed, *diverse* and *diverse-clique*. The latter design is more restricted because its endpoint subblock uses the 4-wire clique layout structure of the *disjoint* switch block.

Table 1. Complete switch block mappings used for white (f) and black (g) squares

| Turn | White Square Switch Block | | | | Turn | Black Square Switch Block | | | |
	disjoint	universal-TG	Imran	shifty		disjoint	universal-TG	Imran	shifty
$f_{e,1}$	t	$W-t-1$	$W-t$	$t-1$	$g_{e,1}$	$t-1$	$W-t-2$	$W-t+3$	$t-8$
$f_{e,2}$	t	t	$t+1$	$t-3$	$g_{e,2}$	$t+1$	$t+1$	$t+3$	$t-7$
$f_{e,3}$	t	$W-t-1$	$W-t-2$	$t-2$	$g_{e,3}$	$t+1$	$W-t$	$W-t+2$	$t-9$
$f_{e,4}$	t	t	$t-1$	$t-4$	$g_{e,4}$	$t-1$	$t-1$	$t+1$	$t-6$
$f_{m,i}$	t	t	t	t	$g_{m,i}$	$t+1$	$t+1$	$t+1$	$t+1$

This design approach is repeated for an architecture containing two layout tiles, f and g, arranged in a checkered pattern.

Design Space Let each switch block mapping function be represented by the equations $f_i(t) = t + a_i \bmod W$ or $g_i(t) = t + b_i \bmod W$, where i represents one of the endpoint or midpoint turn types. The a_i and b_i values are constants which can be summarized in vector form as:

$$\boldsymbol{x}_W = \begin{bmatrix} a_{e,1} & a_{e,2} & a_{e,3} & a_{e,4} & a_{m,1} & a_{m,2} & a_{m,3} & b_{e,1} & b_{e,2} & b_{e,3} & b_{e,4} & b_{m,1} & b_{m,2} & b_{m,3} \end{bmatrix}^T.$$

Note that a solution $\boldsymbol{x}_W$ is only valid for a specific value of W. Constraining f and g in this way explores only a portion the design space. However, this is sufficient to develop very diverse switch blocks.

Enumerating the Path-Pairs Before counting diversity, we enumerate all paths containing two turns and the pairs of these paths that should be diverse.

The six basic two-turn paths created by $\binom{4}{2} = 6$ pairs of single turns are: ENE, ESE, ENW, WNE, NES and SEN, where N, S, E, or W refer to compass directions. For example, two different ENE paths, using columns A and B to reach row $out1$, are shown in Figure 5. In general, the commutative property allows all ENE paths (or ESE paths, etc.) of an infinite routing fabric to be enumerated using the 8×8 grid or *supertile* in Figure 5. The size of the supertile arises from the length-four wires and two (checkerboard) layout tiles. Within it, each subblock is labeled with the mapping functions from one track group.

A number of isomorphic paths can be eliminated using the supertile and commutative property. Longer horizontal or vertical distances would reach another supertile and turn at a switch block equivalent to one in this supertile. Similarly, other input rows can be disregarded. Since NEN and SES paths are commutatively equivalent to ENE and ESE paths, they are also ignored.

For maximum diversity, each *pair of paths* that reach the same output row must reach different tracks. With 8 possible routes (columns A–H), there are $\binom{8}{2} = 28$ pairs of paths to be compared. Hence, for all turn types and all output rows, there are $6 \times 7 \times 28 = 1176$ *path-pairs* to be compared.

Counting Diversity To detect diversity between a pair of paths, first compute the difference between the two permutation mappings, $y = f_{pathA} - f_{pathB}$. The path-pair is diverse if y is non-zero. This can be written in matrix form as $\boldsymbol{y} = \boldsymbol{A} \cdot \boldsymbol{x}_W$ where each
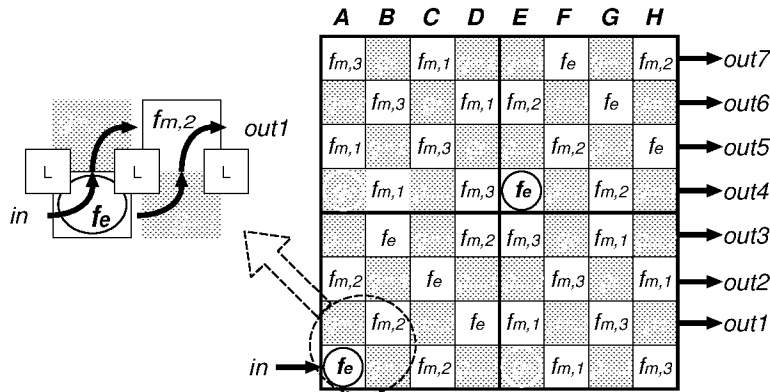
Fig. 5. An 8 × 8 grid or supertile used for enumerating all two-turn paths.

row in A is predetermined based on the path-pair being considered, and each row in y is the corresponding diversity test result. The number of rows has been considerably reduced by the large number of equivalent paths eliminated earlier. Additional types of path-pairs (not only those with two turns) can be represented by adding more rows to A and y.

Diversity of a given switch block x_W is measured by counting the number of non-zero entries in y. For our architecture, the maximum diversity is 1176.

Searching Design Space Rather than solve large matrix equations to maximize the number of non-zero values in y, we performed various random and brute-force searches of x_W for W ranging from 2 to 18. Typically, an exhaustive search produced the best results in about one CPU-day (1 GHz Pentium) even though it wasn't allowed to run to completion.

Switch Blocks Created Using the above procedure, switch blocks named *diverse* are designed for a variety of track group widths, $W \leq 18$. For each W, a solution set x_W is found. Similarly, we designed a *diverse-clique* switch block which preserves the 4-wire clique structure at endpoint subblocks. A layout strategy for these cliques is given in [9]. The precise solution sets obtained for these two switch blocks can be found in [8].

4   Results

The new switch blocks are evaluated below by counting diversity and computing the minimum channel width and area from numerous routing experiments.

Diversity Results The diversity of various switch blocks is shown in Figure 7. The *disjoint* switch block has no diversity but its checkered version has considerably more. The *shifty* switch block and its checkered version provide even more diversity. However, the *diverse* and *diverse-clique* checkered switch blocks reach the highest levels
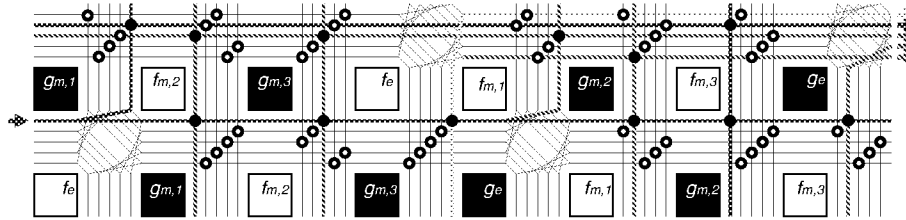
Fig. 6. Checkered layout showing diversity of a $W = 5$ *diverse-clique* switch block.

of diversity. For $W > 10$, these are within 99% of the maximum possible. However, note that it is impossible to attain maximum diversity when $W < 8$ because some of the 8 global routes necessarily map to the same track. Considering this, the *diverse* and *diverse-clique* switch blocks perform very well at being diverse.

Routing Results The experimental environment is similar to the one used in [4]. Benchmark circuits are mapped into 4, 5, and 6-input LUTs, clustered into groups of 6, and placed once. The new switch blocks are evaluated using a modified version [8] of the VPR router [4]. Routing experiments use only length four wires in the interconnect. Half of all tracks use pass transistors 16x minimum width, and the other half use buffers of size 6x minimum [10]. Although not shown, similar results are obtained if all wiring tracks contain buffers.

The routability performance of the new switch blocks is presented in Figures 8 and 9. The former plots the minimum number of tracks required to route, W_{min}, while the latter plots the transistor area of the FPGA at the low-stress point of $1.2 \times W_{min}$ tracks. The graphs on the left compare *shifty* to the older switch blocks, and the graphs on the right compare *disjoint* to the newer switch blocks. A number of different curves are drawn in the graphs, corresponding to different LUT sizes (as labeled) and whether one layout tile is used (**bold** curves) or two tiles are checkered (thin curves). Delay results have been omitted because there is no apparent correlation with switch block style.

The area and W_{min} results exhibit only small variations across the designs, so conclusions might be sensitive to noise and must be carefully drawn. Each data point is an arithmetic average of the twenty largest MCNC circuits, so large variations should not be expected unless many circuits are affected. To mitigate the influence of noise, it is important to identify trends in the results, *e.g.*, across all of the different LUT sizes.

Analysis One clear trend in the routing results is that the plain *disjoint* switch block performs worse than any perturbation of it (including its own checkered version). Beyond this, the ranking of specific switch blocks is difficult. It appears that *shifty* is the best, followed by *universal-TG* and *Imran*, then *disjoint*. The diversity-optimized switch blocks are better than *disjoint*, but worse than *shifty*.

In addition to *shifty*, a variety of other ad hoc switch blocks (both commutative and not) were explored. The *shifty* design gives better results, but the differences are small.
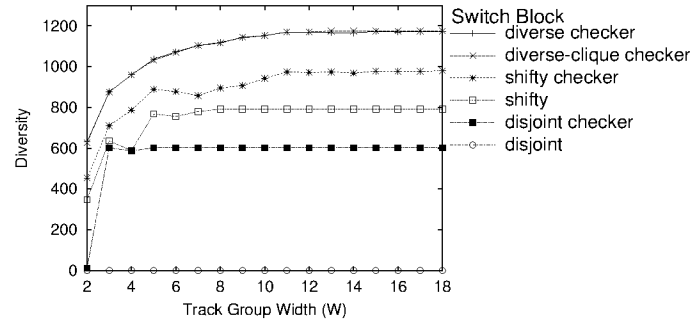
Fig. 7. Diversity of various commutative switch blocks.

These experiments did not clearly suggest that one particular design is significantly better. The effectiveness of *shifty* demonstrates that the twist or non-commutative features of the *universal-TG* and *Imran* blocks is not likely the key factor to their good performance. However, it makes us ask why is track shifting effective? Is it because of increased diversity?

The diverse switch blocks always require fewer routing tracks than the *disjoint* baseline. However, *shifty* always outperforms the diversity-optimized ones. This suggests that it is **not** the diverse property that makes *shifty* effective. It also counters the belief that the *Imran* and *Wilton* switch blocks are effective because they add some diversity.

Why do the diversity-optimized switch blocks not perform as well as anticipated? One conjecture is that negotiated-congestion type CAD tools, like the VPR router, might have difficulty with diversity. This seems plausible because a local re-routing near the source of a net would force most downstream connections to use a new track, even if they continue using the same routing channels. With less diversity, it may be easier for a net to resume using the previous routing tracks. This difficulty might increase the number of router iterations, but our experiments show little increase. Diversity adds a new degree of freedom to routing, but CAD tools must be able to efficiently utilise it.

5   Conclusions

This paper presents an analytical framework for the design of switch blocks. It is the first known framework to consider the switch block as part of an infinite switching fabric that easily works with long wire segments. The most fundamental component of this framework is the new track group switch block model.By separating wiring tracks into independent groups, each can be considered separately. Using permutation mapping functions to model switch block turns adds a mathematical representation to the framework. With commutative switch blocks, the order in which a net executes turns becomes unimportant and the network is easier to analyse. This framework can design diverse switch blocks, but it is not clear the router is utilising this diversity.
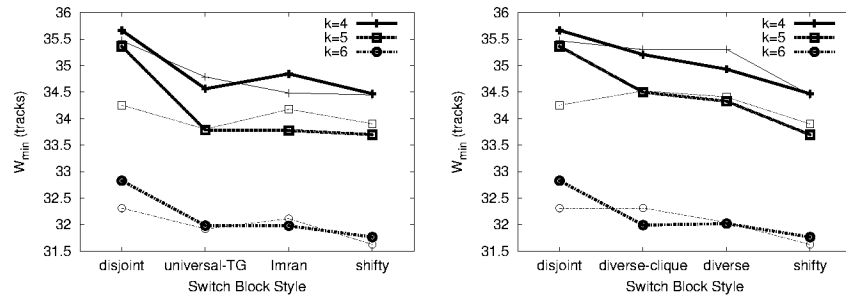
Fig. 8. Minimum channel width results using the new switch blocks.
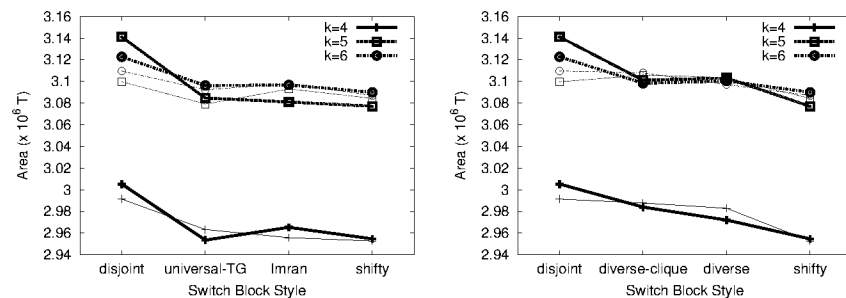
Fig. 9. Area results using the new switch blocks.

References

1. Xilinx, San Jose, CA, *Online Databook*, 2002. www.xilinx.com/partinfo/databook.htm.
2. Y.-W. Chang, D. F. Wong, and C. K. Wong, "Universal switch-module design for symmetric-array-based FPGAs," *ACM Transactions on Design Automation of Electronic Systems*, vol. 1, pp. 80–101, January 1996.
3. S. J. Wilton, *Architectures and Algorithms for Field-Programmable Gate Arrays with Embedded Memories*. PhD thesis, Dept. of ECE, Univ. of Toronto, 1997.
4. V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*. Boston: Kluwer Academic Publishers, 1999.
5. M. I. Masud and S. Wilton, "A new switch block for segmented FPGAs," in *Field Programmable Logic*, August 1999.
6. H. Fan, J. Liu, Y.-L. Wu, and C.-C. Cheung, "On optimum switch box designs for 2-D FPGAs," in *ACM/IEEE Design Automation Conference*, June 2001.
7. J. Rose and S. Brown, "Flexibility of interconnection structures in field-programmable gate arrays," *IEEE J of Solid State Ccts*, vol. 26, pp. 277–282, March 1991.
8. G. Lemieux, *In preparation*. PhD thesis, Dept. of ECE, Univ. of Toronto, 2002.
9. H. Schmit and V. Chandra, "FPGA switch block layout and evaluation," in *Int. Symp. on FPGAs*, (Monterey, CA), pp. 11–18, February 2002.
10. G. Lemieux and D. Lewis, "Circuit design of routing switches," in *Int. Symp. on FPGAs*, (Monterey, CA), pp. 19–28, Feb. 2002.

Generic Universal Switch Blocks

Michael Shyu, Guang-Ming Wu, Yu-Dong Chang, and Yao-Wen Chang, *Member, IEEE*

Abstract—A switch block M with W terminals on each side is said to be *universal* if every set of nets satisfying the dimension constraint (i.e., the number of nets on each side of M is at most W) is simultaneously routable through M [2]. In this paper, we present an algorithm to construct N-sided universal switch blocks with W terminals on each side. Each of our universal switch blocks has $\binom{N}{2}W$ switches and *switch-block flexibility* $N-1$ (i.e., $F_S = N-1$). We prove that no switch block with less than $\binom{N}{2}W$ switches can be universal. We also compare our universal switch blocks with others of the topology associated with Xilinx XC4000-type FPGAs. To explore the area performance of the universal switch blocks, we develop a detailed router for hierarchical FPGAs (HFPGAs) with 5-sided switch blocks. Experimental results demonstrate that our universal switch blocks improve routability at the chip level. Based on extensive experiments, we also provide key insights into the interactions between switch-block architectures and routing.

Index Terms—Analysis, architecture, design, digital, gate array, programmable logic array.

———————————————— ✦ ————————————————

1   INTRODUCTION

A conventional FPGA (see Fig. 1a) consists of an array of logic blocks that can be connected by routing resources [1]. The logic blocks contain combinational and sequential circuits which are used to implement logic functions. The routing resources consist of wire segments and switch blocks. The intersection of a horizontal and a vertical channel is referred to as a switch block; the switch block serves to connect wire segments and this requires using programmable switches inside it. Fig. 1b illustrates a switch block in which the programmable switches, denoted by dashed lines between terminals, are shown.

The studies by [5], [12] have proposed a new routing architecture called an *8-way mesh*. (See Fig. 2a for the architecture.) Similar to a conventional architecture, an 8-way mesh structure also consists of a two-dimensional array of logic blocks. However, unlike the conventional architecture, the pins of a logic block in an 8-way mesh are directly connected to their nearest four switch blocks in diagonal directions. A switch block used in the 8-way mesh is then not only connected by its four nearest neighboring switch blocks, but also by the four diagonal neighboring logic blocks. Thus, the switch blocks used in 8-way mesh FPGAs are 8-sided. (See Fig. 2b for an 8-sided switch block.) A topology of the architecture similar to the 8-way mesh architecture for interchip routing was also studied in [4].

A *hierarchical FPGA (HFPGA)* (see Fig. 3a and Fig. 3c) has a hierarchical interconnection structure [6], [3], [8]. An HFPGA can be hierarchically constructed by connecting logic blocks into clusters. First, k logic blocks are connected with a switch block. This step forms a one-level HFPGA. Then, k clusters are recursively connected together as a *supercluster*. As k clusters are connected into a supercluster, the number of levels of the hierarchy will be increased by one. It is called a k-HFPGA if a cluster has k subclusters [8].

For example, Fig. 3a is a four-level 2-HFPGA and Fig. 3c is a two-level 4-HFPGA.

In an HFPGA, the switch blocks are not the same as the conventional 4-sided switch blocks. For a k-HFPGA, each switch block, except the top-level one, is connected with k logic blocks and another switch block; therefore, the switch block can be considered a $(k+1)$-sided polygonal block. For example, the switch block of 2-HFPGA is 3-sided (see Fig. 3b) and the switch block of 4-HFPGA is 5-sided (see Fig. 3d). Therefore, switch blocks could have arbitrary numbers of sides and it is significant to consider the design and analysis of switch blocks with multiple sides.

For the work on conventional switch blocks (4-sided blocks), Rose and Brown in [10] defined the flexibility of a switch block, represented by F_S, as the number of programing switches between a terminal and others. They investigated the effects of different switch-block flexibilities on routing and suggested that $F_S = 3$ should often be sufficient for high routability. Chang et al. first presented a class of universal switch blocks in [2]. A switch block M with W terminals on each side is said to be *universal* if every set of nets satisfying the dimension constraint (i.e., the number of nets on each side of M is at most W) is simultaneously routable through M [2]. They proved that each of the universal switch blocks can accommodate significantly more routing instances than the Xilinx XC4000-type one of the same size. Recently, a report on the layout implementations of the universal switch blocks and the XC4000-type ones has also concluded that the universal switch blocks need smaller silicon areas—the *decomposition property*[1] of the universal switch blocks makes their layout very regular and compact [14].

In this paper, we present an algorithm to construct generic universal switch blocks with multiple sides. Each of our universal switch blocks has $\binom{N}{2}W$ switches and *switch-block flexibility* $N-1$ (i.e., $F_S = N-1$). We prove that no switch block with less than $\binom{N}{2}W$ switches can be universal.

————————————————————

• *The authors are with the Department of Computer and Information Science, National Chiao Tung University, Hsinchu 300, Taiwan.*
  E-mail: ywchang@cis.nctu.edu.tw.

1. A formal definition of the decomposition property will be given in Section 3.1.
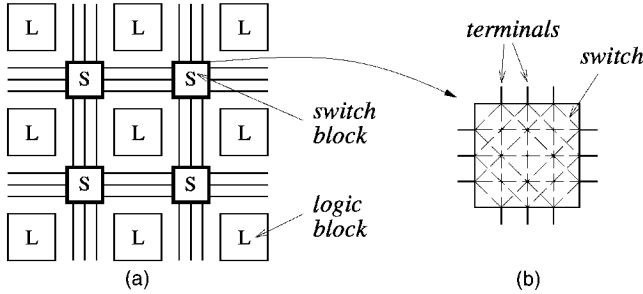
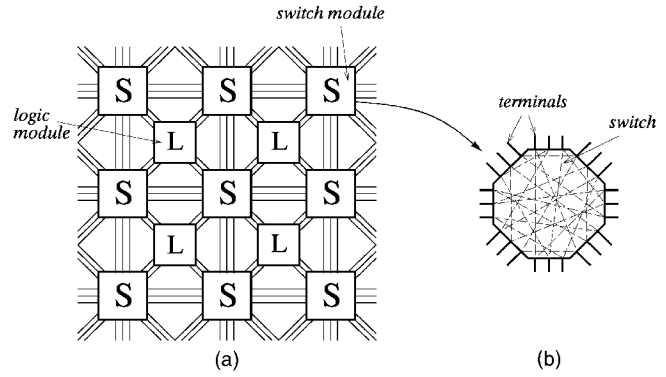Fig. 1. A conventional FPGA and its switch block. (a) A conventional FPGA architecture. (b) A conventional 4-sided switch block.

Fig. 2. An 8-way mesh FPGA and its switch block. (a) The architecture of an 8-way mesh. (b) The switch block used in (a).

We also compare our universal switch blocks with others of the topology associated with Xilinx XC4000-type FPGAs. To explore the area performance of the universal switch blocks, we develop a detailed router for 4-HFPGA. We model an HFPGA as a graph and apply a graph-search technique to HFPGA routing. Experimental results demonstrate that our universal switch blocks improve routability at the chip level. Based on extensive experiments, we also provide key insights into the interactions between switch-block architectures and routing.

The remainder of this article is organized as follows: Section 2 introduces some notation and definitions. Section 3 proposes an algorithm to construct generic universal switch blocks with multiple sides. Section 4 presents a graph modeling of HFPGAs and a graph-search technique for HFPGA routing. Section 5 shows our experimental results and discusses the interactions between the universal switch-block architectures and routing.

2 PRELIMINARIES

An *NSB* is an N-sided switch block with W terminals on each side of the block. We represent an NSB by $M_{N,W}(T, S)$, where T is the set of terminals and S the set of programming switches. Label the terminals

$$t_{1,1}, t_{1,2}, \ldots, t_{1,W}, t_{2,1}, t_{2,2}, \ldots, t_{2,W}, \ldots, t_{N,1}, t_{N,2}, \ldots, t_{N,W},$$

starting from the first terminal on one side and proceeding clockwise. Fig. 4a and Fig. 4b show the examples of $M_{3,3}(T, S)$ and $M_{5,3}(T, S)$, respectively. Let $T_i = \{t_{i,1}, \ldots, t_{i,W}\}$ and $S_i = \{(t_{m,n}, t_{u,v})|$, there exists a programmable switch between terminals $t_{m,n}$ and $t_{u,v}$, $t_{m,n} \in T_i$ or $t_{u,v} \in T_i, m < u\}$ for $1 \leq i \leq N$. Let $L_{i,W}(T_i, S_i)$ denote the terminals of side i and all switches connected with these terminals. Therefore, $S = \cup_{i=1}^{N} S_i$, $T = \cup_{i=1}^{N} T_i$, and $M_{N,W}(T, S) = \cup_{i=1}^{N} L_{i,W}(T_i, S_i)$. For convenience, we often refer to $M_{N,W}(T, S)$ and $L_{i,W}(T_i, S_i)$ simply as $M_{N,W}$ and $L_{i,W}$, respectively, omitting T and S, if there is no ambiguity about T and S or T and S are not of concern in the context.

In an NSB, the switches are electrically *noninteracting* unless they share a terminal. A *connection* is an electrical path between two terminals (say $t_{m,n}$ and $t_{u,v}$) on different sides of a switch block. If the switch $(t_{m,n}, t_{u,v})$ is programmed to be "ON," then a connection between these

two terminals is established. Because each connection is characterized by two sides of a block, we can classify all connections passing through a switch block into a number of categories. For an NSB, connections can be of $\binom{N}{2}$ types. Fig. 5a and Fig. 5b show the type definitions of 3-sided and 5-sided switch blocks, respectively.

A *routing requirement vector (RRV)* $\vec{n}$ for an NSB is an $\binom{N}{2}$-tuple $(n_{1,2}, n_{1,3}, \ldots, n_{1,N}, n_{2,3}, \ldots, n_{2,N}, \ldots, n_{N-1,N})$, where $n_{i,j}$ is the number of type-(i, j) connections required to be routed through an NSB, $0 \leq n_{i,j} \leq W$ for $1 \leq i < j \leq N$. An RRV $\vec{n}$ is said to be *routable* on an NSB $M_{N,W}$ if there exists a routing for $\vec{n}$ on $M_{N,W}$. For example, in a 3-sided switch block, Fig. 6a shows a routing instance with four nets corresponding to the RRV $(1, 2, 1)$, and Fig. 6b and Fig. 6c show two switch blocks with the same flexibility ($F_S = 2$). The RRV $(1, 2, 1)$ is routable on the switch block shown in Fig. 6b and a routing solution is illustrated by the thick lines. In Fig. 6c, however, there is always one net that cannot be routed into M_b. Thus, the RRV $(1, 2, 1)$ is not routable on the switch block shown in Fig. 6c.

The *routing capacity* of a switch block M is referred to as the number of distinct routable vectors on M; that is, the routing capacity of M is the cardinality $|\{\vec{n}|\vec{n}$ is routable on $M\}|$. A switch block M with W terminals on each side is called *universal* if every set of nets satisfying the dimension constraint (i.e., the number of nets on each side of M is at most W) is simultaneously routable through M. The dimension constraint can be denoted by an N-tuple $\vec{D}_{N,W} = (W, W, \ldots, W)$. For an RRV $\vec{n}$, its *dimension requirement vector (DRV)* $\vec{d}$ is an N-tuple $(d_1, d_2, \ldots, d_N)$ (i.e., the routing requirements on sides $1, 2, \ldots, N$ are $d_1, d_2, \ldots, d_N$, respectively), where $d_i = \sum_{1 \leq j \leq N} n_{i,j}$ for $1 \leq i \leq N, n_{i,j} = n_{j,i}$ and $n_{i,i} = 0$.

Note that RRVs and DRVs correspond to different concepts. An RRV, an $\binom{N}{2}$-tuple, is used to characterize nets routed through *a switch block*; however, a DRV, an N-tuple, is used to characterize nets routed through *each side* of a switch block. An RRV $\vec{n}$ satisfies the dimension constraint of size W if its DRV $\vec{d} \leq \vec{D}_{N,W}$ (i.e., $d_i \leq W$ for $1 \leq i \leq N$). Let the DRV of an RRV $\vec{n}$ be $\vec{d}$. We have the following definition.
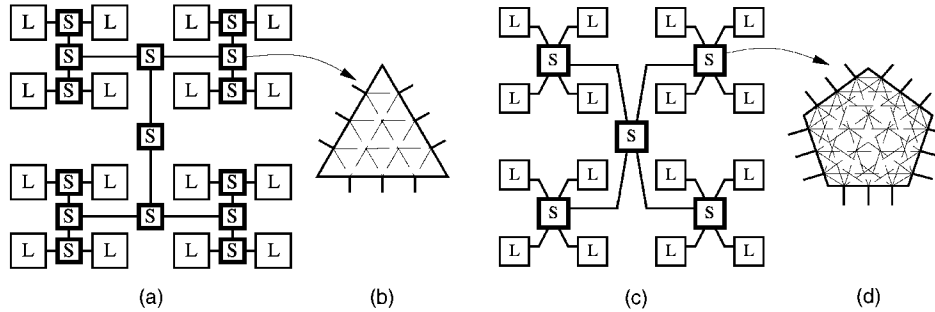
Fig. 3. Two HFPGA architectures. (a) A four-level 2-HFPGA. (b) A 3-sided switch block in the 2-HFPGA. (c) A two-level 4-HFPGA. (d) A 5-sided switch block in the 4-HFPGA.

Definition 1. *A switch block $M_{N,W}$ is called universal if $\vec{d} \leq \vec{D}_{N,W}$ is the necessary and sufficient condition for $\vec{n}$ to be routable on $M_{N,W}$.*

Note that the number of nets routed through each side of a switch block cannot exceed W; therefore, a universal switch block has the maximum routing capacity.

3 UNIVERSAL SWITCH BLOCKS

To identify a universal switch block, we first consider the clique-based one used in the Xilinx XC4000-series FPGAs. Fig. 6c and Fig. 7a and Fig. 7b show three such clique-based switch blocks of three, four, and five sides, respectively. Nevertheless, as shown in Fig. 6c, it is obvious that the clique-based switch blocks are not universal since the RRV $(1, 2, 1)$ which satisfies the dimension constraint is not routable on the 3-sided clique-based switch block of size three.

In the following, we consider a type of switch blocks which can be shown to be universal later.

3.1 Symmetric Switch Blocks

Let N $(N = 2, 3, 4, 5, \ldots)$ be the number of sides of a switch block and W the size of the switch block. Algorithm Symmetric_Switch_Block, shown in Fig. 8, constructs a switch block $M_{N,W}$. We refer to the topology of the switch block constructed by the algorithm as the *symmetric topology* and the switch block as the *symmetric switch block*. Fig. 9 shows three examples of symmetric switch blocks. For a

symmetric switch block, it has a flexibility (F_S) of $N - 1$; thus, the total number of switches used in the symmetric switch block is $\frac{N \times W \times F_S}{2} = \frac{N(N-1)}{2} W = \binom{N}{2} W$. For example, the total number of switches used by the 3-sided symmetric switch block, the square symmetric switch block, and the 5-sided symmetric switch block are $3W$, $6W$, and $10W$, respectively. Note that the switch block shown in Fig. 9b is a universal switch block proposed in [2].

For the symmetric switch blocks, we have the following properties:

Lemma 1 (Decomposition Property). *A symmetric switch block can be partitioned into $\lfloor W/2 \rfloor$ symmetric subblocks of size two and $(W \bmod 2)$ symmetric subblock(s) of size one.*

Proof. Consider Algorithm Symmetric_Switch_Block(N, W). For each k in line 3, we construct a symmetric subblock of size two in lines 4-6. Therefore, we have $\lfloor W/2 \rfloor$ subblocks of size two after $\lfloor W/2 \rfloor$ iterations (see line 3). Further, all these symmetric switch blocks of size two constructed in lines 4-6 have the same topology. Lines 7-10, just for an odd W, construct a clique of N vertices (i.e., a subblock of size one) from the middle terminal of each side of the switch block. Thus, we have ($W \bmod 2$) such subblock of size one (see lines 7-10). □

Lemma 2 (Reduction Property). *$M_{N,W}$ is a symmetric switch block, where $N \geq 3$, so is*

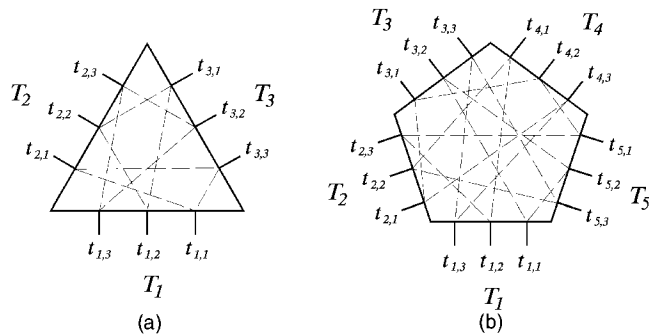$$M_{N-1,W} = M_{N,W} \setminus L_{i,W}, 1 \leq i \leq N.$$

Fig. 4. Two examples of N-sided switch blocks. (a) A 3-sided switch block $(M_{3,3}(T, S))$. (b) A 5-sided switch block $(M_{5,3}(T, S))$.
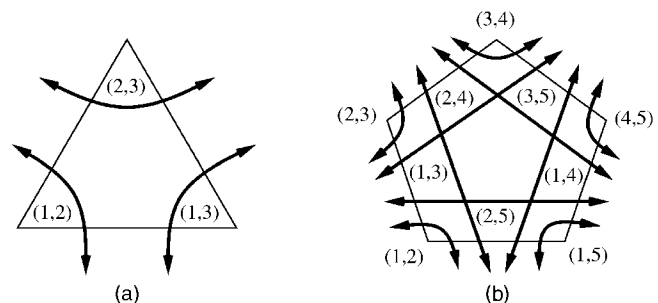
Fig. 5. Examples of types definitions. (a) Three types of connections in a 3-sided switch block. (b) Ten types of connections in a 5-sided switch block.
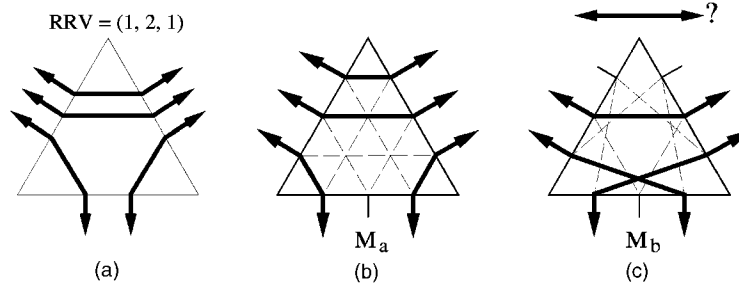
Fig. 6. An example of routing on two 3-sided switch blocks of the same size and flexibility. (a) A routing instance $(1, 2, 1)$. (b) $(1, 2, 1)$ is routable on M_a. (c) $(1, 2, 1)$ is not routable on M_b.

Fig. 10 illustrates the decomposition property. Note that the subblocks of each of the symmetric switch blocks are noninteracting to each other, thus each subblock can be considered independently. Lemma 1 is not only an important property in the proof of the universality of symmetric switch blocks, but is also the key to the layout implementation of a symmetric switch block with a smaller silicon area—the symmetric subblock of size two is a building block for a larger symmetric switch block (see Fig. 10), which can make the layout of a symmetric switch block very regular and compact [14]. Fig. 11 illustrates the reduction property; after removing one side i (terminals and switches) from a symmetric switch block $M_{5,2}$, the remainder is a symmetric switch block $M_{4,2}$.

3.2 Universality of Symmetric Switch Blocks

Let $R(N, W)$ denote the set of all RRVs that satisfy the dimension constraint for an N-sided switch block of size W. We can rewrite Definition 1 as follows:

Definition 2. A switch block $M_{N,W}$ is universal iff $\vec{n}$ is routable on $M_{N,W}$ for each $\vec{n} \in R(N, W)$.

We shall prove that each RRV in $R(N, W)$ is routable on our symmetric switch blocks. The space of $R(N, W)$, however, grows dramatically with N and W. (Specifically, the cardinality of $R(N, W)$ grows in $O(W^{\binom{N}{2}})$.) It is thus desirable to identify "critical" RRVs in $R(N, W)$.

3.2.1 Minimal Dominating Set of RRVs

For each $\vec{m}, \vec{n} \in R(N, W)$, $\vec{n}$ is said to *dominate* $\vec{m}$ if and only if $\vec{n} \geq \vec{m}$, i.e., $n_{i,j} \geq m_{i,j}, i,j = 1, 2, \ldots, N$. Any RRV $\vec{m}$ is routable on a switch block $M_{N,W}$ if there exists another RRV

$\vec{n}$ that is routable on $M_{N,W}$ and $\vec{n} \geq \vec{m}$ [13]. We have the following definition.

Definition 3. A subset $R_d(N, W)$ of $R(N, W)$ is a dominating set of $R(N, W)$ if $\forall \vec{m} \in R(N, W)$, $\exists \vec{n} \in R_d(N, W)$ such that $\vec{n} \geq \vec{m}$. A dominating set $R_d(N, W)$ is minimal if $\forall \vec{n_i}, \vec{n_j} \in R_d(N, W), i \neq j$, $\vec{n_i} \not\geq \vec{n_j}$ and $\vec{n_j} \not\geq \vec{n_i}$.

Similarly to [13], we have the following lemma.

Lemma 3. The minimal dominating set of $R(N, W)$ is unique.

Proof. Suppose that $R_{d1}(N, W)$ and $R_{d2}(N, W)$ are two different minimal dominating sets of $R(N, W)$. Consider the case when $R_{d1}(N, W) \not\subset R_{d2}(N, W)$. In this case, there exists an $\vec{n}$ such that $\vec{n} \in R_{d1}(N, W)$ and $\vec{n} \notin R_{d2}(N, W)$. Since $R_{d2}(N, W)$ is a dominating set, there exists a $\vec{m} \in R_{d2}(N, W)$ such that $\vec{m} \geq \vec{n}$. Since $R_{d1}(N, W)$ is also a dominating set, there exists a $\vec{p} \in R_{d1}(N, W)$ such that $\vec{p} \geq \vec{m}$ and, thus, $\vec{p} \geq \vec{n}$; a contradiction. Similarly, there is a contradiction in the case when $R_{d2}(N, W) \not\subset R_{d1}(N, W)$. Hence, the minimal dominating set of $R(N, W)$ is unique. □

An RRV $\vec{\gamma} \in R(N, W)$ is called a *maximal RRV* (*MRRV*) if there exists no other RRV in $R(N, W)$ that dominates $\vec{\gamma}$. The following lemma is the key to find the minimal dominating set of $R(N, W)$.

Lemma 4. $\Gamma(N, W) = \{\vec{\gamma} \mid \vec{\gamma} \text{ is an MRRV in } R(N, W)\}$ is the minimal dominating set of $R(N, W)$.

Proof. $\Gamma(N, W)$ is a dominating set since, for any RRV $\vec{n} \in R(N, W)$, there exists an MRRV $\vec{\gamma} \in \Gamma(N, W)$ such that $\vec{\gamma} \geq \vec{n}$. $\Gamma(N, W)$ is minimal since, for any two MRRVs in $\Gamma(N, W)$, they cannot be dominated by each other. □

Lemma 5. A switch block $M_{N,W}$ is universal iff $\vec{\gamma}$ is routable on $M_{N,W}, \forall \vec{\gamma} \in \Gamma(N, W)$.

Proof. For each $\vec{n} \in R(N, W)$, there exists $\vec{\gamma} \in \Gamma(N, W)$ that dominates $\vec{n}$. Since $\vec{\gamma}$ is routable on $M_{N,W}$, $\vec{n}$ is also routable on $M_{N,W}$. Hence, $M_{N,W}$ is universal. On the other hand, if $M_{N,W}$ is universal, each $\vec{\gamma} \in \Gamma(N, W) \subset R(N, W)$ is routable on $M_{N,W}$. □

3.2.2 Disjoint Routing Requirement Cycle Set

Based on Lemma 5, we shall focus our discussions on $\Gamma(N, W)$. If we can prove that each MRRV in $\Gamma(N, W)$ is routable on our symmetric NSB of size W, by Lemma 5, our
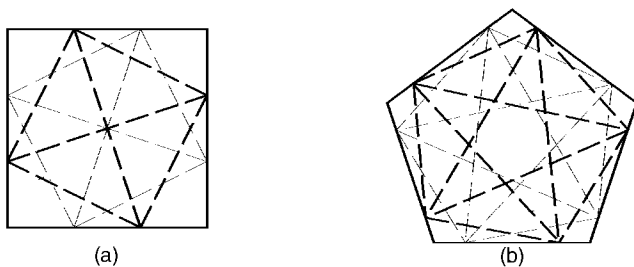

Fig. 7. Clique-based switch blocks. (a) A 4-sided one. (b) A 5-sided one.

Algorithm: Symmetric_Switch_Block(N, W)
Input: N—number of sides of the polygonal switch block;
      W number of terminals on each side of the switch block.
Output: $M_{N,W}(T, S)$—the N-sided symmetric switch block of size W;
      T: set of terminals; S: set of switches.
/* See Figure 4 for the terminal labeling. */

1   $T \leftarrow t_{i,j}, \quad \forall i = 1, 2, \ldots, N, \quad \forall j = 1, 2, \ldots, W$;
2   $S \leftarrow \emptyset$;
3   **for** $k = 1$ **to** $\lfloor \frac{W}{2} \rfloor$ **do**
4   **for** $i = 1$ **to** N **do**
5   **for** $j = 1$ **to** N **do**
6   **if** $i \neq j$
7               $S \leftarrow S \cup \{(t_{i,k}, t_{j,W-k+1})\}$;
8   **if** W is odd
9   **for** $i = 1$ **to** N **do**
10   **for** $j = 1$ **to** N **do**
11   **if** $i \neq j$
12            $S \leftarrow S \cup \{(t_{i,\lceil \frac{W}{2} \rceil}, t_{j,\lceil \frac{W}{2} \rceil})\}$;
13 **Output** $M_{N,W}(T, S)$.

Fig. 8. Algorithm for constructing an N-sided symmetric switch block of size W.
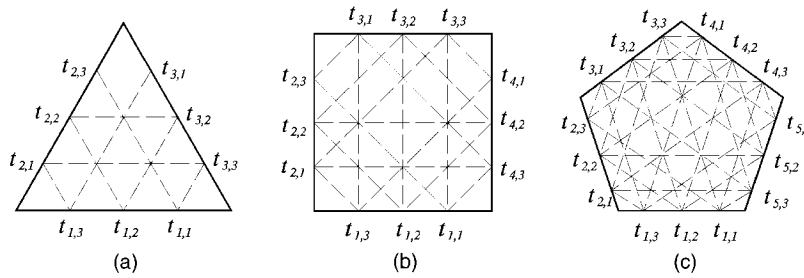
Fig. 9. Three symmetric switch blocks ($W = 3$). (a) A 3-sided symmetric switch block. (b) A square symmetric switch block. (c) A 5-sided symmetric switch block.

symmetric switch blocks are universal. Hence, the MRRV plays a very important role in our proof for universal switch blocks. We have the following lemma associated with MRRVs.

Lemma 6. *When an MRRV $\vec{\gamma} \in \Gamma(N, W)$ is routed on an N-sided switch block of size W, all unused terminals, if any, are on the same side and the number of unused terminals $\phi_{unused} = 2k + (N\ 2)(W \mod 2)$, where $k = 0, 1, 2, \ldots,$ or $\lfloor W/2 \rfloor$.*

Proof. If there are two unused terminals on the different sides (say side i and side j, $i < j$), we can increase $\gamma_{i,j}$ by 1 without violating the dimension constraint, which implies that $\vec{\gamma}$ is not maximal; a contradiction. Hence, all unused terminals, if any, must be on the same side.

The total number of terminals is $\phi_{total} = NW$. Assume that there are ϕ_{used} used terminals; obviously, ϕ_{used} is even since each switch is incident on two terminals and $\phi_{total} - \phi_{used} \leq W$ since all unused terminals, if any, must be on the same side. If N or W is even, ϕ_{total} is even. We have $\phi_{unused} = \phi_{total} - \phi_{used} = 2k$, where $k = 0, 1, 2, \ldots,$ or $\lfloor W/2 \rfloor$. If N and W are both odd, ϕ_{total} is odd. We have

$\phi_{unused} = \phi_{total} - \phi_{used} = 2k + 1$, where $k = 0, 1, 2, \ldots,$ or $\lfloor W/2 \rfloor$. Hence, $\phi_{unused} = 2k + (N \mod 2)(W \mod 2)$, where $k = 0, 1, 2, \ldots,$ or $\lfloor W/2 \rfloor$. $\qquad \square$

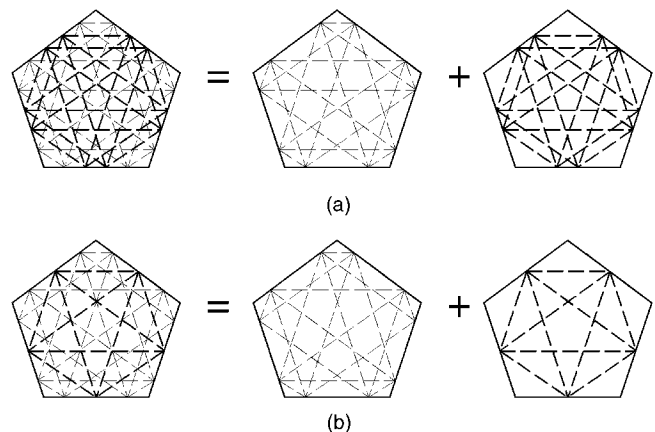

Fig. 10. Two 5-sided symmetric switch blocks and their subblocks. (a) Decomposition of the symmetric switch block of $W = 4$. (b) Decomposition of the symmetric switch block of $W = 3$.
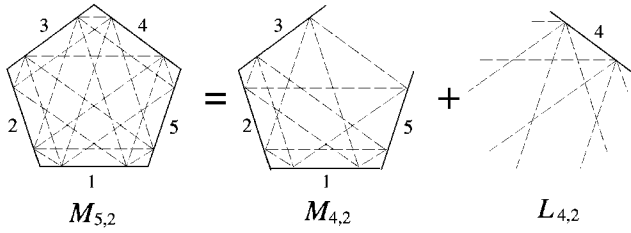
Fig. 11. Reduction of a 5-sided symmetric switch block.

(a)   (b)

(c)   (d)

Fig. 12. Two routing examples of MRRVs in $\Gamma(5,2)$ and their corresponding RRGs. (a) A complete routing example. (b) The corresponding RRG of (a). (c) A degenerate complete routing example. (d) The corresponding RRG of (c).

We first consider the MRRVs in $\Gamma(N, 2)$. By Lemma 6, we can classify the MRRVs into two types. One type is that all terminals are used (i.e., $\phi_{unused} = 0$) and we call it a *complete MRRV* (see Fig. 12a). The other type is that two terminals on some side are unused (i.e., $\phi_{unused} = 2$) and we call it a *degenerate complete MRRV* (see Fig. 12c).

For an RRV $\vec{n}$, we introduce an undirected *routing requirement graph* (RRG) G. Each vertex v_i in $V(G)$ represents one side s_i of $\vec{n}$. If there is a routing requirement between two different sides s_i and s_j, we introduce an edge between vertices v_i and v_j. For a complete MRRV in $\Gamma(N, 2)$, since all terminals are used, the degree of each vertex in the corresponding RRG is two (see Fig. 12b). For a degenerate MRRV in $\Gamma(N, 2)$, since all terminals are used except two terminals on some side, say side s_i, the degree of each vertex in the corresponding RRG is two except that the degree of v_i is zero (see Fig. 12d). We refer to an RRG with the degree of each vertex two as a *2-RRG*. A cycle in an RRG is called a *routing requirement cycle* (RRC). Two routing cycles C_i and C_j are *disjoint* if $V(C_i) \cap V(C_j) = \emptyset$. A *disjoint routing requirement cycle set* (DRRCS) Δ is a set of disjoint RRCs. We have the following lemmas associated with DRRCS.

Lemma 7. *A connected 2-RRG forms a cycle.*

Proof. Since the graph is connected and the degree of each vertex is even, there exists a Eulerian circuit. Since the degree of each vertex is 2, the Eulerian circuit forms a cycle.   □

Lemma 8. *A 2-RRG G can be divided into a DRRCS Δ such that*
$$\cup C_i = G, \sum |V(C_i)| = |V(G)|, \forall C_i \in \Delta.$$

Proof. An RRG G can be divided into k ($k \geq 1$) connected components $C_1, C_2, \ldots, C_k$, where
$$\cup C_i = G, \sum |V(C_i)| = |V(G)|$$
and
$$V(C_i) \cap V(C_j) = \emptyset, i, j = 1, 2, \ldots, k, i \neq j.$$

Let $\Delta = \{C_1, C_2, \ldots, C_k\}$. By Lemma 7, each component C_i forms a cycle. Since $V(C_i) \cap V(C_j) = \emptyset, i, j = 1, 2, \ldots, k, i \neq j$, Δ is a DRRCS.□

For example, in Fig. 12b, the DRRCS is
$$\{< v_1, v_3, v_4 >, < v_2, v_5 >\}.$$

Lemma 9 (MRRV Decomposition Property). *In an RRG G with the degree of each vertex larger than two, there exists a DRRCS Δ such that*
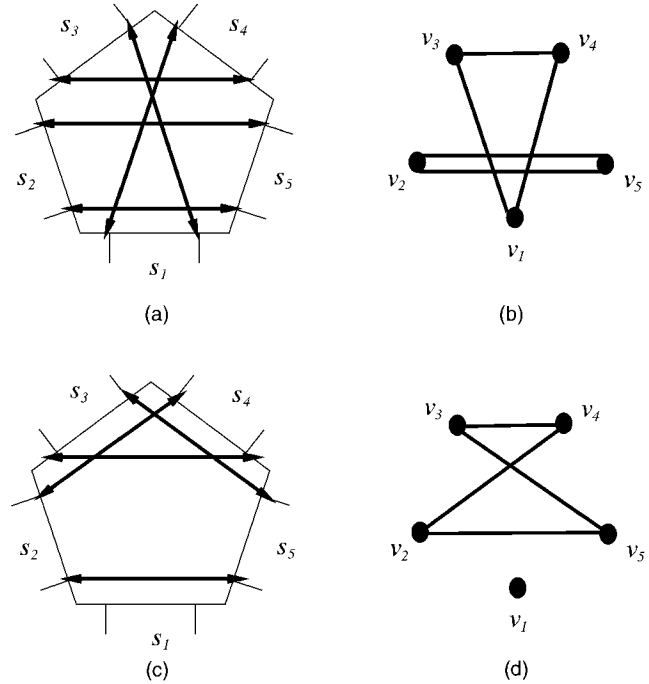$$\cup C_i \subset G, \sum |V(C_i)| = |V(G)|, \forall C_i \in \Delta.$$

Proof. Since the degree of each vertex in G is larger than 2, there exists a 2-RRG $G' \subset G$, where $|V(G')| = |V(G)|$. By Lemma 8, there exists a DRRCS Δ such that $\cup C_i = G' \subset G, \sum |V(C_i)| = |V(G')| = |V(G)|, \forall C_i \in \Delta.$ □

By Lemmas 7 and 8, an RRG constructed by a DRRCS is a 2-RRG. Thus, we shall focus our discussions on the relation between DRRCSs and symmetric switch blocks of size two.

3.2.3 Proof of Universality

We proceed to prove the universality of symmetric switch blocks. Based on the reduction property of symmetric switch blocks (Lemma 2), we have the following lemma by induction.

Lemma 10. *Given a DRRCS Δ, where $\sum |V(C_i)| = N, \forall C_i \in \Delta$. The RRV corresponding to Δ is routable on the symmetric switch block $M_{N,2}$.*

Proof. We can prove this lemma by induction. When $N = 2$, by the definition of symmetric switch blocks, this lemma holds. Assume that the lemma holds for $N < k$. When $N = k$, assume there are p disjoint RRCs in $\Delta = \{C_1, C_2, \ldots, C_p\}$. Let the RRV corresponding to Δ be $\vec{\gamma}$. There are two cases for p. In case 1, when $p > 1$, then $|V(C_1)| < k$. Let $\Delta = \Delta_1 \cup \Delta_2$, where $\Delta_1 = \{C_1\}$ and $\Delta_2 = \{C_2, C_3, \ldots, C_p\}$. Let the RRVs corresponding to Δ_1 and Δ_2 be $\vec{\gamma'}$ and $\vec{\gamma''}$, respectively, where $\vec{\gamma} = \vec{\gamma'} + \vec{\gamma''}$. Since $|V(C_1)| < k$, $\vec{\gamma'}$ is routable on the symmetric switch block $M_{|V(C_1)|,2}$ and must be routable on $M_{k,2}$. $\vec{\gamma'}$ thus can first be routed on $M_{k,2}$. Let $M_{k-|V(C_1)|,2} = M_{k,2} \setminus \{L_{s_i,2}|,$

each side s_i used by $\vec{\gamma}'$}. By Lemma 2, $M_{k-|V(C_1)|,2}$ is still a symmetric switch block. Since

$$\sum |V(C_i)| = k - |V(C_1)| < k, \forall C_i \in \Delta_2,$$

$\vec{\gamma}''$ is routable on $M_{k-|V(C_1)|,2}$. Hence, $\vec{\gamma}$ is routable on $M_{k,2}$ in case 1. In case 2, when $p = 1$, there is only one RRC in Δ and $|V(C_1)| = k$. Let the RRC sequence be $< v_{i_1}, v_{i_2}, \ldots, v_{i_k} >$. By the definition of symmetric switch block, the RRC sequence can be successfully routed by using k switches

$$(t_{s_{i_1},1}, t_{s_{i_2},2}), (t_{s_{i_2},1}, t_{s_{i_3},2}), \ldots, (t_{s_{i_{k-1}},1}, t_{s_{i_k},2}), (t_{s_{i_k},1}, t_{s_{i_1},2}).$$

Therefore, $\vec{\gamma}$ is routable on $M_{k,2}$ in case 2. □

Before proving the universality of generic symmetric switch blocks, we first consider the universality of symmetric switch blocks of size one and that of size two. Based on the definition of symmetric switch blocks and Lemma 10, we have the following lemmas.

Lemma 11. *The N-sided symmetric switch blocks of size one are universal.*

Proof. For an RRV $\vec{n} \in R(N, 1)$, let the corresponding RRG of $\vec{n}$ be G. Since there is only one terminal on each side, the routing requirements in $\vec{n}$ correspond to pairs of vertices in G, where all vertices are distinct. Let the pairs of vertices be $(v_{i_1}, v_{i_2}), (v_{i_3}, v_{i_4}), \ldots, (v_{i_{2k-1}}, v_{i_{2k}})$, where $2k \leq N$. By the definition of symmetric switch blocks, the routing requirements of $\vec{n}$ can be successfully routed by using k switches

$$(t_{s_{i_1},1}, t_{s_{i_2},1}), (t_{s_{i_3},1}, t_{s_{i_4},1}), \ldots, (t_{s_{i_{2k-1}},1}, t_{s_{i_{2k}},1}).$$

Therefore, the N-sided symmetric switch blocks of size one are universal. □

Lemma 12. *The N-sided symmetric switch blocks of size two are universal.*

Proof. By Lemma 6, there are two types of MRRVs in $\Gamma(N, 2)$: One is the complete MRRV; the other is the degenerate complete MRRV. For a complete MRRV $\vec{\gamma} \in \Gamma(N, 2)$, the corresponding RRG is a 2-RRG. By Lemmas 8 and 10, $\vec{\gamma}$ is routable on the symmetric switch block $M_{N,2}$. For a degenerate complete MRRV $\vec{\gamma} \in \Gamma(N, 2)$, the degree of each vertex in its corresponding RRG G is two except one vertex, say v_i, with the degree zero. Then, $G' = G \setminus \{v_i\}$ is a 2-RRG. Let the MRRV corresponding to G' be $\vec{\gamma}' \in \Gamma(N - 1, 2)$. For a symmetric switch block $M_{N,2}$, by Lemma 2, $M_{N-1,2} = M_{N,2} \setminus L_{s_i,2}$ is still a symmetric switch block. By Lemmas 8 and 10, $\vec{\gamma}'$ is routable on $M_{N-1,2}$ and must be routable on $M_{N,2}$. Since the degree of v_i is zero, i.e., there is no routing requirement on side s_i, $\vec{\gamma}'$ and $\vec{\gamma}$ have the same routing requirements. $\vec{\gamma}$ is thus routable on $M_{N,2}$. The degenerate complete MRRV is thus routable on the symmetric switch block of size two. Therefore, by Lemma 5, the symmetric switch blocks of size two are universal. □

Based on Lemmas 11 and 12 and the decomposition property of symmetric switch blocks, we have the following theorem by induction.

Theorem 1. *The N-sided symmetric switch blocks of size W are universal.*

Proof. We can prove this lemma by induction. By Lemma 12, the symmetric switch block $M_{N,2}$ is universal. Assume that the symmetric switch block $M_{N,W}$ is universal for $W < k$, where $k \geq 3$. Now, we shall prove the universality of the symmetric switch block $M_{N,k}$. By Lemma 6, there are two types of MRRVs in $\Gamma(N, W)$: One is $0 \leq \phi_{unused} \leq 1$; the other is $2 \leq \phi_{unused} \leq W$. By Lemma 1, we can decompose $M_{N,k}$ into two universal switch blocks $M_{N,2}$ and $M_{N,k-2}$. For the first type MRRV $\vec{\gamma} \in \Gamma(N, k)$, let the DRV of $\vec{\gamma}$ be $\vec{d}$, where $\vec{d} \leq \vec{D}_{N,k}$. Since $\phi_{unused} \leq 1$, the degree of each vertex in the RRG of $\vec{\gamma}$ is larger than 2. By Lemmas 9 and 10, there exists a DRRCS Δ $(\sum |V(C_i)| = N)$ in which the corresponding RRV for Δ is routable on $M_{N,2}$ and can first be routed on $M_{N,2}$. Then, $\vec{\gamma}$ is reduced to $\vec{\gamma}'$, where the DRV of $\vec{\gamma}'$ is $\vec{d}' = \vec{d} - \vec{D}_{N,2}$. It is clear that $\vec{d}' \leq \vec{D}_{N,k-2}$, i.e., $\vec{\gamma}'$ satisfies the dimension constraint of size $k - 2$. Then, $\vec{\gamma}'$ is routable on $M_{N,k-2}$. Hence, the first type MRRVs are routable on $M_{N,k}$.

For the other type MRRV $\vec{\gamma} \in \Gamma(N, k)$, let the DRV of $\vec{\gamma}$ be $\vec{d}$, where $\vec{d} \leq \vec{D}_{N,k}$. Suppose the side with unused terminals is s_j. Let the RRG of $\vec{\gamma}$ be G and $G' = G \setminus \{v_j\}$. Since $T_{unused} \geq 2$, the number of terminals on each side connected to side s_j is at most $k - 2$. Hence, the degree of each vertex in G' is larger than 2. By Lemmas 9 and 10, there exists a DRRCS Δ $(\sum |V(C_i)| = N - 1)$ in which the corresponding RRV for Δ is routable on $M_{N-1,2}$ and must be routable on $M_{N,2}$. Then, the RRV corresponding to Δ can first be routed on $M_{N,2}$ and $\vec{\gamma}$ is reduced to $\vec{\gamma}'$, where the DRV of $\vec{\gamma}'$ is $\vec{d}'$ and $d_i' = d_i - 2$ for $1 \leq i \leq k$ except $d_{s_j}' = d_{s_j}$. Since $\phi_{unused} \geq 2$, $d_{s_j}' = d_{s_j} \leq k - 2$. It is clear that $\vec{d}' \leq \vec{D}_{N,k-2}$, i.e., $\vec{\gamma}'$ satisfies the dimension constraint of size $k - 2$. Then, $\vec{\gamma}'$ is routable on $M_{N,k-2}$. Hence, the other type MRRVs are routable on $M_{N,k}$. By Lemma 5, the symmetric switch block $M_{N,k}$ is universal, implying that the N-sided symmetric switch blocks of size W are universal. □

We have shown the universality of our generic symmetric switch blocks in Theorem 1, i.e., our generic symmetric switch blocks provide the maximum routing capacity. Next, we shall show that our generic symmetric switch blocks use the minimum number of switches. We have the following theorem.

Theorem 2. *No NSB of size W with less than $\binom{N}{2}W$ switches can be universal.*

Proof. By Definition 1, an RRV with only one nonzero component W, such as

$$(W, 0, \ldots, 0), (0, W, 0, \ldots, 0), \ldots, (0, 0, \ldots, 0, W),$$

is routable on a universal NSB. Hence, it needs at least W noninteracting switches for each type of connections to construct a universal NSB. Since there are $\binom{N}{2}$ types of
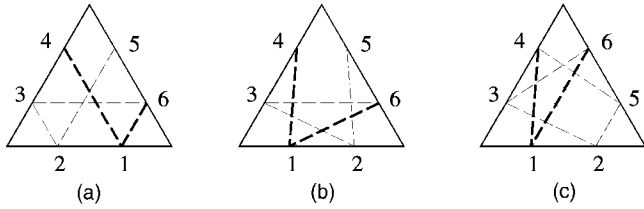
Fig. 13. (a) A 3-sided symmetric switch block. (b), (c) Two 3-sided isomorphic switch blocks of (a).

connections in an NSB, the least number of switches needed to construct a universal NSB is $\binom{N}{2}W$. ☐

As mentioned in Section 3.1, the total number of switches used in an N-sided symmetric switch block of size W is $\binom{N}{2}W$. Thus, our symmetric switch blocks are the "cheapest" universal ones, i.e., it uses the minimum number of switches to provide the maximum routing capacity. Note that the $\binom{N}{2}W$ requirement is quite small compared to a fully connected switch block which has $\binom{N}{2}W^2$ switches.

3.3 Isomorphism of Switch Blocks

In order to identify not only a single, but also a whole class of universal switch blocks, we apply the terminology *isomorphism* used in [2]. Here, we give its definition.

Definition 4. *Two NSBs* $M_{N,W}(T,S)$ *and* $M'_{N,W}(T',S')$ *are isomorphic if there exists a bijection* $f \colon T \to T'$ *such that* $(t_{m,n}, t_{u,v}) \in S$ *iff* $(f(t_{m,n}), f(t_{u,v})) \in S'$ *and for any two terminals* $t_{m,n}, t_{u,v} \in T_p$ *iff* $f(t_{m,n}), f(t_{u,v}) \in T_q$, *where* $p, q = 1, 2, \ldots, N$.

In other words, $M_{N,W}(T,S)$ and $M'_{N,W}(T',S')$ are isomorphic if we can relabel the terminals of M to be the terminals of M', maintaining the corresponding switches in M and M' and, for terminals on the same side of M, their corresponding terminals are also on the same side of M'. Fig. 13 shows three 3-sided isomorphic switch blocks on which their corresponding terminals are indicated by the same number. For any two isomorphic switch blocks, we have the following theorem.

Theorem 3. *Any two isomorphic switch blocks have the same routing capacity.*

Proof. If $M_{N,W}(T,S)$ and $M'_{N,W}(T',S')$ are isomorphic, we can relabel the types of connections and have the same switch-connection configuration with respect to each type (see Fig. 14 for illustration). Let $\vec{n}'$ be a permutation of $\vec{n}$ so that $\vec{n}$ and $\vec{n}'$ correspond to the original and new definitions of the types of connections, respectively. It is obvious that $\vec{n}$ is routable on $M_{N,W}(T,S)$ if and only if $\vec{n}'$ is routable on $M'_{N,W}(T',S')$; thus, $M_{N,W}(T,S)$ and $M'_{N,W}(T',S')$ have the same routing capacity. ☐

Corollary 3.1. *For any two isomorphic switch blocks* $M_{N,W}(T,S)$ *and* $M'_{N,W}(T',S')$, $M_{N,W}(T,S)$ *is universal iff* $M'_{N,W}(T',S')$ *is universal.*

By Corollary 3.1, we can obtain a whole class of universal switch blocks by performing isomorphism operations on a symmetric switch block.
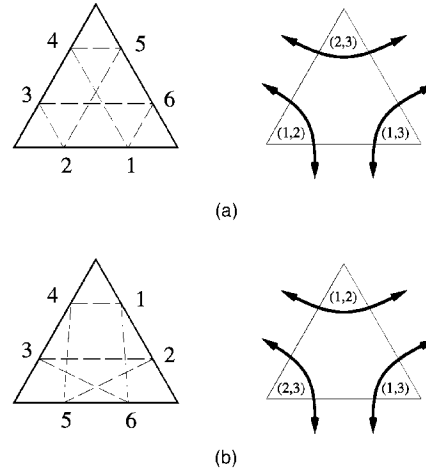

(a)

(b)

Fig. 14. (a) A switch block and the original type definition. (b) An isomorphic switch block of (a) and its new type definition.

4 GRAPH MODELING FOR DETAILED ROUTING

To explore the area performance of switch blocks, we develop a detailed router for 4-HFPGAs. We first model a 4-HFPGA as a graph and apply a graph-search technique to 4-HFPGA routing. For the purpose of easier illustrations, in
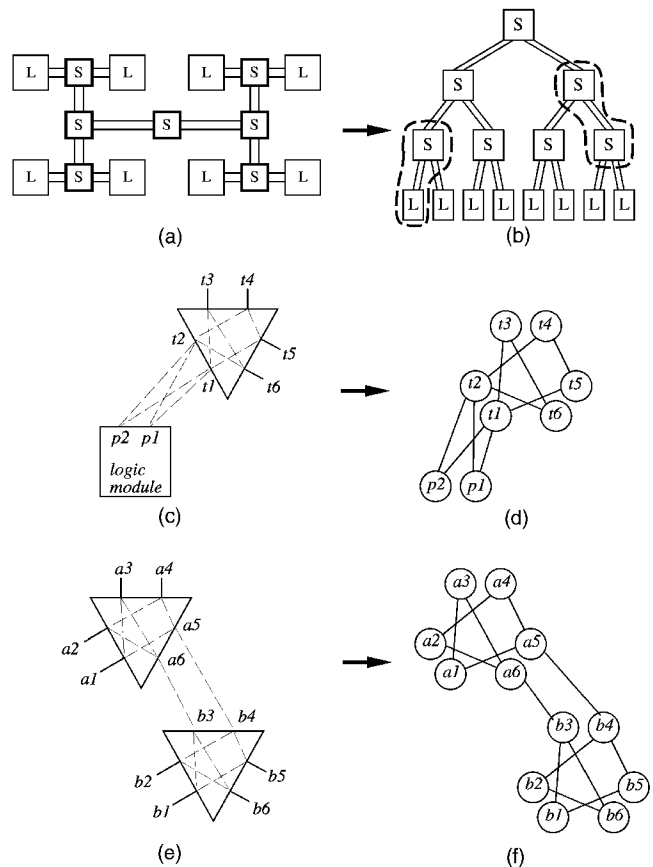

Fig. 15. The graph modeling. (a) A 2-HFPGA architecture. (b) The tree corresponding to (a). (c) The switch connections inside a switch block and that between a logic block and a switch block. (d) The graph corresponding to (c). (e) The switch connections inside switch blocks and that between two switch blocks. (f) The graph corresponding to (e).

TABLE 1
Minimum Number of Tracks Needed for Routing Completion in Situation 1)

p	Net order	Architecture of switch block	\multicolumn{9}{c}{Minimum number of tracks for 100% routing}
			100	200	300	400	500	600	700	800	total
1.0	-	Universal	8	15	19	23	29	34	34	37	199
	-	Clique	8	15	20	25	31	35	37	38	209
0.9	Original net order	Universal	7	13	21	23	27	32	38	41	202
		Clique	7	13	21	25	27	34	40	43	210
	Longest net first	Universal	7	12	20	23	27	34	39	42	204
		Clique	7	14	20	24	28	34	41	44	212
	Shortest net first	Universal	7	14	20	22	27	31	38	42	201
		Clique	8	15	21	23	27	33	41	42	210
0.8	Original net order	Universal	9	13	21	24	30	33	38	38	206
		Clique	10	13	22	24	32	36	39	42	218
	Longest net first	Universal	9	13	21	24	29	32	36	40	204
		Clique	9	14	23	26	30	34	40	42	217
	Shortest net first	Universal	9	14	26	28	31	39	41	47	235
		Clique	10	14	27	30	33	42	43	49	248
0.7	Original net order	Universal	8	15	20	24	28	33	38	43	209
		Clique	9	15	22	24	31	37	39	46	223
	Longest net first	Universal	8	14	20	22	29	32	37	43	205
		Clique	8	15	21	24	31	36	39	46	220
	Shortest net first	Universal	10	21	26	31	39	44	50	>50	>271
		Clique	11	21	26	31	40	49	>50	>50	>278

this section, we use a 2-HFPGA as an example to demonstrate the graph modeling.

For a 2-HFPGA (see Fig. 15a), we first transform it into a tree architecture (see Fig. 15b). There are three situations for switch connections: 1) the switch connections inside a switch block (see Fig. 15a and Fig. 15c), 2) those between a logic block and a switch block (see Fig. 15c), and 3) those between two switch blocks (see Fig. 15e). We use a vertex to represent a switch-block terminal or a logic-block pin. If there is a switch connected between two terminals in a switch block or between two switch blocks, or between a logic-block pin and a terminal, we introduce an edge between the two corresponding vertices.

For Situation 1), Fig. 15c shows the switch connections inside a switch block of size two. We introduce a vertex for each terminal ($t_1, t_2, \ldots, t_6$ in Fig. 15d) and an edge for each corresponding pair of terminals ((t_1, t_3), (t_1, t_5), ..., (t_4, t_5) in Fig. 15d). For Situation 2), we introduce an edge for a connection between a logic-block pin and a terminal on the adjacent side of a switch block. Fig. 15c shows the modeling for the case when the switch connections between a logic block and a switch block are fully connected. We introduce two vertices p_1 and p_2 for the two pins of the logic block. Since both p_1 and p_2 can be connected to t_1 and t_2, we construct the edges (p_1, t_1), (p_1, t_2), (p_2, t_1), and (p_2, t_2) in Fig. 15d). For Situation 3), we introduce an edge for a connection between two adjacent switch blocks. Fig. 15e shows the modeling for the case when each terminal on a switch block connects to only one terminal on the adjacent side of another switch block. We construct two edges (a_5, b_4) and (a_6, b_3) in Fig. 15f).

Based on the graph modeling, we may formulate the routing problem as finding a set of disjoint trees, one tree for a net and each tree connecting all terminals of a net. Any

graph search-based algorithm such as a maze router can be used for detailed routing.

5 EXPERIMENTS AND RESULTS

5.1 Area Performance

In our experiments, we implemented a maze router based on the graph modeling mentioned in the preceding section to explore the effects of switch-block architectures on routing. The router was written in the C language and ran on a SUN Ultra workstation.

We generated connections with lengths based on the *geometric distribution function* because it closely relates to most industrial circuit configurations (e.g., the benchmark circuits in [10], [13], [11]). (Note that no industrial benchmarks for HFPGAs are available.) The geometric distribution function is given as follows:

$$P(p, k) = p(1 - p)^{k-1}, \qquad 0 \le p \le 1, \quad k = 1, 2, \ldots,$$

where k is related to connection length and p is a user specified parameter. In the experiments, we randomly generated a set of benchmark circuits on a three-level 4-HFPGA (64 logic blocks) based on the geometric distribution function with p = 1.0, 0.9, 0.8, and 0.7. The 5-sided switch blocks used in the experiments were the universal and clique-based (Xilinx-like) ones with the same number of switches. We assume that all pins of a logic block can be connected to any terminals on the adjacent side of the switch block and each terminal on a switch block connects to only one terminal on the adjacent side of another switch block.

The quality of a switch block was evaluated by the area performance of the detailed router. We determined the minimum number of tracks (W) required for 100 percent routing completion for each circuit, using the two kinds of

TABLE 2
Minimum Number of Tracks Needed for Routing Completion in Situation 2)

p	Net order	Architecture of switch block	Minimum number of tracks for 100% routing								
			100	200	300	400	500	600	700	800	total
1.0	-	Universal	6	14	19	22	30	33	34	37	195
	-	Clique	6	14	18	22	30	33	34	36	193
0.9	Original net order	Universal	7	12	19	21	26	31	37	40	193
		Clique	7	12	19	22	27	32	38	41	198
	Longest net first	Universal	7	12	19	21	26	31	37	40	193
		Clique	7	12	19	22	27	32	38	41	198
	Shortest net first	Universal	6	13	20	23	27	30	37	40	198
		Clique	6	13	20	22	27	30	38	39	197
0.8	Original net order	Universal	9	13	20	23	29	32	35	37	198
		Clique	10	13	20	25	30	32	36	39	205
	Longest net first	Universal	9	13	20	23	29	32	36	38	200
		Clique	10	13	20	25	32	32	39	39	210
	Shortest net first	Universal	10	14	22	29	32	36	42	47	232
		Clique	10	13	22	28	33	38	43	48	235
0.7	Original net order	Universal	9	14	18	21	27	31	35	41	196
		Clique	9	14	19	23	30	33	38	44	210
	Longest net first	Universal	9	14	18	21	28	31	36	41	195
		Clique	9	14	19	23	30	33	38	44	210
	Shortest net first	Universal	10	20	27	31	40	50	>50	>50	>278
		Clique	9	21	25	34	41	>50	>50	>50	>280

switch blocks. Because net ordering often affects the performance of a maze router, we routed the benchmark circuits by using the following three net-ordering schemes to avoid possible biases: 1) original net order in the benchmark circuits, 2) longest net first, and 3) shortest net first. Also, since our main goal is to make fair comparisons for various switch-block architectures, no rip-up and reroute phase was incorporated in the maze router.

In our maze router, we used a shortest-path algorithm to find a routing path for a net. During the process of the shortest-path algorithm, the algorithm chose an unmarked vertex with minimum-cost value for further processing. However, there may be more than one vertex with the minimum-cost value. For the set M_c of the vertices with the minimum cost, we can randomly choose one vertex from the set M_c or just choose the first vertex in the set.

In the experiments, we also considered the constraint for the number of switches inside a switch block used by a net. With the switch-number constraint, a net can use exactly one switch when it passes through a switch block. Without

TABLE 3
Minimum Number of Tracks Needed for Routing Completion in Situation 3)

| p	Net order	Architecture of switch block	Minimum number of tracks for 100% routing								
			100	200	300	400	500	600	700	800	total
1.0	-	Universal	6	14	18	22	28	33	34	36	191
	-	Clique	6	14	18	22	30	33	34	36	193
0.9	Original net order	Universal	6	12	19	21	26	31	36	40	191
		Clique	7	12	19	22	27	32	38	41	198
	Longest net first	Universal	6	12	19	21	26	31	36	40	191
		Clique	7	12	19	22	27	32	38	41	198
	Shortest net first	Universal	6	12	19	22	26	30	36	40	191
		Clique	6	12	19	21	26	29	38	39	190
0.8	Original net order	Universal	9	12	19	23	29	32	35	37	196
		Clique	10	13	20	25	32	32	39	39	210
	Longest net first	Universal	9	12	19	23	29	32	36	37	197
		Clique	10	13	20	25	32	32	39	39	210
	Shortest net first	Universal	9	13	21	29	27	33	42	47	221
		Clique	10	13	22	28	33	38	43	48	235
0.7	Original net order	Universal	9	14	18	21	27	31	35	40	195
		Clique	9	14	19	23	30	33	38	44	210
	Longest net first	Universal	9	14	18	21	28	31	36	40	197
		Clique	9	14	19	23	30	33	38	44	210
	Shortest net first	Universal	10	18	27	30	40	50	>50	>50	>265
		Clique	9	21	25	34	41	>50	>50	>50	>280
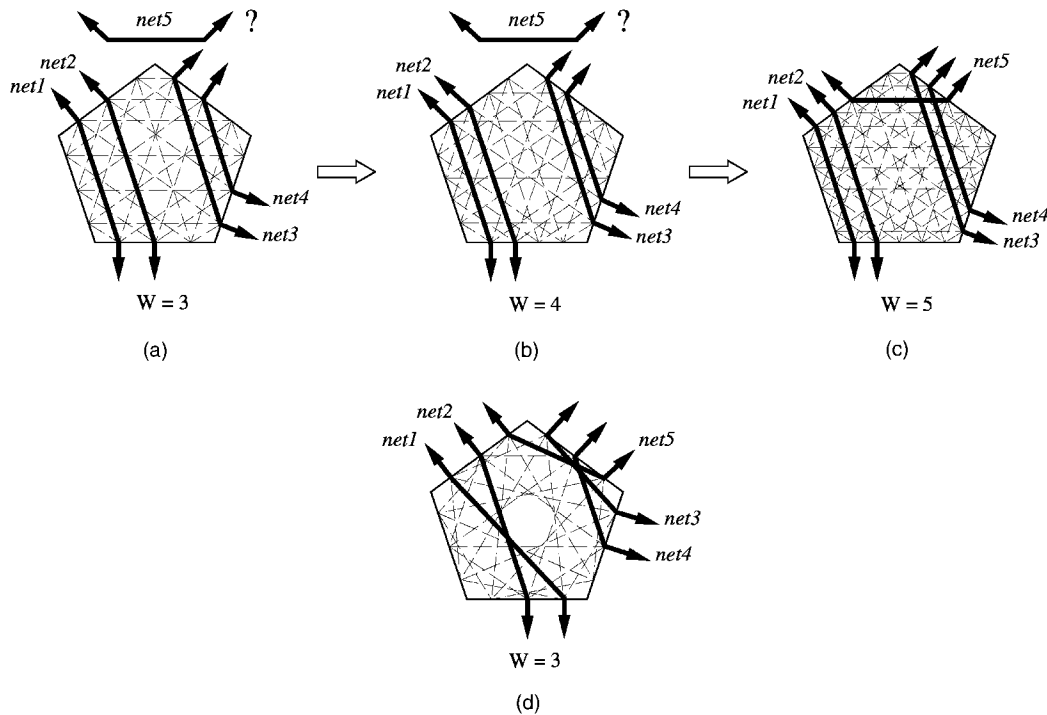
Fig. 16. The example of the blocking anomaly in universal switch-block architectures. Suppose we are to route $net1 - net5$ *sequentially* after $net1 - net4$ have been routed. (a) $net5$ cannot be routed on a symmetric switch block of $W = 3$. (b) $net5$ still cannot be routed after increasing W by one. (c) $net5$ finally can be routed when W is increased to five. (d) The blocking anomaly does not affect the routing in clique-based switch-block architectures.

the switch-number constraint, the number of switches used by a net for passing through a switch block is not limited. Thus, our experiments are classified into three categories by the following situations:

1. Randomly extract a minimum-cost vertex and route a net with the switch-number constraint.
2. Extract the first minimum-cost vertex and route a net with the switch-number constraint.
3. Extract the first minimum-cost vertex and route a net without the switch-number constraint.

The experimental results of Situations 1), 2), and 3) are shown in Tables 1, 2, and 3, respectively. The results show that, no matter in which situation and with which net-ordering scheme, our universal switch blocks usually outperform the clique-based (Xilinx-like) ones in the chip-level area performance.

5.2 Interactions between Switch-Block Architectures and Routing

We explore the interactions between our universal switch-block architectures and routing. We are interested, however, in the case of Situation 2), with $p = 1.0$, in which the total number of tracks needed for the universal switch blocks and the clique-based ones are 195 and 193, respectively—a 1 percent increase in area for the universal switch blocks. In the situation, the router always chose the first minimum-cost vertex and this selection may block subsequent routing nets. Fig. 16a, Fig. 16b, and Fig. 16c illustrate the situation. Fig. 16a shows a 5-sided universal switch block of size three. Suppose we have routed $net1 -$

$net4$ and are routing $net5$. Satisfying the dimension constraint, $net5$ still cannot be routed on the 5-sided switch block. (Note that the universality concept is for routing nets *simultaneously*, but, in this case, we routed the nets *sequentially*.) If we increase W by one and reroute all nets *sequentially*, $net5$ still cannot be routed on the 5-sided switch block of size four (see Fig. 16b). This is because the router always chose the first minimum-cost vertex in the shortest-path algorithm; thus, the terminals on one side would be selected by nets based on the physical order of the terminals. This phenomenon is called *blocking anomaly*. The blocking anomaly introduced by always choosing the first minimum-cost vertex is not favorable to our universal switch blocks, but does not affect the routing in clique-based switch blocks (see Fig. 16d). Therefore, the 1 percent increase in area is caused by the biased router, but not the quality of switch-block architectures. The blocking anomaly is also associated with the switch-number constraint for routing *on a switch block*. In our experiments for Situation 3), when a net was routed without the switch-number constraint, the blocking anomaly did not affect the routing on our universal switch blocks. It is thus significant to consider the interactions between architectures and CAD [11], [15]—even with a best architecture, unsuitable (or biased) routers might offset the advantages of the architecture. Based on our studies, we have four suggestions to avoid the blocking anomaly in our universal switch-block architectures: 1) do not *fix* the scenario for selecting terminals for routing based on their physical order, 2) consider an appropriate switch-number constraint, 3) increase the switch connections between two adjacent

switch blocks to facilitate track permutations, and 4) use a concurrent (nonsequential) router such as hierarchical routers [9], [7] to route all nets simultaneously.

6 CONCLUSIONS

We have proposed algorithms to construct a class of generic universal switch blocks. Our universal switch blocks not only have the maximum routing capacities, but also use the minimum numbers of switches. Further, the decomposition property of a universal switch block provides a key insight into its layout implementation with a smaller silicon area. We also developed a maze router for 4-HFPGAs for experimentation. Experimental results show that our universal switch blocks usually outperform the clique-based (Xilinx-like) ones in the chip-level area performance. We further explored the interactions between the universal switch-block architectures and routing. It is significant to consider architectures and CAD simultaneously—even with a best architecture, unsuitable (or biased) routers might offset the advantages of the architecture.

ACKNOWLEDGMENTS

REFERENCES

[1] S.D. Brown, R.J. Francis, J. Rose, and Z.G. Vranesic, *Field-Programmable Gate Arrays*. Boston: Kluwer Academic, 1992.
[2] Y.-W. Chang, D.F. Wong, and C.K. Wong, "Universal Switch Modules for FPGA Design," *ACM Trans. Design Automation of Electronic Systems*, vol. 1, no. 1, pp. 80-101, Jan. 1996.
[3] R.I. Greenberg, "The Fat-Pyramid and Universal Parallel Computation Independent of Wire Delay," *IEEE Trans. Computers*, vol. 43, no. 12, pp. 1,358-1,364, Dec. 1994.
[4] S. Hauck, G. Borriello, and C. Ebeling, "Mesh Routing Topologies for FPGA Arrays," *Proc. ACM Int'l Workshop FPGAs*, pp. 1-10, 1994.
[5] K. Kawana, H. Keida, M. Sakamoto, K. Shibata, and I. Moriyama, "An Efficient Logic Block Interconnect Architecture for User-Reprogrammable Gate Array," *Proc. IEEE Custom Integrated Circuits Conf.*, pp. 313.1-31.3.4, 1990.
[6] C.H. Leiserson, "Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing," *IEEE Trans. Computers*, vol. 34, no. 10, pp. 892-901, Oct. 1985.
[7] U. Lauther, "Top-Down Hierarchical Global Routing for Channelless Gate Arrays Based on Linear Assignment," *Proc. IFIP VLSI 87*, pp. 109-120, 1987.
[8] Y.T. Lai and P.T. Wang, "Hierarchical Interconnection Structures for Field Programmable Gate Arrays," *IEEE Trans. VLSI Systems*, pp. 186-196, 1997.
[9] M. Marek-Sadowska, "Route Planner for Custom Chip Design," *Proc. IEEE/ACM Int'l Conf. Computer-Aided Design*, pp. 246-249, 1986.
[10] J. Rose and S. Brown, "Flexibility of Interconnection Structures for Field-Programmable Gate Arrays," *IEEE J. Solid State Circuits*, vol. 26, no. 3, pp. 277-282, 1991.
[11] B. Tseng, J. Rose, and S. Brown, "Improving FPGA Routing Architectures Using Architecture and CAD Interactions," *Proc. IEEE Conf. Computer Design*, pp. 99-104, 1992.
[12] N. Togawa, M. Sato, and T. Ohtsuki, "Maple: A Simultaneous Technology Mapping, Placement, and Global Routing Algorithm for Field-Programmable Gate Arrays," *Proc. IEEE/ACM Int'l Conf. Computer-Aided Design*, pp. 156-163, 1994.
[13] S. Thakur, Y.W. Chang, D.F. Wong, and S. Muthukrishnan, "Algorithms for an FPGA Switch Module Routing Problem with Application to Global Routing," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Jan. 1997.
[14] W. Tsu, "A Comparison of Universal and Xilinx Switches," CS294-7 Project Report, Univ. of California-Berkeley, Spring 1997.
[15] Y.-L. Wu, S. Tsukiyama, and M. Marek-Sadowska, "Graph Based Analysis of 2-D FPGA Routing," *IEEE Trans. Computer-Aided Design*, vol. 15, no. 1, pp. 33-44, Jan. 1992.
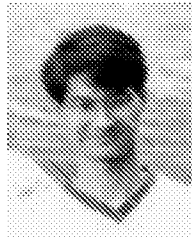
Michael Shyu received the BS degree in computer science and information engineering from National Chao Tung University, Hsinchu, Taiwan, Republic of China, in 1995, and the MS degree in computer and information science from National Chao Tung University, Hsinchu, Taiwan, Republic of China, in 1999. His research interests include design automation and optimization for VLSI design.
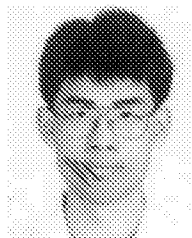
Guang-Ming Wu received the BS degree in information and computer engineering from Chung Yuan Christian University, Chun-Li, Taiwan, Republic of China, in 1990, and the MS degree in computer science from National Chao Tung University, Hsinchu, Taiwan, Republic of China, in 1994. He is currently working toward the PhD degree in computer science from National Chao Tung University. His research interests include design automation and optimization for VLSI design.

Yu-Dong Chang received the BS degree from the Department of Computer Science at National Tsing Hua University Hsinchu, Taiwan Republic of China, in 1995, and the MS degree from the Department of Computer and Information Science at National Chao Tung University, Hsinchu, Taiwan, Republic of China, in 1998. His research interests include design automation and optimization for VLSI design.

Yao-Wen Chang (S'94-M'96) received the BS degree in computer science and information engineering from National Taiwan University in 1988, and the MS and PhD degrees in computer science from the University of Texas at Austin in 1993 and 1996, respectively.
He was with the IBM T.J. Watson Research Center, Yorktown Heights, New York, in the VLSI group during the summer of 1994. He is currently an associate professor in the Department of Computer and Information Science at National Chiao Tung University, Hsinchu, Taiwan. His research interests lie in design automation, architectures, and systems for VLSI and combinatorial optimization.
Dr. Chang received Best Paper Award at the 1995 IEEE International Conference on Computer Design (ICCD-95) for his work on FPGA routing and MS Thesis Supervision Award from the Institute of Electrical Engineering, Taiwan in 1998. He is a member of the IEEE, the IEEE Circuits and Systems Society, the ACM, and ACM/SIGDA.

Folded Fat H-Tree: an interconnection topology for Dynamically Reconfigurable Processor Array

Yutaka Yamada[1], Hideharu Amano[1], Michihiro Koibuchi[1], Akiya Jouraku[1],
Kenichiro Anjo[1], and Katsunobu Nishimura[2]

[1] Department of Information and Computer Science, Keio University
3-14-1, Hiyoshi Yokohama 223-8522, Japan
wasmii@am.ics.keio.ac.jp
[2] Faculty of Commerce and Ecomonics, Chiba University of Commerce
1-3-1 Kohnodai, Ichikawa, Chiba 272-8512, Japan

Abstract. Fat H-Tree is a novel on-chip network topology for a dynamic reconfigurable processor array. It includes both fat tree and torus structure, and suitable to map tasks in a stream processing. For on-chip implementation, folding layout is also proposed. Evaluation results show that Fat H-Tree reduces the distance of H-Tree from 13% to 55%, and stretches the throughput almost three times.

1   Introduction

A Dynamically Reconfigurable Processor Array (DRPA) is a reconfigurable device consisting of coarse grain processing elements which can change their structure and connection quickly. It has been received attention as a cost-effective solution for streaming and network processing, and commercial chips are recently available[1][2][3][4]. Some of them use a flat array structure[2], while a cluster structure is introduced in other systems[1][4].

Processing elements in such an array structure tend to be connected with statically programmable switches similar to those used in common FPGAs. On the contrary, for inter-cluster connection, a Network on Chip (NoC) based on a simple packet switching is advantageous because of its flexibility and cost effective use of wires. Quicksilver's ACM[3] equips a simple packet switching network called Matrix Interconnection Network (MIN) for connecting reconfigurable units. We also proposed a simple packet switching network based on local labels called Black-Bus[6] for connecting reconfigurable processing arrays.

Since the size of array is not so large in the current state of technology, simple network topologies including H-Tree or two dimensional torus is utilized for connecting such arrays. However, like interconnection networks used in multiprocessors, network topology which distributes traffic and avoids packet congestion will be a key of such interconnection networks.

In this paper, we propose a novel interconnection network called Folded Fat H-Tree for DRPAs. It includes a fat tree[11] and two dimensional torus structure, and is suited to traffic distribution of stream processing. By folding the mesh structure for on-chip layout, long wires crossing the whole die can be avoided.

2   Interconnection Networks for DRPAs

2.1   The structure of SARA

Although interconnection networks proposed here can be used for various architectures, the target reconfigurable architecture called SARA(Stream processing Architecture with Reconfigurable processor Array) is introduced first. As shown in Figure 1, SARA is consisting of a cluster of DRPAs and embedded RISC CPU. The unit of the DRPA that is called "Tile" models a unit array in NEC's DRP(Dynamically Reconfigurable Processor)[1]. In SARA, they are connected with a simple packet switching network[6].
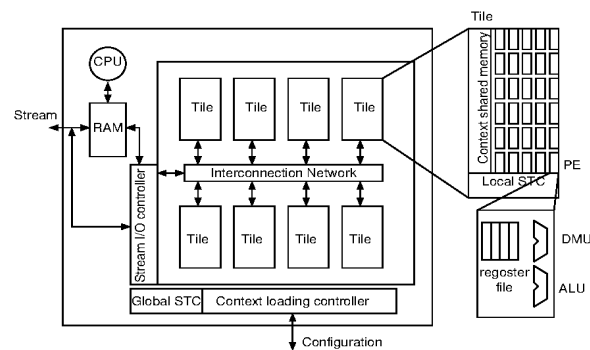
Fig. 1. Diagram of the target architecture SARA

The primitive modules of a tile are processing elements(PEs), State Transition Controller (STC), 2-ported distributed memory modules. There are 8×8 PEs located in one tile. It has an 8-bit ALU, an 8-bit DMU, an 8-bit$\times$16-word register file, and an 8-bit flip-flop. Those units are connected by programmable switches and wires in the similar manner to common FPGAs. These bitwidths are ranging from 8B to 18B according to the location. PE has 16-depth instruction memories and supports multiple context operation. Its instruction pointer is delivered from STC. STC is a programmable sequencer in which a certain FSM (Finite State Machine) can be stored.

SARA equips tens of tiles which are connected with a simple packet switching network called Black-Bus[6]. In Black-Bus, a local identifier (ID) is attached to each raw data as routing information. Unlike the traditional packet transfer, the local ID is transferred on dedicated wires attached to data lines to remove complicated packet generation procedure in a tile. Only a small-sized local ID is required to specify routing tags to the destination, and intermediate routers change it to solve local ID conflict between paths on a physical channel.

Since main target application of SARA is stream processing including media and network processing, the stream I/O controller is provided for high speed stream transfer. The virtual hardware support mechanism and context cache are provided for a large target application or multi-target processing.

2.2   Stream processing of SARA

In most stream processing, a series of processing is performed to a certain amount of data. A unit of such processing is called the "Task"[12]. Figure 2 shows task diagram of JPEG2000 decoding. In the processing, each task can be mapped into each tile in SARA, and is performed in the pipelined manner. In this case, the communication between tiles becomes almost linear. However, the framed part called EBCOT(Embedded Block Coding with Optimal Truncation) requires a high computation power and bottlenecks the whole stream flow if each processing in EBCOT is assigned into a tile. For equalizing the stream flow, the processing of EBCOT must be distributed into several tiles and executed in parallel. In this case, the communication between tiles includes stream fork and join.

```
┌──────────────┐   ┌──────────────┐   ┌──────────────────────────────────┐   ┌──────────────┐   ┌──────────────┐
│ DC transform │   │   Wavelet    │   │ ┌────────────┐    ┌────────────┐ │   │  Arithmatic  │   │Code sequence │
│Color transform│→ │  transform   │ → │ │Quantization│ →  │ Coefficient│ │ → │   Encoding   │ → │   Control    │ →
│              │   │              │   │ │            │    │  Modeling  │ │   │              │   │              │
└──────────────┘   └──────────────┘   │ └────────────┘    └────────────┘ │   └──────────────┘   └──────────────┘
                                       └──────────────────────────────────┘
                                                      EBCOT
```
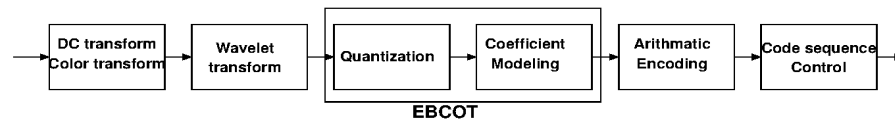
Fig. 2. A task graph for JPEG2000 decoding

On the other hand, connection topologies of recent DRPAs are classified into follows:

- Linear (Or strongly directed two-dimensional) : PipeRench[5] and IPFlex's DAP/DNA[2].
- Two-dimensional mesh: NEC's DRP[1] and inner-unit connection of Pact Xpp[4].
- Tree: Quicksilver's ACM[3] and inter-unit connection of Pact Xpp[4].

Linear and directed two dimensional mesh structures are advantageous to map a linear connection pattern of stream processing, while fork/join communication pattern fits to the tree-structures. Since a simple tree-structure tends to introduce traffic congestion around the root, fat-tree structure is advantageous. DeHon proposed Butterfly Fat Tree[8] which includes a fat-tree structure and its efficient layout on the chip. However, it does not include directed mesh structure which fits to the basic communication pattern of stream processing.

A large number of interconnection networks which have both tree and grid structures have been researched for large scale parallel machines. For example, Recursive Diagonal Torus [9] is an extended hierarchical torus which also has properties of the tree. However, they are designed for large scale machines and its connection structure tend to be complicated. The layout on the chip is also difficult.

3   Fat H-Tree

3.1   Extension of H-Tree

Figure 3 shows that a typical surface-layout of the tree-structure. This structure, called H-Tree, is used in Quicksilver's ACM[3] and Caltech SCORE[7]. Here, squares in the figure represents a switch, and number in a switch represents its rank. A rank-0 switch

is a network interface which connects one or a few tiles to upper rank switches. Although H-Tree is considered to be placed on a square die of an IC chip, the topology is equivalent to a simple tree. Therefore, it cannot be free from common weak points of a tree: links or switches around the root are frequently congested, and there is a long distance between two rank-0 switches placed on the tree boundary.
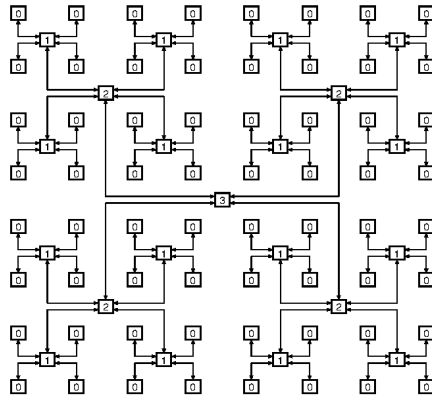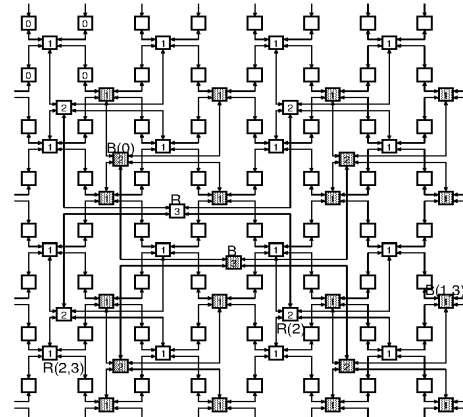

Fig. 3. H-Tree

Fig. 4. Fat H-Tree

In order to make up these problems, we introduce two tree structures mutually for upper ranks so as to form a torus interconnection as shown in Figure 4. That is, two trees consisting of white switches and black switches are formed. Here, the former tree is called "red tree", while the latter is called "black tree", and the whole structure is called Fat H-Tree. Fat H-Tree includes two tree structures and a grid structure.

In Figure 4, the number attached into each switch (1,2 or 3) represents its rank in the tree. These numbers are omitted for most of rank-0 switches which are connected directly with tiles. Two tree structures can be located separately near to each edge of the die, upper rank switches can be used for input/output streams.

Although the red tree and black tree are independent here, fat tree structures[11] can be introduced also further upper ranks of red/black tree. Although such additional fat tree structure enhances the transfer bandwidth, the total structure becomes complicated. So, we only treat the simple structure with independent red and black trees.

3.2   Folded Fat H-Tree

By providing a torus structure, Fat H-Tree introduces a layout problem. That is, a lot of long feed-back links laid across the chip are required. Although they are omitted in Figure 4, rightmost/top links must be connected with the leftmost/bottom switches.

In order to cope with this problem, the torus structure must be folded. As shown in Figure 5, the order of nodes is changed so that every link is connected to the next neighbor node[12][9].

By applying the folding method is applied to the rank-0 switches for both x and y direction, Folded Fat H-Tree is formed.

(a) before   (b) after

Fig. 5. Folding of a torus

Although Folded Fat H-Tree is a layout suitable to be implemented on the chip, it is topologically equivalent to the Fat H-Tree. In order to avoid complicated figures, Fat H-Tree is discussed in the rest of this paper. Folded Fat H-Tree is used when it is implemented on the chip.

4   Formal specification of Fat H-Tree

4.1   Definition

Rank-0 switch: Assume that $4^n = 2^{2n}$ switches are aligned in $2^n \times 2^n$ two dimensional square grid structure, and two dimensional number (x,y) is assigned to each switch. One or a few tiles are connected to each switch, and it is called a rank-0 switch.

Red tree label: For a rank-0 switch (x,y), the red tree label $R(r_0, r_1, ...r_{n-1})$ is assigned as follows.

$$r_j = ((x/2^j) \ mod \ 2) + 2 \times ((y/2^j) \ mod \ 2)$$

This label is corresponding to the relative position (0: upper left, 1: upper right, 2: lower left and 3: lower right) when the two-dimensional mesh is divided into 2×2 regions recursively as shown in Figure 6. For example, the label of the rank 0 switch (3,4) is R(1,1,2) and that of the rank 0 switch (6,7) is R(2,3,3) as shown in Figure 6.
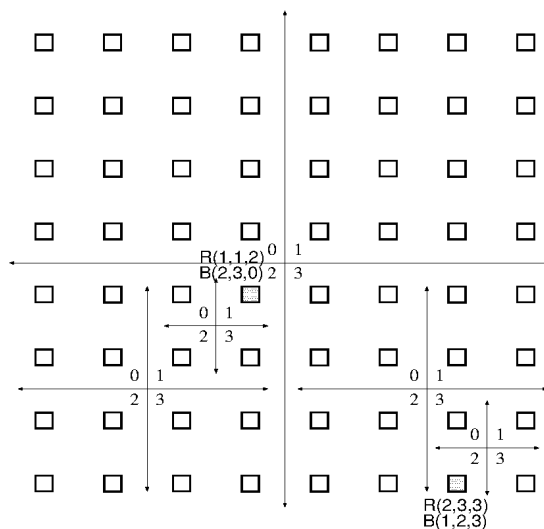
Fig. 6. Red tree label

Black tree label: For a rank-0 switch (x,y), the black tree label $B(b_0, b_1, ...b_{n-1})$ is assigned as follows.

$$b_j = (((x-1)/2^j) \bmod 2) + 2 \times (((y-1)/2^j) \bmod 2)$$

If x and y are zero, let x-1 and y-1 be $2^n - 1$.

As shown in Figure 6, the black tree label is located to the lower left direction of the switch which has the same red tree label.

Red tree and black tree: Connect four red switches labeled $R(r_0, r_1, ...r_{n-1})$ which have the same part of label $(r_1, ...r_{n-1})$ to a rank-1 switch labeled $R(r_1, , , , r_{n-1})$. In the same manner, four rank-(k-1) red switches $R(r_k, ..., r_{n-1})$ which have the same part of label $(r_{k+1}, ..., r_{n-1})$ are connected to the rank-k switch labeled $R(r_{k+1}, .., r_{n-1})$. Finally, four rank-(n-1) red switches R(0), R(1), R(2) and R(3) are connected with the root switch R to form the rank-n red tree.

Similarly, for all k from 1 to n, four rank-(k-1) black switches $B(b_k, ..., b_{n-1})$ which have the same part of label $(b_{k+1}, ..., b_{n-1})$ are connected to the rank-k switch labeled $B(b_{k+1}, .., b_{n-1})$. Finally, four rank-(n-1) black switches B(0), B(1), B(2) and B(3) are connected with the root switch B to form the rank-n black tree.

As examples, some labels of upper rank switches are shown in Figure 4.

Fat H-Tree: On $2^n \times 2^n$ rank-0 switches, a Fat H-Tree is formed with both the rank-n red tree and the rank-n black tree.

4.2   Routing on Fat H-Tree

Since Fat H-Tree provides independent two upper trees, paths through two tree structures must be searched. Both in the red and black trees, the distance between two switches $(s_0, s_1, ...s_{n-1})$ and $(d_0, d_1, ...d_{n-1})$ is $2k+1$ for the largest k where $d_k \neq s_k$. Thus, the routing on Fat H-Tree is as follows:

- Compute the distance between a source s and a destination d in the red tree, and let it be r_s.
- Three rank-0 switches which share the same red rank-1 switch as the source s are named sru, srv and srw. Compute the distances between sru, srv and srw to the destination in the black tree, and let them be b_u, b_v and b_w.
- Compute the distance between a source s and a destination d in the black tree, and let it be b_s.
- Three rank-0 switches which share the same black rank-1 switch as the source s are named sbu, sbv and sbw. Compute distances between sbu, sbv and sbw to the destination in the red tree, and let them be r_u, r_v and r_w.
- Find the minimum number from r_s, b_s, $b_u + 2$, $b_v + 2$, $b_w + 2$, $r_u + 2$, $r_v + 2$ and $r_w + 2$.
  - If the minimum number is r_s, the path in the red tree is used.
  - If the minimum number is b_s, the path in the black tree is used.
  - If the minimum number is $b_u + 2$, $b_v + 2$ or $b_w + 2$, transfer the corresponding switch with the red tree first, and then the black tree is used.

- If the minimum number is $r_u + 2$, $r_v + 2$ or $r_w + 2$, transfer the corresponding switch with the black tree first, and then the red tree is used.
- If multiple minimum paths are found, select one with a certain algorithm. The path distribution policy must be designed carefully, since it influences the performance directly. In the evaluation, three path selection algorithms: port-order, random selection and Sancho's algorithm[13] are investigated.

Since a packet is transferred to the upper direction first and then lower direction in each tree structure, the routing is deadlock free like the up*/down* routing[10]. However, paths beyond two trees may generate deadlock, dedicated buffers or channels are required in rank-1 switches. Fortunately, since the packet transfer beyond two trees is only once in a path, required buffer size or number of channel is small.

5   Performance evaluation

Here, required number of switches, average distance, traffic distribution and throughput of Fat H-Tree are evaluated, and compared with H-Tree and mesh/torus structure.

5.1   Required number of switches

Fat H-Tree requires larger number of switches than that for a simple H-Tree.

- For each rank-0 switch that takes a role of network interface also provides an extra input/output port.
- The required number of rank-1,2 and 3 switches is double of that for simple H-Tree. That is, for a system with 64 rank-0 switches, a simple H-Tree requires 64 rank-0 switches and 21 upper rank switches, while Fat H-Tree requires 64 rank-0 switches and 42 upper rank switches.

The number of required crosspoints is shown in Table 1. For two dimensional mesh structure, 64 rank-0 switches(interfaces) and corresponding switches with five ports are assumed.

Table 1 shows that the required hardware of Fat H-Tree is about double as that of H-Tree, and slightly smaller than that of two dimensional mesh. The aim of the following part is to show that Fat H-Tree gains enough performance improvement considering the above increasing hardware.

Table 1. The number of total crosspoints

Size	16	64
H-Tree	180	772
Fat H-Tree	376	1608
2D-mesh	464	1856

Table 2. Average Distance

Traffic	To-all	Linear	Mult
H-Tree	4.35	2.46	3.19
Fat H-Tree	3.78	1.12	2.12
2D-mesh	5.75	2.89	4.60

5.2   Static analysis of the topology

Average distance Fat H-Tree reduces a gap between branches of tree by providing two upper trees. In H-Tree, the distance of two of four neighbors becomes 5, since they are located beyond the gap between branches of a tree. However, in Fat H-Tree, the distance of four neighbors always becomes 1, since the gap is resolved with duplicated tree structures.

Average distance is computed assuming the following communication pattern between tasks.

- To-all: Sending a packet from a task to all other tasks.
- Linear: A stream flows straightly through tasks as shown in Figure 2. That is, transfer is limited between two neighboring tasks.
- Mult: A stream between tasks includes both linear and fork/join. In this evaluation, three forks and joins are provided in a linear stream.

Here, 64 tasks are assigned into 64 Tiles each of which is connected with a rank-0 switch. A simple task assignment policy is used: task k is assigned into a Tile connected with switch-0 labeled (x, y) where $k = 8 \times y + x$.

The average distance is shown in Table 2.

As shown in Table 2, average distance is reduced from 13% to 55%, when Fat H-Tree and H-Tree are compared. The effect is especially large for Linear and Mult traffic. Since a switch is assumed to each network interface in 2D-mesh, the average distance becomes large.

5.3   Traffic distribution

Providing two independent upper trees, Fat H-Tree can distribute traffic to avoid the congestion around the root switch. Table 3 shows average and the largest number of paths which go through a switch, when random path selection algorithm is used. The largest number is represented in the parentheses. Table 3 shows that the number of paths are concentrated to upper rank switches in H-Tree, while they are distributed to all ranks in Fat H-Tree. In two dimensional mesh, e-cube routing[14], which can distribute paths easily in k-ary n-cube is assumed. So, the paths are well distributed compared with random distribution in H-Tree and Fat H-Tree.

Throughput Throughput and latency of Fat H-Tree with 64 tiles are evaluated with a simple flit level simulator written in C++. Each switch provides five bi-directional links for connecting Tile and upper/lower rank switches. In this simulation, we used a packet transferring scheme used in Black-Bus[6]. That is, a flit packet with local ID is transferred between small buffers provided in a switch. Unlike H-Tree, there exist alternative minimal paths between some pairs of nodes in Fat H-tree. In this evaluation, three methods: port-order, random selection and Sancho's algorithm[13] are used, and the throughput and the latency between two Tiles under the uniform traffic is evaluated. In two dimensional mesh, e-cube routing which can avoid deadlock and distribute paths well is used. In the simulation, 1,000,000 packets are sent and received. The latency

Table 3. Average and the largest number of paths

Traffic	Rank	To-all	Linear	Mult
H-Tree	rank0	1.97 (63)	1.97 (2)	2.63 (9)
	rank1	7.68 (63)	5.8 (6)	8.88 (14)
	rank2	27 (60)	11.5 (12)	23 (24)
	rank3	48	15	34
Fat H-Tree	rank0	1.73(63)	2.0 (2)	2.89(11)
	red rank1	3.73 (27)	2.18 (3)	4.5(7)
	red rank2	7.75 (18)	0 (0)	5(6)
	red rank3	7	0	3
	blk rank1	2.75 (14)	2 (2)	3.38(9)
	blk rank2	5.5 (12)	0.25 (1)	2.5(6)
	blk rank3	5	0 (0)	2
2D-mesh		5.75 (63)	2.84(4)	6.03(13)

versus traffic of the result is shown in Figure 7. Here, throughput means the limited traffic that the network can be sustained. That is, the traffic that rapidly increases the latency is called the throughput.
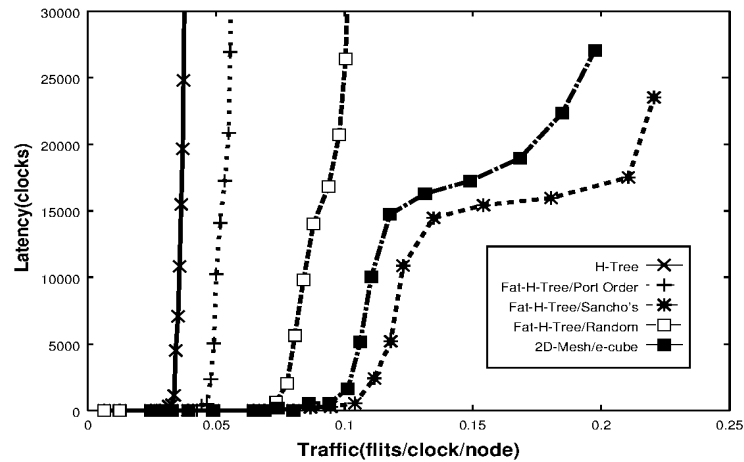
Fig. 7. Latency versus Traffic

Figure 7 shows that the throughput of Fat H-Tree is severely influenced with path distribution policy, since there are a large number of minimum paths. Port-order policy selects the path to the smallest number of port, that is the traffic tends to be concentrated. On the contrary, using random selection policy, the path is naturally distributed, and throughput is more than double of that of H-Tree. Since Sancho's algorithm, which uses the static analysis of routing paths, achieves well distributes paths, the throughput can be stretched to almost three times as that of H-Tree.

Since two dimensional mesh also provides enough number of switches as shown in Table 1, and the traffic is well distributed with e-cube routing. Thus, the throughput

is far larger than that of H-Tree. Nevertheless, Fat H-Tree with Sancho's algorithm overcomes two dimensional mesh in performance.

6   Conclusion

Fat H-tree is an novel on-chip network topology for Dynamically Reconfigurable Processor Array. It includes both fat tree and torus structure, and suitable to map tasks in a stream processing. For on-chip implementation, folding layout is also proposed. Evaluation results show that Fat H-tree reduces the distance of H-tree from 13% to 55%, and stretches the bandwidth almost three times.

Acknowledgments

The authors represent their sincere gratitude to all research group members of DRP of NEC electronics and NEC laboratories.

References

1. M.Motomura:"A Dynamically Reconfigurable Processor Architecture," Microprocessor Forum, Oct. 2002.
2. IPFlex Inc. http://www.ipflex.com
3. P.Master: "The Age of Adaptive Computing Is Here," Proceedings of the Field-Programmable Logic and Applications, pp.1–3, Sep. 2002.
4. PACT XPP Technologies http://www.pactcorp.com
5. H. Schmit, D. Whelihan, A. Tsai, M. Moe, B. Levine, and R.R. Taylor, "PipeRench: A Virtualized Programmable Datapath in 0.18 Micron Technology," Proceedings of the IEEE Custom Integrated Circuits Conference (CICC), pp.63–66, Oct. 2002.
6. K.Anjo, Y.Yamada, M.Koibuchi, A.Jouraku, H.Amano, "BLACK-BUS: A New Data-Tranfer Technique using Local Address on Networks-on-Chips," Proceedings of IEEE International conference on Parallel and Disptributed Processing Systems, Apr. 2004.
7. E.Capsi, M.Chu, R.Huang, J.Yeh, J.Wawrzynek, A.DeHon," Stream Computations Organized for Reconfigurable Execution (SCORE)," Proceedings of the Field-Programmable Logic and Applicatoins, pp.605–615, Sep. 2000.
8. A.DeHon, "Compact, Multilayer Layout for Butterfly Fat-Tree," Proceedings of the twelfth annual ACM symposium on Parallel algorithms and architectures, pp.206–215, Jul. 2000.
9. Y.Yang, A.Funahashi, H.Nishi, H.Amano, T.Sueyoshi, "Recursive Diagonal Torsu: an interconnection network for massively parallel computers," IEEE Transaction on Parallel and Distributed Systems, Vol.12, No.7, pp.701–715, Jul. 2000.
10. M.D.Schroedor and el. al. "Autonet: a high-speed self configuring local area network using point-to-point links," IEEE Selected Area in Communications, 9, pp.1318–1335, 1991.
11. C.E. Leiserson, "Fat-trees: Universal Networks for Hardware-Efficient Supercomputing," IEEE Transaction on Computer, vol. 34 no. 10, pp. 892-901, Oct. 1985.
12. W.J.Dally and B.Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks," Proceedings of 38th Design Automation Conference, pp.684–689, Jun. 2001.
13. J.C.Snacho, A.Robles and J.Duato, "Effective Strategy to Compute Forwarding Tables for InfiniBand Networks," Proceedings of the International Conference of Parallel Processing," pp.48–53, Jan. 2001.
14. W.J.Dally, "Virtual Channel Flow Control," IEEE Transaction on Parallel and Distributed Systems, Vol.3, No.2, pp.194–205. Mar. 1992.

Electronic Acknowledgement Receipt

EFS ID:	34485990
Application Number:	16202067
International Application Number:	
Confirmation Number:	8134
Title of Invention:	VLSI Layouts of Fully Connected Generalized Networks
First Named Inventor/Applicant Name:	Venkat Konda
Customer Number:	38139
Filer:	Venkat Konda
Filer Authorized By:	
Attorney Docket Number:	V-0070US
Receipt Date:	04-DEC-2018
Filing Date:	27-NOV-2018
Time Stamp:	20:55:17
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Information Disclosure Statement (IDS) Form (SB08)	16-202067-sb0008b.pdf	450240 7c7295695e5f14f0d45a6606949771f8abbd86ea	no	2

Warnings:

Information:					
This is not an USPTO supplied IDS fillable form					
2	Non Patent Literature	chen97tighter.pdf	323946 0dd2b5cbe895ff3ee8cd042029c2681fdd9d e6d3	no	17
Warnings:					
Information:					
3	Non Patent Literature	DeHon-Unify-Mesh-Tree.pdf	1438096 6ddb3d1e190b2a26073561208584bd2420 e7395c	no	15
Warnings:					
Information:					
4	Non Patent Literature	dehon99balancing.pdf	245980 70bb376a2e4a41c4ebcf25f402ea77dccc63 4c7c	no	10
Warnings:					
Information:					
5	Non Patent Literature	fold-spaa2000.pdf	199719 c70318b2c9e36d6a8e97400a432f5d2c3b5 ed502	no	10
Warnings:					
Information:					
6	Non Patent Literature	Segmented-Routing.pdf	1410463 74349da7d5c21a06a3fbbe8a991296781c7 7b49e	no	17
Warnings:					
Information:					
7	Non Patent Literature	Switchbox-Design.pdf	151991 87be02898d4d0bf00141b602a90007c7f9a 76e09	no	10
Warnings:					
Information:					
8	Non Patent Literature	tc-gusb.pdf	578658 0281587e0334ddcb7ce8b8fc128a84a2972 ced97	no	12
Warnings:					
Information:					

| 9 | Non Patent Literature | yamada-euc-2004.pdf | 199391 | no | 10 |
| | | | ca0d82b4e0733949bdac38ce71e1bb66ec5ba302 | | |

Warnings:

Information:

	Total Files Size (in bytes):	4998484

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.
National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.
New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

Substitute for form 1449/PTO **INFORMATION DISCLOSURE STATEMENT BY APPLICANT** *(Use as many sheets as necessary)*	**Complete if Known**	
	Application Number	16/202067
	Filing Date	12-4-2018
	First Named Inventor	Venkat Konda
	Art Unit	
	Examiner Name	
Sheet 1 of 1	Attorney Docket Number	V-0070US

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
	1	Ivo Dobbelaere, Mark Horowitz, and Abbas El Gamal. Regenerative feedback repeaters for programmable interconnections. IEEE Journal of Solid-State Circuits, 30(11), 1995.	
	2	F. Petrini et. al., "k-ary n-trees: High performance networks for massively parallel architectures, in: Proceedings of the 11th Intl Parallel Proc. Symp. , IPPS'97, pp. 87-93	
	3	P.Pande et al. "Evaluation of MP-SoC Interconnect Architectures: a Case Study", Proceedings of 4th IWSOC, Banff, Alberta, Canada, 19th-21st July, 2004	
	4	Yeh, C.-H., Varvarigos, E.A., Parhami, B.: Multilayer VLSI layout for interconnection networks. In: Proc. Intl. Conf. on Parallel Processing, 2000.	
	5	M. Lin, A. El Gamal, "A Low-Power Field-Programmable Gate Array Routing Fabric," IEEE Transactions on Very Large Scale Integration, Vol. 17, No. 10, pp. 1481-1494, Oct. 2009	
	6	AVIOR, A et. al., A Tight Layout of the Butterfly Network. Proc. 8-th Annual ACM Symp. on Parallel Alg. and Arch. (SPAA '96), ACM Press Ed., 1996, pp 170–175.	
	7	A. El Gamal et. al., "An Architecture for Electrically Configurable Gate Arrays," IEEE Jrnl of Solid-State Circuits, Vol. 24, No. 2, pp. 394-398, April 1989.	
	8	Vaughn Betz et. al., Directional bias and non-uniformity in FPGA global routing architectures. In IEEE/ACM Intl. Conference on Computer-Aided Design, pp. 652-659, san jose, 96	
	9	André DeHon. Rent's Rule Based Switching Requirements. In System-Level Interconnect Prediction (SLIP 2001), pages 197--204, March 31--April 1, 2001	

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

1 Applicant's unique citation designation number (optional). 2 Applicant is to place a check mark here if English language Translation is attached.
This collection of information is required by 37 CFR 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

Privacy Act Statement

The **Privacy Act of 1974 (P.L. 93-579)** requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (*i.e.*, GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

Electronic Acknowledgement Receipt

EFS ID:	34486057
Application Number:	16202067
International Application Number:	
Confirmation Number:	8134
Title of Invention:	VLSI Layouts of Fully Connected Generalized Networks
First Named Inventor/Applicant Name:	Venkat Konda
Customer Number:	38139
Filer:	Venkat Konda
Filer Authorized By:	
Attorney Docket Number:	V-0070US
Receipt Date:	04-DEC-2018
Filing Date:	27-NOV-2018
Time Stamp:	21:12:23
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Non Patent Literature	iccad96.pdf	61851 db9bd3def8037be018e5dfcab0485641a2d52eb1	no	8

Warnings:

Information:

2	Non Patent Literature	Interconnect-Architecture.pdf	477835	no	5
			d983056a29d99c7a98d2a192fc68dd6319b5a62e		

Warnings:

Information:

3	Non Patent Literature	light-layout-of-butterfly-network.pdf	286748	no	14
			b1185d69acf8ae11965f973e3ce25d3039363dcf		

Warnings:

Information:

4	Non Patent Literature	LP-FPGA-Fabric.pdf	1080349	no	14
			f9418381d3375c3a053afb58a207d362057b4d46		

Warnings:

Information:

5	Non Patent Literature	ML-VLSI-layouts-for-InterconNetwks.pdf	109196	no	8
			d09cd8c0e4fa8875381783ff939a165cd9ba8884		

Warnings:

Information:

6	Non Patent Literature	pandep-MP-SoC-Interconn.pdf	210576	no	4
			dec2bd4b967d5facd4b59d783b97636b4a922a1e		

Warnings:

Information:

7	Non Patent Literature	petrini97kary.pdf	119892	no	7
			86e78185b99255ad0e063a3eeb151106ff4c05ae		

Warnings:

Information:

8	Non Patent Literature	Programmable-interconnections.pdf	959944	no	8
			5d5bbd957fbdba455f41cfad06e99e3448faf29c		

Warnings:

Information:

| 9 | Non Patent Literature | rentsw-slip01.pdf | 1446852 | no | 8 |
| | | | b1cba481afb7f05bd0bd03cc64bdecde5eb 2db58 | | |

Warnings:

Information:

| 10 | Information Disclosure Statement (IDS) Form (SB08) | 16-202067-sb0008b.pdf | 510695 | no | 2 |
| | | | ca2029a064a9471ff7059e8014502262bc8a 4068 | | |

Warnings:

Information:

This is not an USPTO supplied IDS fillable form

	Total Files Size (in bytes):	5263938

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

<u>**New Applications Under 35 U.S.C. 111**</u>
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.
<u>**National Stage of an International Application under 35 U.S.C. 371**</u>
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.
<u>**New International Application Filed with the USPTO as a Receiving Office**</u>
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

ICCAD 1996

Directional Bias and Non-Uniformity in FPGA Global Routing Architectures

Vaughn Betz and Jonathan Rose

Department of Electrical and Computer Engineering, University of Toronto

Toronto, Ontario, Canada   M5S 3G4

{vaughn, jayar}@eecg.utoronto.ca

Abstract

This paper investigates the effect of the prefabricated routing track distribution on the area-efficiency of FPGAs. The first question we address is whether horizontal and vertical channels should contain the same number of tracks (capacity), or if there is a density advantage with a directional bias. Secondly, should the channels have a uniform capacity, or is there an advantage when capacities vary from channel to channel? The key result is that the most area-efficient global routing architecture is one with uniform (or very nearly uniform) channel capacities across the entire chip in both the horizontal and vertical directions. Several non-uniform and directionally-biased architectures, however, are fairly area-efficient provided that appropriate choices are made for the pin positions on the logic blocks and the logic array aspect ratio.

1   Introduction

In recent years Field-Programmable Gate Arrays (FPGAs) have seen explosive market growth because they offer instant manufacturing and much lower non-recurring engineering costs than Mask-Programmed Gate Arrays. FPGAs enable fast manufacturing and low development costs because their logic and routing resources are prefabricated and are customized in the field by the designer [1].

The prefabrication of routing resources in an FPGA implies that the number of routing tracks in each channel is set by the manufacturer. It is vital that these routing resources be distributed in a manner that allows their efficient utilization by the largest class of circuits. If there are too few tracks in some area of the chip then many circuits will be unroutable, while if there are too many tracks, they may be wasted.

This paper addresses several questions concerning the distribution of routing tracks across an FPGA. First, should the number of tracks in the horizontal channels be different from the number in the vertical channels? We refer to this as a *directional bias*; Figure 1(a) illustrates an example directionally-biased FPGA. In essence, we are investigating if there is an intrinsic property of circuits that makes a directional bias more area efficient. If so, what

amount of bias is best? Commercial FPGAs with both unbiased routing [2, 3] and biased routing [4, 5] exist, so this question has clear commercial relevance.

Second, should all channels in the same direction in an FPGA be the same width or should some channels be wider than others to facilitate routing in congested regions? We refer to architectures in which the channels in some regions are wider than the channels in others as *non-uniform* routing architectures; Figure 1(b) depicts an example.[1] Many in the FPGA community believe that most routing congestion occurs near the center of an FPGA, and hence channels in this region should be wider than the channels near the edges. In fact, the Lucent Technologies (formerly AT&T) ORCA 2C series FPGAs have an extra-wide channel in the center of the chip to improve routability [6]. In addition, board-level constraints often force designers to fix the position of an FPGA's I/Os, and some believe that this increases congestion near the chip edges so that the channel between the pads and the logic should be made extra wide. The Xilinx 5000 series FPGA has a wide channel between the pads and logic, at least partially to improve routability when the I/O locations are fixed [7]. In this paper, we determine the best distribution of tracks across an FPGA both when the I/O assignment to pads is unconstrained and when it is fixed in a poor configuration.

We evaluate FPGA architectures experimentally;

(a) Directionally-Biased   (b) Non-Uniform
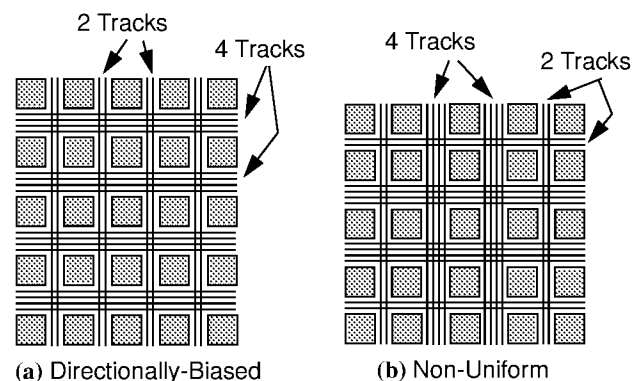
Figure 1: Types of Global Routing Architectures.

1. Note that any given channel will always have the same number of tracks along its entire length. We did not consider varying the channel capacity along its length as this makes it very difficult, and likely impractical, to lay out the FPGA.

ICCAD 1996

benchmark circuits are placed and routed into FPGAs with different global routing architectures to determine the relative area consumed by the circuit in each architecture. To obtain meaningful results, the CAD tools used to place and route these circuits must take advantage of the biased and non-uniform nature of these architectures. Accordingly, we have created a new placement and routing tool which aggressively seeks to minimize congestion and fully utilize the channels of the specified architecture during both placement and global routing.

The organization of this paper is as follows. Section 2 outlines the CAD flow used to evaluate the different FPGA architectures. Section 3 describes the custom placement and routing CAD tools. We evaluate the area-efficiency of FPGAs with differing amounts of directional routing bias in Section 4. In Section 5 we address the uniform vs. non-uniform channel thickness question. Finally, we summarize our results and conclusions.

2 Experimental Methodology

To compare the area-efficiency of the different global routing architectures we technology-map, place and route 26 of the largest MCNC benchmark circuits [8] into each architecture. In this section we describe the CAD flow, the area-efficiency metric used to compare architectures, and several important architectural details.

2.1 CAD Flow

Figure 2 summarizes the CAD flow. First, the circuit is optimized by SIS [9]; next it is technology-mapped by Flowmap [10] into four-input look-up tables (4-LUTs) and flip flops. The logic block used in these experiments contains a 4-LUT and a flip-flop, as illustrated in Figure 3. A custom-built program (blifmap) packs the 4-LUTs and flip flops together into these logic blocks.

The netlist of logic blocks and a description of the FPGA global routing architecture are then read into the placement and global routing tool, VPR. This program
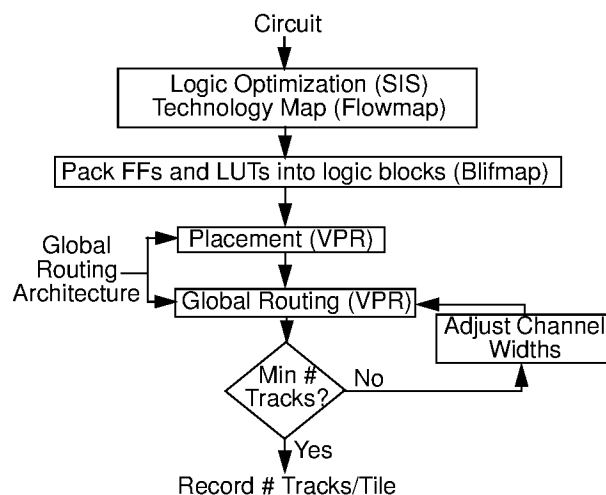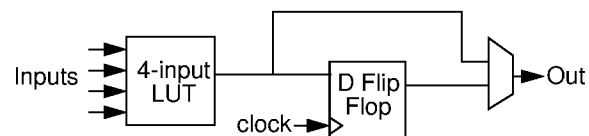
Figure 3: Logic Block Structure.

places the circuit, and then repeatedly routes (or attempts to route) the circuit with different numbers of tracks in each channel (*channel capacities*). VPR performs a binary search on the channel capacities, increasing them after a failed routing and reducing them after a successful one, until it finds the minimum number of tracks required for the circuit to route successfully on a given global routing architecture. While the absolute number of tracks per channel is adjusted upwards or downwards after each attempted routing, the *relative* numbers of tracks in the various channels across the FPGA are always kept at the values specified by the FPGA architecture. For example, VPR's first attempt at routing a circuit in an architecture with a two-to-one directional bias might assume horizontal channel capacities of twelve tracks and vertical channel capacities of six tracks. If this routing was successful, VPR would next attempt to route the circuit in an FPGA with horizontal channel capacities of six tracks and vertical channel capacities of three tracks, and so on until the minimum number of tracks required for routing is determined.

The benchmark circuits used in this study consist of 14 combinational and 12 sequential MCNC benchmark circuits [8], which vary in size from 222 to 1878 of our logic blocks.

2.2 Area-Efficiency Metric

Our goal is to measure the area-efficiency of different global routing architectures without reference to the detailed routing architecture (e.g. segmentation and switch block topology). At this level, it is the amount of "global wiring" that changes as we vary the architecture. A simple track count will not accurately represent the wiring area of rectangular FPGAs, as the tracks in one direction are longer than those in the other. Accordingly, we define a *track segment* to be a prefabricated wire that spans one logic block; a channel of width W tracks that spans L logic blocks contains WL track segments. The total number of track segments an FPGA must contain to globally route a circuit is a representative metric of the "global wiring" area. In order to average the results from circuits of differing sizes we use the average number of track segments per tile (i.e. per logic block) as our area measure. For example, in a square N x N uniform FPGA with W tracks in each channel, the total number of track segments is $2WN^2$, and the number of tracks per tile is 2W. Note that the routing area is given by the total number of track segments in the entire FPGA, and not the number of track segments which are actually used by a circuit.

Figure 2: Architecture Evaluation Flow.

ICCAD 1996

2.3 Significant FPGA Architectural Details

Several architectural parameters other than the global routing architecture must be specified in order to define an FPGA. We set these parameters to be as close to those of commercial FPGAs as possible.

First, the size of the FPGA array used for a given circuit (i.e. the number of logic blocks) is set to be the *smallest* FPGA with the desired aspect ratio (number of columns / number of rows) with sufficient logic blocks to accommodate the circuit. This situation, in which there is minimal "spare room" in the FPGA, presents the greatest challenge to routing completion and is normally the case manufacturers wish to optimize.

In this study the number of I/O pads that can fit into the height or width of a logic block is set to two. This number is commensurate with the relative sizes of I/O pads and 4-LUTs in current FPGAs [2, 3, 5] and ensures that none of the 26 benchmarks is pad-limited.

Finally, we do not route the clock net in sequential circuits, since this net is normally distributed through a special clocking network in commercial FPGAs.

3 Tuned Placement and Routing Algorithms

In FPGA architecture explorations of this kind [1] one must ensure that the CAD tools used are responsive to the architectural parameters being varied. To ensure a fair comparison between different global routing architectures, we created a new placement and global routing tool which directly exploits biased and non-uniform routing architectures. As this CAD tool is capable of mapping to a wide variety of FPGA architectures, we named it VPR, short for Versatile Place and Route; it is publicly available from http://www.eecg.toronto.edu/~jayar/software.html.

3.1 Global Routing Resource-Aware Placement

We employ the simulated annealing algorithm [11] for placement. The key to a routing-resource-aware placement tool is ensuring that the cost function correctly models the relative difficulty of routing connections in regions with different channel widths. After significant experimentation with many alternatives [15], we have developed a *linear congestion* cost function which provides the best results in reasonable computation time. Its functional form is

$$Cost_{linear} = \sum_{n=1}^{M} q(n) \left[\frac{bb_x(n)}{C_{av,x}(n)^{\alpha}} + \frac{bb_y(n)}{C_{av,y}(n)^{\alpha}} \right]$$

where the summation is over the M nets in the circuit. For each net, bb_x and bb_y denote the horizontal and vertical spans of its bounding box, respectively. The q(n) factor compensates for the fact that the bounding box wire length model underestimates the wiring necessary to connect nets with more than three terminals, as suggested in [12]. Its value depends on the number of terminals of net n; q is 1 for nets with 3 or fewer terminals, and slowly increases to

2.79 for nets with 50 terminals. $C_{av,x}(n)$ and $C_{av,y}(n)$ are the average channel capacities (in tracks) in the x and y directions, respectively, over the bounding box of net n.

This cost function penalizes placements which require more routing in areas of the FPGA that have narrower channels. The exponent, α, in the cost function allows the relative cost of using narrow and wide channels to be adjusted. When α is zero the linear congestion cost function reverts to the standard bounding box cost function. The larger the value of α, the more wiring in narrow channels is penalized relative to wiring in wider channels; we have experimentally found that setting α to 1 results in the highest quality placements.

Since C_{av} depends only on the channel capacities, which do not change during a placement, and on the maximum and minimum coordinates of the bounding box, we precompute all possible $C_{av,x}$ and $C_{av,y}$ values. Consequently, recomputing this cost function is essentially as fast as recomputing the traditional bounding box cost function.

In an FPGA where all channels have the same capacity, C_{av} is also a constant and hence the linear congestion cost function reduces to a bounding box cost function. In non-uniform and directionally-biased FPGAs, however, this cost function results in higher quality placements than a bounding box cost function. The exact amount of routability improvement depends on the precise global routing architecture used; as one would expect, those in which there is a large difference between the widths of channels in different regions show the largest improvement. For the architectures studied in this paper, placements produced with the linear congestion cost function typically require 5 to 10% fewer tracks to route than placements produced with a bounding box cost function.

We also implemented the cost function of [12], which we call a *non-linear congestion* cost function. This cost function divides the FPGA into an array of N x N regions and attempts to model the routing resource demand and supply in each of these regions. When a placement causes the routing resource demand to exceed the supply in some regions, the placement is heavily penalized. We found that this non-linear congestion cost function, when computed on a 4 x 4 grid (16 regions), generally produces placements which require 2 to 4% fewer tracks to route than those produced by the linear congestion cost function. However, keeping track of the routing resource demand in the various chip regions is computationally expensive, and placement with this cost function requires five times greater CPU time than the linear congestion function. Dividing the FPGA into smaller subregions to make localized congestion more visible did not work well; a non-linear congestion cost function computed on a 16 x 16 grid (256 regions) performs only marginally better than a cost function computed on a 4 x 4 grid, yet consumes sixteen times the CPU time.

We considered the reductions in track count achieved by the non-linear congestion cost function too small to warrant the additional CPU time, so the results presented

in this study all use the linear congestion cost function. Nonetheless, we did rerun a few of our experiments with the non-linear congestion cost function and found that its use did not change any of the architectural conclusions presented below.

3.2 Congestion-Driven Global Routing

It is crucial for the global router to leverage the differences in the capacities of the various FPGA channels. The global router developed for this study employs a variant of the PathFinder negotiated congestion algorithm [13]. This algorithm consists of routing each net with a maze router [14], then ripping up and rerouting each net in sequence several times. In each of these subsequent routing iterations, the cost of using a node (which is *either* a channel segment or a logic block input pin) is modified, based on the competition for that node in both the current iteration and all previous iterations. A channel segment is the length of channel that spans one logic block; in an FPGA composed of an N x N array of logic blocks each channel contains N segments. We define the cost of a routing node somewhat differently than [13]; the cost of using routing node *n* is

$$c_n = (1 + h_n \cdot h_{fac}) \times (1 + p_n \cdot p_{fac}) + b_{n, n-1}.$$

The p_n term is a measure of the present congestion at this node. It is updated *every time any net* is ripped-up and rerouted. The value of p_n is equal to the overuse of this node that would occur if one more route were to use it, since the decision we are making during routing is whether another net should go through this node or not. For example, consider a channel with a capacity of six tracks and a segment of this channel in which all six tracks are currently used. The p_n value of this channel segment is one, since routing one more net through this channel will result in an overuse of one.

The h_n term accounts for the historical, or past, congestion at this node. It is updated *only after an entire routing iteration* is completed; i.e. after every net in the circuit has been ripped up and rerouted. Initially h_n is 0; at the end of each routing iteration h_n is increased by the amount by which demand for this node outstrips its capacity.

The $b_{n,n-1}$ term penalizes bends, since global routes with many bends in them present a more difficult detailed routing problem in FPGAs with segmented routing, and will generally lead to detailed routes that are both slower and require more tracks. The value of $b_{n,n-1}$ is one if making the connection from node n-1 to node n implies a bend (i.e. node n-1 is a horizontal channel segment and node n is a vertical channel segment or vice versa), and is zero otherwise. Including this bend cost in the total cost of using a node produces routes with very few unnecessary bends and increases the global routing track count very little.

The key idea of Pathfinder is that the p_{fac} term is 0 for the first routing iteration, and is gradually increased in successive iterations. Hence, each net is initially routed by the

shortest path found. In successive iterations, the p_{fac} term is gradually made larger so that congestion becomes more expensive and those nets which have alternate routes move out of the congested areas. The history term, h_{fac}, allows information from previous routing iterations to affect the current routing, further improving the router's ability to find and avoid congestion. By treating both channel segments and input pins as routing nodes, this algorithm makes use of the functional equivalence of LUT input pins in a very natural way. Initially, each connection uses the logic block input pin which leads to the shortest route. As the cost of congestion increases, nets are gradually forced to use unique input pins.

In our implementation each channel can have a different capacity. Since the cost of a channel segment is based on the amount by which routing demand exceeds its capacity, this router will automatically act to relieve pressure on narrow channels by rerouting nets through wider channels whenever necessary.

Considerable effort was spent tuning the *routing schedule* (the values of p_{fac} and h_{fac} over the course of the iterations). The best routing schedule we found set p_{fac} to 0 for the first iteration, 0.5 for the second iteration, and 1.5 times the previous p_{fac} value for all subsequent iterations. The value of h_{fac} was set to 0.2 for all iterations; the fact that h_n can only increase from iteration to iteration provides sufficient increase in the historical congestion penalty. This routing schedule increases the cost of congestion slowly enough that the net ordering is not very important - - nets with the most alternate routes move out of congested areas first. Increasing the cost of congestion more slowly than this reduced the number of tracks required only by 1 - 2% while increasing the CPU time by a factor of 2 to 3. Setting h_n to 0 so that the router has no information about past congestion increased the number of tracks required by 15%.

4   Experimental Results for FPGAs with Directionally-Biased Routing Resources

The experimental framework and tools described above were employed to answer the questions posed in the introduction to this paper: first, is there an area-efficiency advantage to a directionally-biased architecture? Recall that in a directionally-biased FPGA the number of routing tracks in the horizontal and vertical directions are different. In essence, we are investigating if there is an exploitable directional bias in the basic nature of circuits. We characterize directionally-biased FPGAs by the ratio of the width of a horizontal channel to the width of a vertical channel, denoted as R_h. For example, Figure 1(a) depicts an FPGA with a 2:1 directional bias, i.e. $R_h = 2$.

We need to define an additional architectural feature which markedly affects our results: the positioning of the pins on the logic block. The two main cases of interest are illustrated in Figure 4. In Figure 4(a), the logic block input and output pins are distributed evenly around the entire perimeter of each logic block. We call this the *full-perime-*

ICCAD 1996
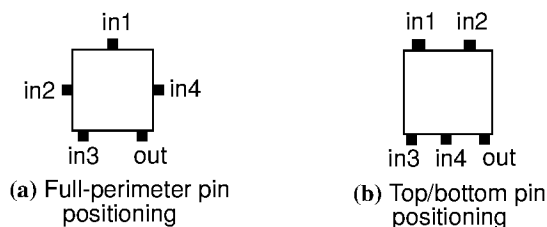
Figure 4: Logic Block Pin Position Alternatives.

ter pin positioning, and it is similar to the pin positioning used in Xilinx and ORCA FPGAs [2, 3]. Figure 4(b) illustrates the *top/bottom* pin positioning, which restricts the logic block input pin locations to lie only on the top and bottom of the logic block; it is similar to the pin positioning used in Actel FPGAs [4]. In all the results we show in this paper, each logic block pin appears (physically) on only one side of a logic block. We have found that for the channel connectivity values (F_c [1]) found in today's commercial FPGAs this leads to the most area-efficient FPGAs [15].

We have also found that the ratio of the number of columns to the number of rows in an FPGA, which we call the aspect ratio, significantly affects area efficiency. Since most FPGAs have the same number of rows and columns, we first present the results for square (aspect ratio 1) FPGAs, before discussing the more general case of rectangular FPGAs in Section 4.2.

4.1 Results for Square FPGAs

Twenty-six large MCNC benchmarks were passed through the experimental flow of Figure 2 for values of R_h ranging from 1 to 4. As discussed in Section 2, the result for each circuit is the number of track segments *per tile* needed to successfully global route the circuit in an FPGA[2] with the specified value of R_h. Figure 5 is a plot of area-efficiency versus the degree of routing direction bias, R_h, for both types of pin positioning. The vertical axis is the average number of tracks per tile required to successfully route the 26 benchmarks.

For the full-perimeter logic pin positioning, the best architecture has no directional bias. However, when the pins are restricted to the top and bottom of the logic block, the most efficient architecture has horizontal channels which are roughly twice as thick as the vertical channels. An important conclusion is that the best full-perimeter architecture is about 8% more area-efficient than the best top/bottom pin architecture.

The full-perimeter architecture is more area-efficient because there is a greater chance that the block input pins are closer to their desired connections when they are in this configuration than when they are in the top/bottom configuration. For example, consider the two routings of a multi-terminal net shown in Figure 6. The top/bottom pin configuration needs six track segments to route this net,

---

2. Track segments are counted whether or not they are actually used, so this is a true representation of the area that must be devoted to routing in the layout.

Horizontal Capacity / Vertical Capacity (R_h)
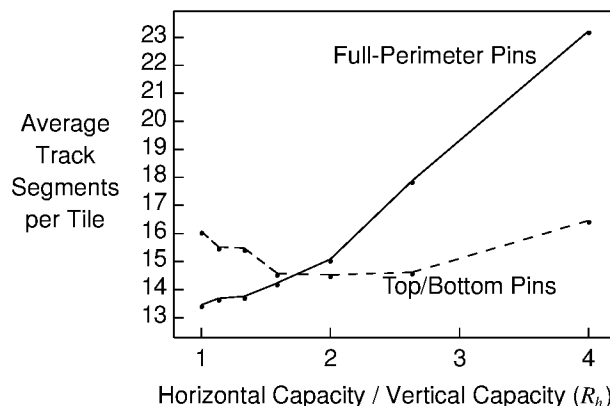
Figure 5: Area-Efficiency vs. Directional Bias for Square FPGAs.

while the full-perimeter configuration requires only five. By making use of the functional equivalence of LUT input pins during routing, the router can often connect to a logic block pin adjoining a track segment it needs to use for other connections, essentially making the connection to this logic block for free. Since the top/bottom pin configuration has input pins bordering on only the horizontal channels, such "free" connections into logic blocks are less frequent, reducing area-efficiency.

The full-perimeter pins configuration achieves the highest area-efficiency when there is no directional bias to the routing because this makes the difficulty of routing to each of a logic block's nearest neighbors roughly equal. Consequently, the placement software can use all the nearby logic block locations equally to cluster the fanout of a net around its driver. Essentially, this allows one to cluster tightly coupled portions of logic in the smallest possible area. The top/bottom pins configuration, on the other hand, prefers a 2:1 directional bias because every connection to a logic block pin must come from a horizontal channel. This extra pressure on the horizontal routing resources is significant, since the typical distance routed between pins is only about 3 track segments.

4.2 Rectangular FPGAs

In order to increase the IO-to-logic ratio, FPGA manufacturers may want to build rectangular FPGAs, as this increases the die perimeter and hence the number of pads. In this case the channels in one direction are longer and have more blocks connected to them than the orthogonal channel, so the best amount of directional bias may

(a) Full-Perimeter Pins.   (b) Top/Bottom Pins.
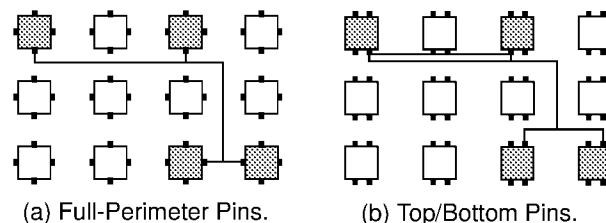
Figure 6: Example Multi-Terminal Net Routing.

ICCAD 1996
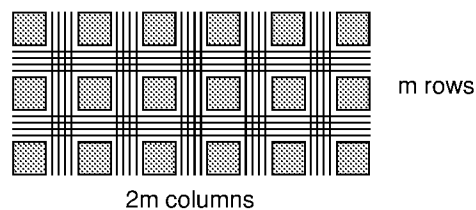
2m columns

Figure 7: An FPGA with an Aspect Ratio of 2.

change. We refer to the ratio of the number of columns in an FPGA to the number of rows as its aspect ratio. Figure 7 depicts an FPGA with an aspect ratio of two.

Figure 8 is a plot of the required tracks per tile versus R_h for various chip aspect ratios for an FPGA with the full-perimeter logic block pin positioning. There are two features of interest in Figure 8. First, notice that the minimum of the aspect ratio = 1 curve is the lowest of the three, indicating that a square FPGA is most area-efficient. Secondly, the value of R_h at which the minimum area occurs *increases* as the aspect ratio increases. As the aspect ratio increases, the horizontal channels become longer than the vertical channels and this results in greater demand for horizontal track segments. The best value of R_h increases from 1 for a square FPGA to 1.33 and 1.59 for aspect ratios of 2 and 3, respectively.

The solid curve in Figure 9 shows how area-efficiency varies with aspect ratio when we set R_h to the most appropriate value for each aspect ratio. The dotted curve in Figure 9 keeps R_h fixed at 1, which is the best value for a square FPGA. The routing resource requirements increase moderately with aspect ratio; an FPGA with an aspect ratio of 3 requires 18% more tracks per tile than a square FPGA when R_h is 1. When the most appropriate value of R_h is used for each aspect ratio, however, an FPGA with an aspect ratio of 3 requires only 4% more track segments than a square FPGA. Thus we conclude that, as long as the horizontal and vertical channel widths are appropriately balanced, chip aspect ratios, and hence I/O counts, can be increased with little impact on the core area.

The variation of core routing area with aspect ratio is similar for FPGAs that use the top/bottom logic block pin positioning [15]. In this case an FPGA with an aspect ratio

Horizontal Capacity / Vertical Capacity (R_h)
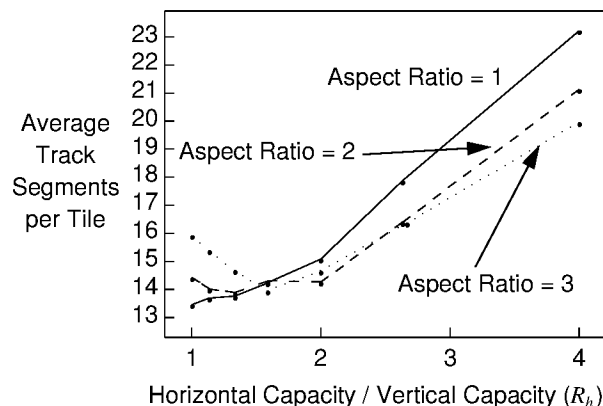
Figure 8: Area-Efficiency of Rectangular FPGAs with Full-Perimeter Pins.
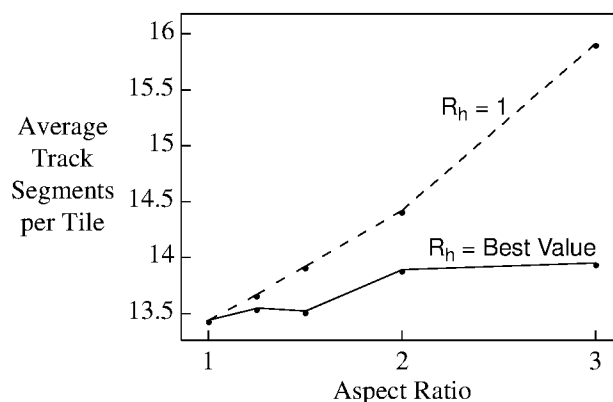
Aspect Ratio

Figure 9: Area-Efficiency vs. Aspect Ratio for FPGAs with Full-Perimeter Pins.

of 3 requires only 5% more tracks per tile than a square FPGA. For FPGAs of this type, however, the increase in the best value of R_h as aspect ratio increases is less dramatic. The best square FPGA with top/bottom pins has horizontal channels which are twice as wide as vertical channels; the thicker horizontal channels are better able to cope with the increased pressure for horizontal tracks as aspect ratio increases.

5   Experimental Results for FPGAs With Non-Uniform Routing

The second key issue we explore concerns the area-efficiency obtained when the channels in different regions of an FPGA have different capacities. We only investigate FPGAs which use the full-perimeter pin positioning, as Section 4 showed that this pin positioning is best.

We define a non-uniform routing architecture to be one in which the number of tracks per channel changes from channel to channel across an FPGA. For example, Figure 1(b) illustrates a non-uniform FPGA in which the channels near the chip center are wider than those near the periphery. If congested regions of a circuit can be localized and placed in the portions of the FPGA with the widest channels, a non-uniform FPGA could have better area efficiency than a uniform FPGA. We will investigate two types of non-uniform FPGAs: one in which we vary the center/edge channel capacity ratio, and one in which we vary the I/O channel capacity.

5.1 Center/Edge Capacity Ratio

There is a widespread belief that most congestion occurs in the center of FPGAs, and hence having wider channels near the FPGA center and narrower channels near the edges is expected to improve area-efficiency. To keep the layout problem tractable, we restrict ourselves to FPGAs which use channels of only two different widths. We can describe global routing architectures of this form with two parameters. Let R_w be the ratio of the widths of the channels near the center of the FPGA to the widths of the channels near the FPGA edges, i.e. W_{center} / W_{edge}. Let

ICCAD 1996

R_c be the ratio of the number of channels with width W_{center} to the total number of channels. For example, the FPGA of Figure 1(b) has $R_w = 2$ and $R_c = 0.5$.

Using the flow of Section 2, we again implemented 26 benchmark circuits in several architectures to determine their area-efficiency. We examined FPGAs with R_w equal to 0.75, 1.18, 1.33, and 2, and with R_c values varying from 0 to 1. The relative density of FPGAs with $R_w = 1.33$ and $R_w = 2$ is summarized in Figure 10. Note that the points at which R_c equals 0 or 1 correspond to a uniform FPGA.

The results generally show that the less uniform the channel widths, the worse the FPGA area-efficiency. The worst area-efficiency with $R_w = 2$ occurs when R_c is 0.5, meaning that half the FPGA channels are twice as wide as the other half. In fact, only two non-uniform FPGAs show even marginal area-efficiency improvements over the uniform case and both these FPGAs are very close to a uniform architecture. In one, the 10% of channels nearest the center are 33% wider than the other channels, while in the other the 90% of channels closest to the center are 33% wider than the channels nearest the edges. The reduction in tracks per tile over a uniform FPGA is less than 1% for both of these FPGAs, so the improvement is not sufficient to justify the extra layout effort required in the physical design of such an FPGA.

We also evaluated FPGAs in which only the center-most channel in each direction was extra-wide, since a commercial FPGA of this architecture has recently been introduced [6]. A uniform FPGA outperformed this non-uniform architecture as well [15].

These results are significant because there is a common belief among FPGA architects that there would be significant benefit to these kinds of non-uniform architectures. The fundamental reason they do not show any benefit is that there is not much more congestion in the center of an FPGA than there is near its edges. In order to determine the "natural" routing demand distribution of circuits, we placed and routed the 26 benchmark circuits with all congestion avoidance features disabled, so that placement minimized wirelength and the router connected each net by the shortest path. Figure 11 plots the maximum and average number of tracks required by the horizontal chan-
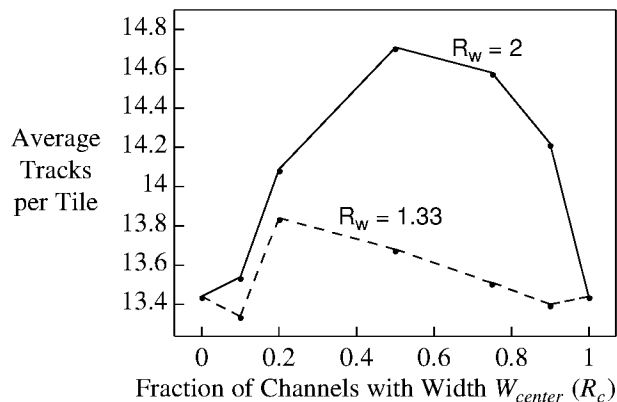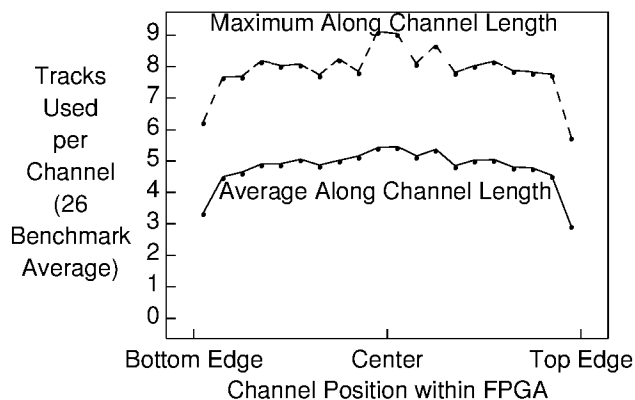
Figure 11: Average over Benchmarks of Track Demand vs. Position for Horizontal Channels.

nels as a function of the channel position within the FPGA, averaged over the 26 benchmark circuits. The results for individual circuits closely parallel these overall averages. Demand for routing tracks is relatively constant over the middle 90% of the FPGA, and there is only a moderate decrease as one gets very close to the chip edges.

In addition, typical circuits contain numerous local congestion "hotspots" (small regions where all the channels are full) and some of these hotspots occur quite close to the FPGA edge. Consequently, in order for an FPGA with thicker channels near its center to use fewer routing resources, the placement software must move all of these hotspots into the FPGA center. As discussed in Section 3.1, we spent considerable time investigating placement cost functions that modelled congestion well. The more advanced, and computationally expensive, cost functions, however, improved the performance of the uniform FPGA more than they did the non-uniform FPGA. We believe it is therefore more effective for CAD tools to attempt to spread out congestion as much as possible, rather than to try to localize it to a designated portion of a chip.

5.2 I/O Channel

Another major FPGA vendor has added routing resources to the "I/O-channel" that runs between the I/O pads and the logic blocks, at least in part to ensure that fixed I/O pad placement does not impact routability and speed [7]. We define R_{io} to be the ratio between the width of this outermost channel and the width of the other channels. Figure 12 is a plot of the average tracks/tile required for the 26 benchmarks circuits versus R_{io}. The solid line in Figure 12 shows the trend when the I/O locations are chosen by the placement tool, while the dashed line is found when the I/O pads are "fixed" in a random location, to model the effect of poor (from the FPGA's point of view) pin constraints.

There are several features of interest in Figure 12. First notice that fixing the I/O locations increases the number of routing tracks required by 12% on average. Architects must take this into account when designing FPGAs. Secondly, the curve where the I/O locations are chosen by

Figure 10: Area-Efficiency vs. Routing Architecture.
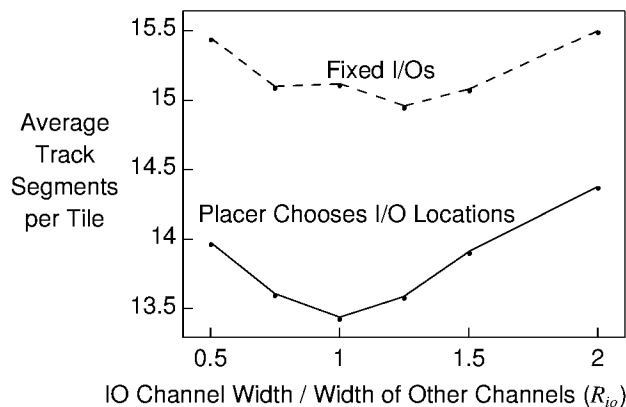
Figure 12: Routability vs. I/O Channel Width.

the placement tool has its minimum value when $R_{io} = 1$, again showing that it is best to spread routing resources evenly across the chip. Fixing the I/O pins shifts the minimum in the tracks per tile curve slightly so that it now occurs when $R_{io} = 1.25$. While fixing the I/O pins leads to a significant increase in the number of routing tracks required, this increase is, for the most part, spread over the FPGA and not confined to the channels connecting to the I/O pads. Consequently, one should not make very wide channels adjoining the pads in order to improve routability with pin constraints, although a small increase in the I/O channel capacity is a net benefit.

6 Conclusions

The most interesting (and unexpected) conclusion of this work is that the most area-efficient global routing structure is one with completely uniform channel capacities across the entire chip and in both horizontal and vertical directions. The basic reason is that most circuits "naturally" tend to have routing demands which are evenly spread across an FPGA. The only (slight) exception we found to this "uniform is better" rule occurred when the I/O locations of circuits were fixed by board-level constraints. In this case making the I/O channel 25% wider than the other channels was a net benefit.

Of almost equal note, the area-efficiency is decreased only slightly by some non-uniform or directionally-biased architectures, provided the pin placement on the logic blocks is well-matched to the channel capacity distribution. Hence if such architectures are desirable for other reasons the impact on core area doesn't preclude their use. For example, one reason for widening the center-most channel is to provide the extra routing required by a larger FPGA while reusing the basic tile layout created for smaller members of the FPGA family.

More specifically, of the FPGA architectures studied, a full-perimeter pin position FPGA with no directional routing bias and uniform channel widths is most area-efficient. Employing a logic block with the top/bottom pin position requires approximately 8% more routing resources than full-perimeter FPGAs, and the most area-

efficient top/bottom FPGA has twice as many horizontal routing tracks as vertical ones. We also found that one can construct rectangular FPGAs which are only slightly less dense than square FPGAs provided one adjusts the degree of directional bias in the routing resources to best match the chip aspect ratio.

Our experimental results in this paper were gathered with the linear congestion cost function in the placement tool because we felt the non-linear cost function was too slow to be commercially viable. However, it is interesting to note that while the non-linear function improved the routability of circuits for all FPGA architectures, it improved routability the most for uniform routing architectures. Apparently it is easier for advanced CAD tools to spread out congested regions than it is to localize them to designated portions of a chip that have extra routing resources. Consequently, we expect that future advances in CAD tools will tend to slightly increase the advantages of uniform routing architectures over their non-uniform counterparts.

References

[1] S. D. Brown, R. J. Francis, J. Rose, and Z. G. Vranesic, *Field-Programmable Gate Arrays*, Kluwer Academic Publishers, 1992.

[2] Xilinx Inc., *The Programmable Logic Data Book*, 1994.

[3] AT & T Inc., *ORCA Datasheet*, 1994.

[4] Actel Inc., *FPGA Data Book and Design Guide*, 1994.

[5] Altera Inc., *Data Book*, 1993.

[6] B. K. Britton, et al., "Second Generation ORCA Architecture Utilizing 0.5μ Process Enhances the Speed and Usable Gate Capacity of FPGAs," *IEEE Int. ASIC Conf.*, Sept. 1994, pp. 474-478.

[7] D. Tavana, W. Yee, S. Young, and B. Fawcett, "Logic Block and Routing Considerations for a New SRAM-Based FPGA Architecture," *CICC*, 1995, pp. 24.6.1 - 24.6.4.

[8] S. Yang, "Logic Synthesis and Optimization Benchmarks, Version 3.0, *Tech. Report*, Microelectronics Centre of North Carolina, 1991.

[9] E. M. Sentovich et al, "SIS: A System for Sequential Circuit Analysis," *Tech. Report No. UCB/ERL M92/41*, University of California, Berkeley, 1992.

[10] J. Cong and Y. Ding, "FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs," *IEEE Trans. Computer-Aided Design*, Jan. 1994, pp. 1-12.

[11] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, May 13, 1983, pp. 671 - 680.

[12] C. E. Cheng, "RISA: Accurate and Efficient Placement Routability Modeling," *ACM Design Automation Conf.*, 1994, pp. 690 - 695.

[13] C. Ebeling, L. McMurchie, S. A. Hauck and S. Burns, "Placement and Routing Tools for the Triptych FPGA," *IEEE Trans. on VLSI*, Dec. 1995, pp. 473 - 482.

[14] C. Y. Lee, "An Algorithm for Path Connections and its Applications," *IRE Trans. Electron. Comput.*, Vol. EC-10, 1961, pp. 346 - 365.

[15] V. Betz and J. Rose, "On Biased and Non-uniform Global Routing Architectures and CAD Tools for FPGAs," Technical Report, University of Toronto, 1996.

An Architecture for Electrically Configurable Gate Arrays

ABBAS EL GAMAL, SENIOR MEMBER, IEEE, JONATHAN GREENE, JUSTIN REYNERI,
ERIC ROGOYSKI, KHALED A. EL-AYAT, AND AMR MOHSEN, SENIOR MEMBER, IEEE

Abstract — An architecture for electrically configurable gate arrays using a two-terminal anti-fuse element is described. The architecture is extensible, and can provide a level of integration comparable to mask-programmable gate arrays. This is accomplished by using a conventional gate array organization with rows of logic modules separated by wiring channels. Each channel contains segmented wiring tracks. The overhead needed to program the anti-fuses is minimized by an addressing scheme that utilizes the wiring segments, pass transistors between adjacent segments, shared control lines, and serial addressing circuitry at the periphery of the array. This circuitry can also be used to test the device prior to programming and observe internal nodes after programming. By providing sufficient wiring tracks segmented into carefully chosen lengths and a logic module with a high degree of symmetry, fully automated placement and routing is facilitated.

I. INTRODUCTION

MASK-programmable gate arrays offer the architectural flexibility and efficiency to integrate thousands of gates, but require long development time and high nonrecurring engineering costs. On the other hand, the convenience of field programming is available with programmable logic device (PLD) technologies, but their architectures have not allowed integration of a wide variety of applications exceeding a few hundred gates [1], [2].

We describe a novel gate array architecture [3] which combines the flexibility of mask-programmable arrays with the convenience of field programmability. Its implementation is made possible by a two-terminal electrically programmable anti-fuse offering low resistance in its conducting state and small area.

The architecture supports a design style similar to conventional gate arrays, including fully automatic placement and routing algorithms attaining 85–95-percent utilization. This required considerable emphasis on symmetry and routability, which we touch on below.

The anti-fuse is so called because it irreversibly changes from high to low resistance when "blown" by applying a programming voltage across it. The anti-fuse, or fuse for short, has an ON-state resistance of approximately 500 Ω. The layout area of the fuse cell is generally limited by the
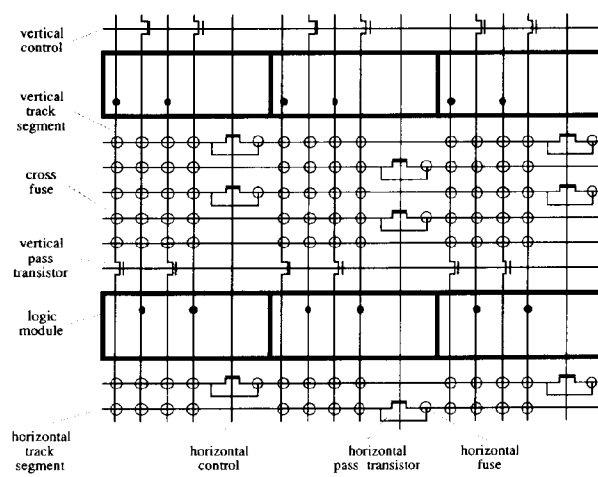
Fig. 1.   Interconnect architecture.

pitch of the first- and second-level metal lines that connect to it; it is about the same size as a via.

This paper focuses on the architecture itself, which is fairly independent of the exact details of the particular CMOS technology and the anti-fuse. Other papers describe more fully the anti-fuse [4], a CMOS circuit implementing the architecture [5], and a study comparing the architecture's logic density to that of conventional gate arrays [6].

II. PROGRAMMABLE INTERCONNECT ARCHITECTURE

The general architecture, shown in Fig. 1, exhibits the familiar gate array organization: rows of logic cells interspersed with routing channels. There are, of course, several key differences.

The tracks in the channels are not simply empty areas in which metal lines can be arranged for a specific design. Rather, they contain predefined wiring "segments" of various lengths. Other wiring segments pass through the channels vertically. Each input and output of a logic module is connected to a dedicated vertical segment. Other vertical segments just pass through the modules, serving as feedthroughs between channels. (The number and lengths of segments in Fig. 1 are only suggestive.)
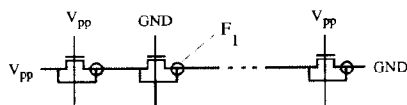
Fig. 2.   Horizontal fuse programming.

Fig. 3.   Cross-fuse programming.

Fig. 4.   A sneak path.

TABLE I

macro	4 transistor cells	modules
3 input NOR	2	1
4:1 mux, non-inverting	6	1
D latch with clear	4	1
D flip-flop with clear/set	7	2
full adder	10	2
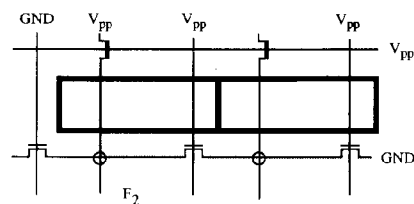
A fuse is located at each crossing of a horizontal and vertical segment. Programming one of these "cross fuses" provides a low-resistance bidirectional connection between the segments. Other fuses are located between adjacent horizontal segments within a track. When blown, these "horizontal fuses" connect the two segments to form a longer one. (Although not shown in the diagram, fuses may also be provided to connect adjacent vertical segments.)

In order to program a fuse, we need to apply high voltage across it. This is accomplished by an efficient addressing scheme that uses the wiring segments themselves, pass transistors connecting adjacent segments, and control logic at the periphery of the array. Fuse addresses are shifted into the chip serially.

As shown in Fig. 1, each column of "horizontal pass transistors" connecting horizontal tracks is controlled by a shared "horizontal control" line running across the array. Each row of "vertical pass transistors" is controlled by a "vertical control" line. The peripheral circuitry can drive the control lines and the segments at the end of each track.

Horizontal fuse programming is quite simple. In the example of Fig. 2, we apply programming voltage V_{PP} across the fuse F_1. All horizontal control lines except the one in the column containing F_1 are turned on by connecting them to V_{PP}, and the appropriate track segments are driven to GND and V_{PP} as shown. (Vertical fuses, if present, are programmed similarly.) Cross-fuse programming uses both horizontal and vertical control lines as shown in Fig. 3. Segments not driven to either GND or V_{PP} are left precharged to $V_{PP}/2$. Thus the voltage across fuses not being programmed is either zero or $V_{PP}/2$.

Some care is required to assure that a unique fuse is addressed. Fig. 4 shows how previously blown fuses can divert current along a "sneak path," in this case programming fuse F_5 through previously blown fuses F_3 and F_4 instead of programming F_2. Fortunately, we are not interested in blowing an arbitrary pattern of fuses (this is not a PROM!). For example, we are not concerned with programming a pattern that connects two outputs together since this does not form a useful net. If we consider only relevant patterns, it can be shown that programming the
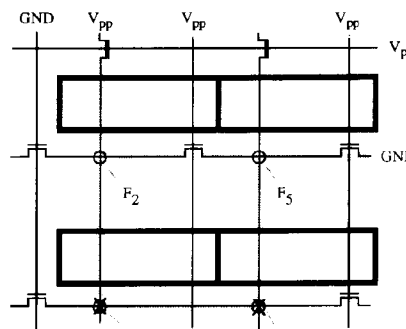
fuses in a carefully chosen order eliminates sneak paths. In general, fuses must be programmed starting from the center of the chip and moving outward, channel by channel. Determining the proper order is a bin sort operation, and can be done by software in linear time.

The pass transistors and peripheral control logic are also used to test the chip; this is discussed in detail later.

III.   CHOICE OF THE LOGIC MODULE

As outlined so far, the programmable interconnection architecture could be used with a variety of logic modules. Which would be best? This turned out to be a very difficult question, involving subtle trade-offs among routability, the logical capability of the module as perceived by the user, and delays due to capacitive loading in the routing segments.

The complexity of the module must be balanced with the routing overhead. Mask-programmed gate arrays provide very flexible and efficient routing. They therefore use a simple four-transistor cell. On the other hand, routing is very expensive in both area and delay with present programmable logic arrays. These generally use a module capable of implementing more complex functions [2]. The architecture outlined here has a cost of routing closer to a conventional gate array, suggesting a logic module of intermediate size. Because this is about the same complexity as conventional gate array hard macros, the designer can use a library like the familiar gate array cell libraries; there is no need to map logic into a more complex module. Table I lists several typical gate array macros and the numbers of four-transistor cells and logic modules required to implement them.
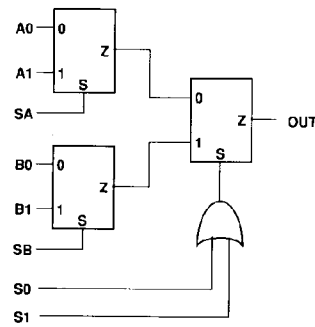
Fig. 5. Module function.

Fig. 6. A routing path.

Our chosen module, shown in Fig. 5, has eight inputs and a single output. It is composed of three two-to-one multiplexors, with an OR gate on the last stage's select input. Various macros, such as those in the table, are implemented by using an appropriate subset of the inputs and tying the remaining inputs high or low. Thus the module can implement all macros with two inputs, most with three inputs, many with four inputs, etc.

The module's output is connected to a vertical segment spanning several channels. Each input is connected to a short vertical segment spanning one channel. Four of these span the channel above the module, four the channel below. The use of short segments for the inputs reduces parasitic capacitance and hence delay.

Note that each input is accessible from either the channel above or below but not both. At first, this would appear to limit routability compared to a conventional "double-entry" gate array cell, in which signals may enter from either channel. However, there is nearly always more than one way to implement a macro. For example, there may be up to four distinct ways to implement a two-input gate: with both signals connecting to inputs in the top channel, with both signals connecting to inputs in the bottom channel, with one signal in the top and the other in the bottom channel, or vice-versa.

By letting the router choose an implementation that uses inputs accessed from convenient channels, the benefits of full double-entry symmetry are approached or sometimes attained. The degree of symmetry possible for a particular macro m implemented in a given module is reflected in the following measure S:

$$S(m) = \log_2 (N(m))$$

where $N(m)$ is the number of distinct possible implementations of the macro m. Full double-entry symmetry would correspond to a value $S(m)$ equal to the fan-in of the macro. To evaluate the overall symmetry of a module, we average $S(m)$ over the macro library, weighted by relative macro usage $U(m)$ and the fan-in $F(m)$:

$$\frac{\sum\limits_{m} U(m)S(m)}{\sum\limits_{m} U(m)F(m)}.$$
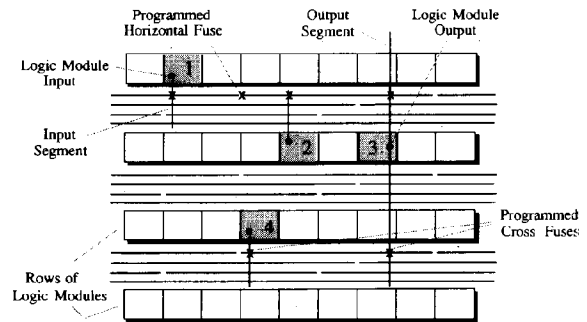
This is the effective fraction of macro inputs in a typical design that have double-entry symmetry, and is an important criterion for choosing a module.

IV. ROUTING

Fig. 6 illustrates the routing of a net. The vertical segment connected to the driving module's output is connected by cross fuses to horizontal segments, which in turn connect to the segments associated with module inputs. In the top channel, a horizontal fuse is used to link two segments into a longer one.

The resistance of the blown fuses and the parasitic capacitance of the segments used form an RC tree, with the driver of the net as the root. Note that each input is driven through a maximum of three and generally two fuses to limit the delay. (If the number of series stages in the RC tree were allowed to increase further, the delay through the routing would increase rapidly.) The maximum number of fuses and the segment lengths (hence capacitances) can be altered to suit the chip dimensions and the resistance of the fuse technology.

In rare instances, it is not possible to place the macros so that all inputs on the net lie within the channels spanned by the output segment of the driving module. To handle this case, a few additional uncommitted long vertical segments are provided. The net is then routed from the output segment to a horizontal segment, then to the long vertical segment, then to another horizontal segment, and finally to the necessary input segments. To keep the number of fuses in series limited to four, no horizontal fuses are allowed in such nets. (If necessary, the architecture can be extended to provide a special fuse connecting the output directly to a long vertical segment passing over the driving module, thus eliminating the first horizontal segment and reducing the total number of fuses in series back to three.)

A means must also be provided to connect internal signals to the bonding pads of the chip. Each pad has a dedicated bidirectional buffer, which connects to the array through an associated I/O module. The I/O modules fit in the outer columns and rows of the array next to the logic modules. Each I/O module has two inputs, data and enable, and an output. The data and enable signals are

sent to the output buffer of the associated bonding pad, and the module's output comes from the input buffer of the pad. Thus the I/O module can be configured to provide input, output, tristate, or bidirectional capability.

To minimize clock skew due to differential routing delay, one entire track (or more if needed) in each channel is set aside for clock distribution. These tracks are connected directly to buffers, so that each input presents a similar load driven through exactly one fuse.

An interesting theoretical question is whether more horizontal tracks are needed in each channel here (where the lengths of the wiring segments must be predetermined) than in mask-programmed routing (where the wiring is customized for the design). Surprisingly, a high probability of routability is obtained with only a few tracks above channel density.

This requires a careful choice of the lengths of the segments, based on statistics from an extensive suite of design examples. This was done by first determining the distribution of net lengths, i.e., the length each net would run along each channel if the constraint of fixed segmentation were absent (as in a conventional gate array). The distribution of physical segment lengths provided on the chip was chosen to obey similar statistics. Then the segmentation was "tuned" manually based on actual routings which obeyed the constraints it imposed.

To obtain good routing performance it is also necessary to take advantage of the symmetry of the macros where possible. For example, observe that if macro 4 in Fig. 6 permits its input to be routed from either the upper or lower channel, there is a better chance of finding a free horizontal segment to connect it.

V. TESTING

To assure high programming yield, it is necessary to thoroughly test the chip for defects in the modules and fuses prior to programming. With a simple addition, the addressing circuitry used for programming suffices for this purpose as well.

Continuity of the tracks is easily verified by turning on all vertical and horizontal pass transistors, and using the peripheral circuits to drive the tracks from one end and read them back from the other. Testing for the absence of shorts between adjacent tracks is done in a similar way by applying a pattern of alternating ZERO's and ONE's.

Shorted or weak cross fuses are detected by turning on all horizontal and vertical pass-transistor lines, grounding all horizontal segments, and driving all vertical segments to some stress voltage. Horizontal fuses are tested column by column, with the same addressing method that is used to program them.

To verify the functional operation of the modules, we need to apply test vectors to their inputs and read their outputs. A vector is applied simultaneously to an entire row of modules by turning on all vertical pass transistors except those in the row being tested. Data are applied to
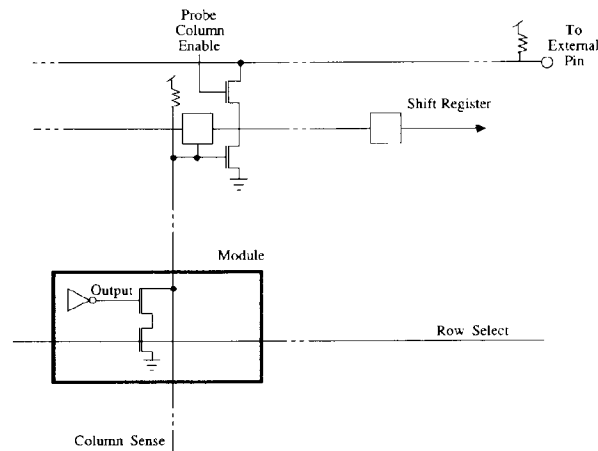
Fig. 7. Probe circuit.

the inputs in the channel above the row from the periphery at the top of the array, and to the inputs in the channel below the row from the bottom of the array.

Since the outputs of the modules share a vertical track with outputs of other modules above and below them, some other means is required to read the module outputs at the array periphery. As shown in Fig. 7, a select line is provided along each row of modules, and a sense line along each column. Activating the select line for the row of modules under test gates their output values onto the sense lines. The sense lines are loaded into a shift register at the top of the array.

This ability to read the output of any module at the array periphery is highly useful *after* programming as well. Only a small amount of extra circuitry is required to select one of the sense lines and make its value available at an external pin of the chip. Thus by shifting in the appropriate address, the user can observe any internal node of his design externally in real time. This virtual probe can be used and its address changed even as the programmed chip is operating in the user's system.

VI. IMPLEMENTATION: SILICON AND SOFTWARE

The architecture has been implemented in a CMOS device. For details, including the speed of the module in isolation and in an application, see [5].

Computer-aided design tools have been developed to support the architecture. Designs are entered as schematics or net lists using a cell library.

The placement and routing algorithms are specific to the architecture. As usual these are time consuming, taking up to a few hours on a low-cost workstation. They achieve 100-percent routing completion. (Even expert users have never been able to improve manually on the automatic router.) The probability of successful routing can be predicted by analyzing some statistics of the design.

Because the nets are *RC* trees, delays are not a simple function of capacitive load as with mask-programmed gate

arrays. Nevertheless, we are able to quickly calculate precise delays at each input for post-layout simulation and timing verification.

ACKNOWLEDGMENT

The authors gratefully acknowledge the technical contributions of J. Chang, D. Gluss, R. Guo, D. How, and F. Sohail.

REFERENCES

[1] S. Wong, H. So, C. Hung, and J. Ou, "CMOS erasable programmable logic with zero standby power," in *ISSCC Dig. Tech. Papers*, Feb. 1986, pp. 242–243.

[2] H. Hsieh *et al.*, "A second generation user programmable gate array," in *Proc. Custom Integrated Circuits Conf.*, May 1987, pp. 515–521.

[3] A. El Gamal, K. El-Ayat, and A. Mohsen. "Programmable interconnect architecture," pending U.S. patent.

[4] E. Hamdy *et al.*, "Dielectric based antifuse for logic and memory ICs," in *IEDM Tech. Dig.* (San Francisco, CA), 1988, pp. 786–789.

[5] K. El-Ayat *et al.*, "A CMOS electrically configurable gate array," in *ISSCC Dig. Tech. Papers*, Feb. 1988, pp. 76–77.

[6] B. Osann and A. El Gamal, "Compare ASIC capacities with gate array benchmarks," *Electron. Des.*, vol. 36, no. 23, pp. 93–98, Oct. 13, 1988.

Abbas El Gamal (S'71–M'73–SM'83) received the B.Sc. degree in electrical engineering from Cairo University, Egypt, in 1972, and the M.Sc. degree in statistics and the Ph.D. degree in electrical engineering both from Stanford University, Stanford, CA, in 1977 and 1978, respectively.

From 1978 to 1980 he was an Assistant Professor of electrical engineering at the University of Southern California, Los Angeles. Since 1980 he has been with the Electrical Engineering Department of Stanford University where he is currently an Associate Professor. From 1984 to 1986 he was Director of the Systems Research Laboratory, LSI Logic Corporation, Milpitas, CA. He is a co-founder and Chief Scientist of Actel Corporation, Sunnyvale, CA.
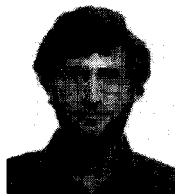
Jonathan Greene received the Sc.B. degree in biology from Brown University, Providence, RI, and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, in 1983, where he performed research on configurable VLSI arrays, VLSI complexity, and information theory.

During 1984 he was with Hewlett-Packard Laboratories. From 1984 to 1986 he worked on cell design automation and module compilation at the LSI Logic Systems Research Laboratory in Palo Alto, CA. He is currently Manager of System Architecture at Actel Corporation, Sunnyvale, CA.

Justin Reyneri received the B.S. and M.S.E.E. mathematics degrees from Stanford University, Stanford, CA, in 1978, and the Ph.D. degree in electrical engineering, also from Stanford, in 1985, having done research in cryptology and information theory.

He is now Manager of System Development at Actel Corporation, Sunnyvale, CA, where he has worked since 1986. Prior to that he worked at LSI Logic's System Research Laboratory on automated data-path layout, and at Hellman Associates on communications security systems.

Eric Rogoyski received the B.S. degree in mathematics from the State University of New York at Stony Brook in 1972.

He was at IBM Corporation from 1974 to 1982 with his last position in EDS at East Fishkill, NY. He held various positions in physical design from 1982 to 1984 at the company he co-founded, California Automated Design Inc., and from 1984 to 1986 at Mentor Graphics Corporation. He is currently a software consultant at Actel Corporation, Sunnyvale, CA, supporting the architectural development and physical design of Actel's configurable technology.

Khaled A. El-Ayat received the B.Sc. degree in electrical engineering from the University of Cairo, Egypt, in 1968, the M.Sc. degree in electrical engineering and computer science from the University of Toronto, Canada, in 1971, and the Ph.D. degree in electrical engineering and computer science from the University of California, Santa Barbara, in 1977.

In May 1977 he joined Intel Corporation, Santa Clara, CA, to work in the Microprocessor Design Group, where he worked on the definition and development of industry standard microprocessor families such as 8086, 80186, and 80386. As a Project Manager at Intel, he was responsible for the design of the control structures of the 80386 Microprocessor. After leaving Intel, he cofounded Actel Corporation in Sunnyvale, CA, and was the Program Manager responsible for development of electrically configurable gate arrays. His present research interests include configurable logic and application-specific architectures, VLSI design, and microprocessor architectures. He has authored many articles and holds patents covering Actel's architecture and testability techniques. Presently he is a Chief Engineer working on the definition of the next generation of products.

Amr Mohsen (S'72–M'74–SM'84) received the Ph.D. degree in electrical engineering and applied physics from the California Institute of Technology, Pasadena.

He is the founder, President, and Chief Executive Officer of Actel Corporation, Sunnyvale, CA, and has more than 20 years of experience in the semiconductor industry. Before founding Actel, he was a Senior Engineering Manager in the technology division at Intel Corporation. He also worked on charge-coupled device development at Bell Laboratories and served as a consultant. He has authored more than 45 articles relating to semiconductors and is responsible for inventions covered by 20 patents.

A Tight Layout of the Butterfly Network

Aythan Avior[1]   Tiziana Calamoneri[2]   Shimon Even[3]
Ami Litman[4]   Arnold L. Rosenberg[5]

Abstract. We establish upper and lower bounds on the layout area of the butterfly network, which differ only in low-order terms. Specifically, the N-input, N-output butterfly network can be laid out in area $(1 + o(1))N^2$, while no layout of the network can have area smaller than $(1 - o(1))N^2$. These results improve both the known upper bound and the known lower bound on the area of butterfly network layouts.

---

[1] Computer Science Department, Technion, Haifa 3200, Israel [`aythan@cs.technion.ac.il`]

[2] Computer Science Department, University of Rome "La Sapienza", Via Salaria 113, 00198 Roma, Italy [`calamo@dsi.uniroma1.it`]

[3] Computer Science Department, Technion, Haifa 3200, Israel [`even@cs.technion.ac.il`]

[4] Computer Science Department, Technion, Haifa 3200, Israel [`litman@cs.technion.ac.il`]

[5] Computer Science Department, University of Massachusetts, Amherst, MA 01003, USA [`rsnbrg@cs.umass.edu`]

1   Introduction

1.1   Overview

Layouts of graphs on rectilinear grids are of wide interest in the study of the VLSI layout problem for integrated circuits [13], as well as in the study of algorithms for drawing graphs. Further, each such layout is a restricted form of embedding of a graph in the grid, hence contributes to the study of the mapping problem for parallel architectures [2, 5], particularly the problem of mapping parallel programs onto mesh-structured parallel architectures; cf. [12].

The fields of graph embedding and VLSI layout have developed powerful techniques which produce embeddings and layouts which are quite efficient—often within constant factors of optimal; cf. [3, 4]. However, even a modest constant factor may render an asymptotically optimal layout or embedding unacceptably inefficient in practice. This observation motivates the current paper.

The aim of this paper is to find a grid-layout of the *butterfly network* [3] whose deviation from optimality is of lower order than a constant factor. We achieve this goal by presenting, in Section 2, a layout of the N-input, N-output butterfly network whose area is $(1 + o(1))N^2$, and by proving, in Section 3, that no layout of this network can have area smaller than $(1 - o(1))N^2$. Thus, our upper and lower bounds coincide, up to a low-order additive term.

Both the upper- and lower-bound components of our result improve prior bounds for butterfly network layouts. The previously known lower bound for the layout area of the N-input, N-output butterfly network was $\frac{1}{4}N^2$ [13, 14]. The 1981 upper bound of $2N^2$ for the same problem [15] was improved only in 1992, to $\frac{11}{6}N^2$ [6].

1.2   The Formal Setup

A. The Graphs of Interest

We define several graphs that are germane to our study. We begin with the two graphs that are of primary interest, butterfly networks and grids.

Butterfly networks. For each integer n, the *n-level butterfly network* $\mathcal{B}_n$ has node-set[1]

$$\{0, 1, \ldots, n\} \times \{0, 1\}^n.$$

---

[1] $\{0, 1\}^n$ denotes the set of length-n binary strings.

For each $0 \leq \ell \leq n$, the set $\{\ell\} \times \{0,1\}^n$ is the ℓth *level* of $\mathcal{B}_n$. The nodes at level 0 of $\mathcal{B}_n$ are called *inputs*, and those at level n are called *outputs*.[2] The string $x \in \{0,1\}^n$ is the *position-within-level string* (*PWL string*, for short) of node $\langle \ell, \ x \rangle$. Each node

$$\langle \ell, \ \beta_0 \beta_1 \cdots \beta_{\ell-1} \beta_\ell \beta_{\ell+1} \cdots \beta_{n-1} \rangle$$

on level ℓ ($0 \leq \ell < n$; each $\beta_i \in \{0,1\}$) of $\mathcal{B}_n$ is connected by a *level-ℓ straight-edge* with node

$$\langle \ell + 1, \ \beta_0 \beta_1 \cdots \beta_{\ell-1} \beta_\ell \beta_{\ell+1} \cdots \beta_{n-1} \rangle$$

on level $\ell + 1$, and by a *level-ℓ cross-edge* with node

$$\langle \ell + 1, \ \beta_0 \beta_1 \cdots \beta_{\ell-1} \bar{\beta}_\ell \beta_{\ell+1} \cdots \beta_{n-1} \rangle$$

on level $\ell + 1$. When we draw $\mathcal{B}_n$ level by level, in such a way that, at each level, the PWL strings are the reversals of the binary representations of the integers $0, 1, \ldots, 2^n - 1$, in that order[3], we get the familiar drawing of $\mathcal{B}_n$ shown in Figure 1(a).

(a)   (b)
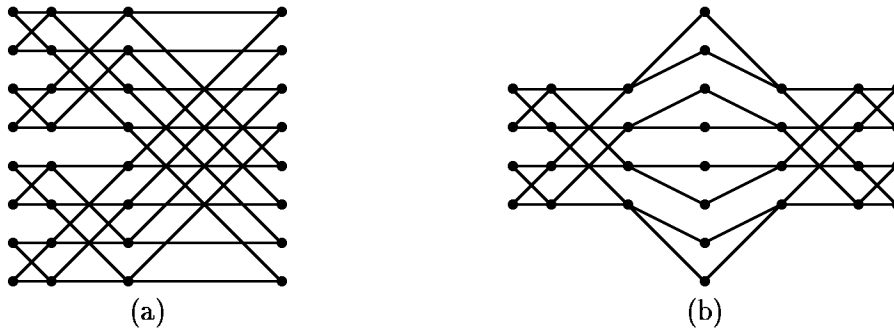
Figure 1: (a) $\mathcal{B}_3$; (b) another view of $\mathcal{B}_3$.

The following folklore lemma will be useful in our layout of $\mathcal{B}_n$ [7].

Lemma 1.1 *For any nonnegative integers j, k the subgraph of $\mathcal{B}_n$ induced by the nodes of levels $j, j + 1, \ldots, j + k$ is the disjoint* sum[4] *of 2^{n-k} copies of $\mathcal{B}_k$.*

---

[2]The terms "input" and "output" derive from the fact that the $(n + 1)$-level butterfly network is the data-dependency graph of the 2^n-input Fast Fourier Transform algorithm [1].

[3]We call this the *natural ordering* of the levels of $\mathcal{B}_n$.

[4]We use the term "sum" here, rather than "union" to emphasize that the constituent graphs share neither nodes nor edges.

Grids. For integers m and n, the $m \times n$ *grid* (or, *mesh*) $\mathcal{M}_{m,n}$ has node-set

$$\{1, 2, \ldots, m\} \times \{1, 2, \ldots, n\}.$$

The edges of $\mathcal{M}_{m,n}$ connect nodes $\langle i, j \rangle$ and $\langle i', j' \rangle$ just when $|i - i'| + |j - j'| = 1$. The path induced by the set of nodes $\{i\} \times \{1, 2, \ldots, n\}$ (resp., the set $\{1, 2, \ldots, m\} \times \{j\}$) is the ith *row* (resp., the jth *column*) of $\mathcal{M}_{m,n}$. Row 1 is *the top side* of $\mathcal{M}_{m,n}$; row m is *the bottom side*; column 1 is *the left side* of $\mathcal{M}_{m,n}$; column n is *the right side*. Finally, we call the product mn the *area* of grid $\mathcal{M}_{m,n}$.

Three auxiliary graphs will be used in our study. Two augmented versions of the butterfly network will be useful in constructing our layout of $\mathcal{B}_n$ in Section 2.

Augmented butterfly networks. We denote by $\mathcal{B}'_n$ the network obtained by appending two new nodes, called *output terminals* to each output of $\mathcal{B}_n$; see Figure 2(a). We denote by $\mathcal{B}''_n$ the network obtained by appending two new nodes, called *input terminals* to each input of $\mathcal{B}'_n$; see Figure 2(b). We refer to the input terminals collectively and to the output terminals collectively as a *terminal group.*

Figure 2: (a) $\mathcal{B}'_2$; (b) $\mathcal{B}''_2$.

The following folklore result, a proof of which can be found in [7], indicates that the designations "input" and "output" in our definitions of $\mathcal{B}_n$ and $\mathcal{B}''_n$ are artificial and are useful only as an aid in visualizing the networks. This fact is crucial in our layout strategy.

Lemma 1.2 *There is an automorphism of $\mathcal{B}_n$ which maps each level $0 \leq \ell \leq n$ of the networks onto level $n - \ell$. There is a similar automorphism of $\mathcal{B}''_n$, which also maps the input terminals onto the output terminals and vice versa.*

Our lower bound argument in Section 3 uses the following well-known auxiliary graph.

Complete bipartite graphs. The $N \times N$ *complete bipartite graph* $\mathcal{K}_{N,N}$ has N input nodes $V^{(i)}$ and N output nodes $V^{(o)}$; its edges connect every input $u \in V^{(i)}$ with every output $v \in V^{(o)}$.

4

B. Graph Embeddings and Grid Layouts

Graph embeddings. An *embedding of graph $\mathcal{G}$ into graph $\mathcal{H}$* (which has at least as many nodes as $\mathcal{G}$) comprises a one-to-one association α of the nodes of $\mathcal{G}$ with nodes of $\mathcal{H}$, plus a *routing ρ* which associates each edge (u, v) of $\mathcal{G}$ with a path in $\mathcal{H}$ that connects nodes $\alpha(u)$ and $\alpha(v)$. The *congestion* of embedding $\langle \alpha, \rho \rangle$ is the maximum, over all edges e of $\mathcal{H}$, of the number of edges of $\mathcal{G}$ whose ρ-routing paths contain edge e.

Grid layouts. We formulate the notion of a *grid-layout* of a graph $\mathcal{G}$ as a special kind of embedding of $\mathcal{G}$ into a grid; there are alternative, equivalent formulations of the notion which make it a special kind of drawing of $\mathcal{G}$ in the plane. Following [13], a *layout* of an N-node graph $\mathcal{G}$ in a grid $\mathcal{M}_{m,n}$, where $N \leq mn$, is an embedding $\langle \alpha, \rho \rangle$ of $\mathcal{G}$ into $\mathcal{M}_{m,n}$ whose routing paths collectively satisfy the following conditions.

- Distinct routing paths are edge-disjoint, so that the embedding that embodies a layout has unit congestion. It follows that at most two routing paths can "cross" at a node of $\mathcal{M}_{m,n}$, i.e., touch the node without terminating there.

- Routing paths that share an intermediate node of $\mathcal{M}_{m,n}$ must *cross* at that node; that is, one path enters the node from the left and leaves toward the right, while the other path enters the node from the bottom and leaves toward the top. Thus, we do not allow "knock-knee" routing [9].

- A routing path may touch no image node $\alpha(u)$, except at its endpoints.

We denote by $AREA(\mathcal{G})$ the area of the smallest grid in which $\mathcal{G}$ can be laid out.

C. Graph Splicing

Our layout of $\mathcal{B}_n$ in Section 2 proceeds by finding layouts of subgraphs of $\mathcal{B}_n$ and "splicing" them together. We now define this intuitive operation formally.

Let us be given a graph $\mathcal{G}$, a graph $\mathcal{H}$, a sequence $\sigma = \langle u_1, u_2, \ldots, u_k \rangle$ of distinct nodes of $\mathcal{G}$, and an equal-size sequence $\sigma' = \langle v_1, v_2, \ldots, v_k \rangle$ of distinct nodes of $\mathcal{H}$. Say that $\mathcal{G}$ has nodes $U \cup \{u_1, u_2, \ldots, u_k\}$ and that $\mathcal{H}$ has nodes $V \cup \{v_1, v_2, \ldots, v_k\}$, where the sets U, V, $\{u_1, u_2, \ldots, u_k\}$ and $\{v_1, v_2, \ldots, v_k\}$ are pairwise disjoint. The operation of *splicing graphs $\mathcal{G}$ and $\mathcal{H}$ along sequences σ and σ'* produces the graph $\mathcal{F}$ whose nodes comprise the set

$$U \cup V \cup \{\langle u_1, v_1 \rangle, \langle u_2, v_2 \rangle, \ldots, \langle u_k, v_k \rangle\}$$

and whose edges connect node w_1 and w_2 just when:

- $\{w_1, w_2\} \subseteq U$ (resp., $\{w_1, w_2\} \subseteq V$), and w_1 and w_2 are adjacent in $\mathcal{G}$ (resp., in $\mathcal{H}$);

- $w_1 = \langle u_i, v_i \rangle$, $w_2 \in U$, and u_i and w_2 are adjacent in $\mathcal{G}$;

- $w_1 = \langle u_i, v_i \rangle$, $w_2 \in V$, and v_i and w_2 are adjacent in $\mathcal{H}$;

- $w_1 = \langle u_i, v_i \rangle$, $w_2 = \langle u_j, v_j \rangle$, and either u_i and u_j are adjacent in $\mathcal{G}$, or v_i and v_j are adjacent in $\mathcal{H}$ (or both).

Examples of splicing. (a) One can splice one copy of $\mathcal{M}_{m,n}$ along its right side to another copy of $\mathcal{M}_{m,n}$ along its left side, to produce an instance of $\mathcal{M}_{m,2n-1}$. (b) One can produce $\mathcal{B}_3$ by appropriately splicing the disjoint sum of two copies of $\mathcal{B}_2$ with the disjoint sum of four copies of $\mathcal{B}_1$; the former sum produces the first two levels of $\mathcal{B}_3$, the latter sum produces the last level of $\mathcal{B}_3$, and the two sums combine to produce the third level of $\mathcal{B}_3$; see Figure 1(a).

Our final example, depicted in Figure 1(b), is so germane to our layout that we encapsulate it as a lemma (whose proof is left to the reader).

Lemma 1.3 *Let the sequence $\sigma = \langle u_1, u_2, \ldots, u_{2^n+1} \rangle$ list all the output terminals of $\mathcal{B}'_n$, in an arbitrary order. Splicing $\mathcal{B}'_n$ with a copy of itself along sequence σ produces $\mathcal{B}_{n+1}$.*

We finally have all the machinery we need to study grid layouts of butterfly networks. To simplify our exposition, let n henceforth be an arbitrary positive integer, and let $N = 2^n$.

2   The Upper Bound on Layout Area

This section is devoted to proving our upper bound on the layout area of $\mathcal{B}_n$.

Theorem 2.1 *For all positive integers n, there is a grid-layout $\mathcal{L}_n$ of $\mathcal{B}_n$ with AREA $(N + o(N))^2$. Hence,*
$$AREA(\mathcal{B}_n) \leq (1 + o(1))N^2.$$

We prove Theorem 2.1 via a sequence of reductions.

6

2.1   The First Reduction

We show how to construct the desired layout of $\mathcal{B}_{n+2}$ from four copies of a suitable layout of $\mathcal{B}_n''$.

Claim 2.1 *One can construct a grid-layout $\mathcal{L}_{n+2}$ of $\mathcal{B}_{n+2}$ with the area indicated in Theorem 2.1, from four copies of a grid-layout $\mathcal{L}_n''$ of $\mathcal{B}_n''$ that has the following properties.*

- $\mathcal{L}_n''$ *places* $\mathcal{B}_n''$ *in a* $(2N + o(N)) \times (2N + o(N))$ *grid* $\mathcal{M}$;

- $\mathcal{L}_n''$ *places one terminal group of* $\mathcal{B}_n''$ *on a vertical side of* $\mathcal{M}$ *and the other terminal group on a horizontal side of* $\mathcal{M}$.

Proof: Assume, with no loss of generality, that the given layout $\mathcal{L}_n''$ places the terminal groups on the bottom and right sides; see Figure 3(a). Flip $\mathcal{L}_n''$ around its right side to produce layout $\widetilde{\mathcal{L}_n''}$ of $\mathcal{B}_n''$. Splice layouts $\mathcal{L}_n''$ and $\widetilde{\mathcal{L}_n''}$ along the pivot line, as depicted in Figure 3(b). By Lemma 1.3, the resulting layout, call it $\mathcal{L}_{n+1}'$, is a layout of $\mathcal{B}_{n+1}'$.
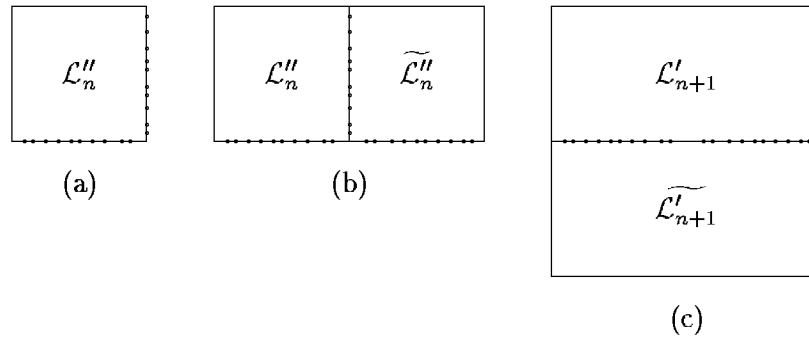
Figure 3: (a) $\mathcal{L}_n''$: a layout of $\mathcal{B}_n''$; (b) $\mathcal{L}_{n+1}'$: a layout of $\mathcal{B}_{n+1}'$; (c) $\mathcal{L}_{n+2}$: a layout of $\mathcal{B}_{n+2}$.

Next, flip layout $\mathcal{L}_{n+1}'$ around its bottom to produce layout $\widetilde{\mathcal{L}_{n+1}'}$ of $\mathcal{B}_{n+1}'$. Splice layouts $\mathcal{L}_{n+1}'$ and $\widetilde{\mathcal{L}_{n+1}'}$ along the pivot line to produce the layout $\mathcal{L}_{n+2}$; see Figure 3(c). By Lemma 1.3, $\mathcal{L}_{n+2}$ is a layout of $\mathcal{B}_{n+2}$. (Note the implicit use of Lemma 1.2 here.) Clearly, layout $\mathcal{L}_{n+2}$ resides in a $(4N + o(N)) \times (4N + o(N))$ grid, hence satisfies the conditions of the Theorem. $\square$

We have thus reduced our layout problem to one of producing a layout $\mathcal{L}_n''$, as used in Claim 2.1.

2.2 The Second Reduction

We now show how to construct the layout $\mathcal{L}_n''$ of Claim 2.1.

Claim 2.2 *Say that we can lay any $\mathcal{B}_m''$ out in a $(2^{m+1} + o(2^m)) \times o(4^m)$ grid, in such a way that each terminal group of $\mathcal{B}_m''$ resides on one of the length-$(2^{m+1} + o(2^m))$ (vertical) sides of the grid. Then one can construct the grid-layout $\mathcal{L}_n''$ of $\mathcal{B}_n''$ described in Claim 2.1.*

Proof: Let the levels of $\mathcal{B}_n''$ be numbered $-1, 0, \ldots, n, n + 1$, where levels (-1) and $(n + 1)$ are the terminal groups. We create our layout of $\mathcal{B}_n''$ in stages.

First, pick any $k \in \{0, 1, \ldots, n - 1\}$, and construct the graph $\mathcal{B}_n^{(k)}$ from $\mathcal{B}_n''$ by placing a new node—called a *token*—on every edge connecting levels k and $k + 1$ of $\mathcal{B}_n''$ (or, equivalently, by replacing every such edge by a length-2 path). Note that $\mathcal{B}_n^{(k)}$ has $n + 4$ levels, numbered $-1, 0, 1, \ldots, n, n + 1, n + 2$, with the tokens residing in level $k + 1$. Clearly, any layout of $\mathcal{B}_n^{(k)}$ is also a layout of $\mathcal{B}_n''$.

Next, decompose $\mathcal{B}_n^{(k)}$ along the token-level into

- $\mathcal{G}_{k,1}$: the induced subgraph of $\mathcal{B}_n^{(k)}$ on levels $-1, \ldots, k + 1$

- $\mathcal{G}_{k,2}$: the induced subgraph of $\mathcal{B}_n^{(k)}$ on levels $k + 1, \ldots, n + 2$.

Easily, one can obtain $\mathcal{B}_n^{(k)}$ by splicing $\mathcal{G}_{k,1}$ and $\mathcal{G}_{k,2}$ along the replicated level. Importantly, by Lemma 1.1, $\mathcal{G}_{k,1}$ is the disjoint sum of 2^{n-k} copies of $\mathcal{B}_k''$, while $\mathcal{G}_{k,2}$ is the disjoint sum of 2^{k+1} copies of $\mathcal{B}_{n-k-1}''$.

For definiteness, let us now assume that n is odd, and let us consider the graphs $\mathcal{B}_n^{(k)}$, $\mathcal{G}_{k,1}$, and $\mathcal{G}_{k,2}$ when $k = (n - 1)/2$. When n is even, we must adjust the details of our layout and its analysis, but only in ways that affect just low-order terms; details are left to the reader. In the case at hand, both $\mathcal{G}_{k,1}$ and $\mathcal{G}_{k,2}$ are disjoint sums of $2^{(n+1)/2}$ disjoint copies of $\mathcal{B}_{(n-1)/2}''$.

Now we are ready to construct the desired layout of $\mathcal{B}_n''$ from the claimed layout of $\mathcal{B}_{(n-1)/2}''$. To this end, let $\mathcal{L}$ be a layout of $\mathcal{B}_{(n-1)/2}''$ in a $(2^{(n+1)/2} + o(2^{n/2})) \times o(2^n)$ grid, in which the terminal groups reside on opposing vertical sides (of size $2^{(n+1)/2} + o(2^{n/2})$).

We first construct a layout $\mathcal{L}^{(1)}$ of $\mathcal{G}_{(n-1)/2,1}$, by abutting[5] $2^{(n+1)/2}$ copies of $\mathcal{L}$, with its token level on the left side, along their (long) horizontal sides. Layout $\mathcal{L}^{(1)}$ resides in a $(2N + o(N)) \times o(N)$ grid.

---

[5]Note that we are *not* splicing these grids: abutting two copies of the $m \times n$ grid along vertical sides creates a copy of the $m \times 2n$ grid.
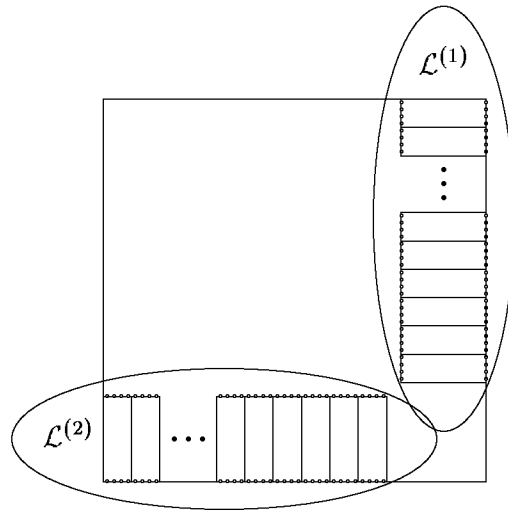
8

Figure 4: Resplicing $\mathcal{L}^{(1)}$ and $\mathcal{L}^{(2)}$.

Next, we rotate layout $\mathcal{L}^{(1)}$ by 90 degrees to produce $\mathcal{L}^{(2)}$, a layout of $\mathcal{G}_{(n-1)/2,2}$ with the token level on the top side.

As the next to last step, we place layouts $\mathcal{L}^{(1)}$ and $\mathcal{L}^{(2)}$ in the smallest grid $\mathcal{M}$ which will hold them in the following non-overlapping configuration. We position layout $\mathcal{L}^{(1)}$ flush with the top and right sides of $\mathcal{M}$, and we position layout $\mathcal{L}^{(2)}$ flush with the left and bottom sides of $\mathcal{M}$; see Figure 4. One verifies easily that a $(2N+o(N))\times(2N+o(N))$ grid is large enough to accommodate this placement of layouts $\mathcal{L}^{(1)}$ and $\mathcal{L}^{(2)}$.

Finally, we must splice $\mathcal{G}_{(n-1)/2,1}$ and $\mathcal{G}_{(n-1)/2,2}$ along the token level, to recreate $\mathcal{B}_n^{((n-1)/2)}$. Since all of the nodes to be "merged" have unit degree, we can accomplish the splicing by routing a specific bijection from nodes on the left side of $\mathcal{L}^{(1)}$ to nodes on the top side of $\mathcal{L}^{(2)}$. Our positioning of layouts $\mathcal{L}^{(1)}$ and $\mathcal{L}^{(2)}$ within $\mathcal{M}$ has left a large unpopulated area (as one can see in Figure 4) in which any such bijection can be routed in a cross-bar fashion.

This completes the layout of $\mathcal{B}_n^{((n-1)/2)}$, hence of $\mathcal{B}_n''$.   □

2.3   The Third Reduction

Our final task is to construct the layouts of $\mathcal{B}_n''$ demanded in Claim 2.2. We construct these layouts implicitly, by appealing to a result of Pinter [10] on channel routing.

9

An (h, l, k) *cross-channel routing problem* involves an $h \times l$ grid (the *channel*) and a set of k *two-point nets*: each net is a pair of gridpoints which reside on opposite vertical sides of the grid. The problem is to construct k edge-disjoint paths which connect every net and which can simultaneously be *laid out* in the grid in the sense of Section 1.2.B. Such a layout may not be possible if the grid is too small; Pinter guarantees that the grid need not be too big.

Lemma 2.1 [10] *Any (h, l, k) cross-channel routing problem satisfying $h > k$ and $l > \frac{3}{2}k + 1$ can be routed within the given grid.*

Lemma 2.1 enables the desired layouts of $\mathcal{B}_n''$ in the following way.

Claim 2.3 *One can lay $\mathcal{B}_n''$ out in an $(2^{n+1} + 1) \times O(n2^n)$ grid, in such a way that each terminal group of $\mathcal{B}_n''$ resides on one of the vertical sides.*

Proof: We place each of the $n+1$ internal levels of nodes of $\mathcal{B}_n''$ in a $(2^{n+1} + 1) \times (2^n + 2)$ grid, in the staggered fashion depicted in Figure 5. That is, the nodes of a level are placed on grid-points of the form $\langle 2i, i+1 \rangle$, where $i = 1, 2, \ldots, 2^n$. (We shall see momentarily that Lemma 2.1 allows us to specify the exact mapping of butterfly nodes to these grid-points in any arbitrary way.)
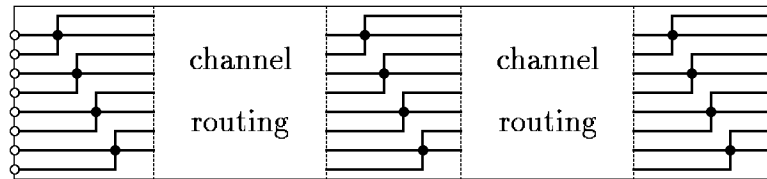
Figure 5: A layout of $\mathcal{B}_2''$.

Now, route four edges out of each node to four "terminals," two on each vertical side of the grid, as depicted in Figure 5. Align the layouts of the $n + 1$ internal levels of $\mathcal{B}_n''$ horizontally, keeping a space of $\frac{3}{2} \, 2^{n+1} + 2$ between them, so that we can apply Lemma 2.1. Easily, this produces the claimed layout of $\mathcal{B}_n''$ in a $(2^{n+1} + 1) \times O(n2^n)$ grid. □

We now have provided all the machinery necessary to create the layout of Theorem 2.1. The overall layout of $\mathcal{B}_n$ implicit in our proof is depicted in Figure 6; the inputs and outputs of the network are on the middle vertical and horizontal lines, respectively; the shadowed areas contain no butterfly nodes, being dedicated to routing butterfly edges. □

10
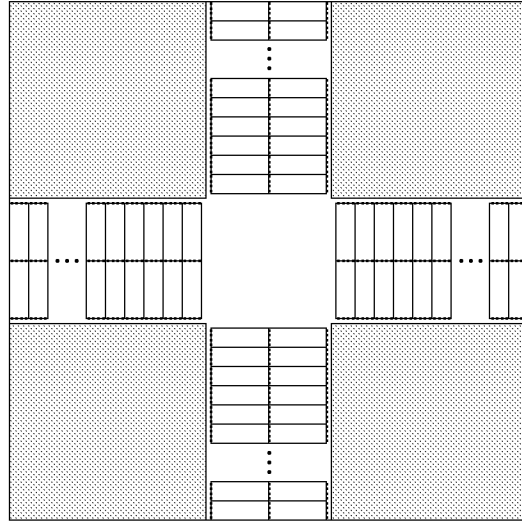
Figure 6: The overall layout of $\mathcal{B}_n$.

3   The Lower Bound on Layout Area

This section is devoted to proving our lower bound on the layout area of $\mathcal{B}_n$.

Theorem 3.1 *For all positive integers n, any grid-layout of $\mathcal{B}_n$ has AREA at least $(2^n - 1)^2$. Hence,*

$$AREA(\mathcal{B}_n) \geq (1 - o(1))N^2.$$

The bounding strategy. We modify the basic lower-bound strategy invented in [13] via the nonstandard notion of a *special-bisection* of a graph.

Let $\mathcal{G}$ be a graph having a designated set of $2c > 0$ *special* nodes. The *minimum special-bisection width* of $\mathcal{G}$, denoted $MSBW(\mathcal{G})$, is the smallest number of edges whose removal partitions $\mathcal{G}$ into two disjoint subgraphs, each containing half of $\mathcal{G}$'s special nodes.

Lemma 3.1 *For any graph $\mathcal{G}$,*

$$AREA(\mathcal{G}) \geq (MSBW(\mathcal{G}) - 1)^2.$$

Proof: We consider an arbitrary layout of $\mathcal{G}$ in $\mathcal{M}_{m,n}$, where, with no loss of generality, $m \leq n$. As in [13], we find that there is a line L that has a single unit-length jog, which can be positioned on a drawing of $\mathcal{M}_{m,n}$ in the following way.

11

- L is aligned with the columns of $\mathcal{M}_{m,n}$ in such a way that the portion of L above the jog lies to the left of some column c of $\mathcal{M}_{m,n}$; the jog of L lies below some row of $\mathcal{M}_{m,n}$; the portion of L below the jog either lies outside of $\mathcal{M}_{m,n}$, or it lies to the right of column c.

- Removing the grid-edges crossed by L yields a special-bisection of $\mathcal{G}$.

By definition, at least $MSBW(\mathcal{G})$ edges of $\mathcal{G}$ must cross line L. By construction, at most $m + 1$ edges of $\mathcal{M}_{m,n}$ cross line L. It follows that $m \geq MSBW(\mathcal{G}) - 1$, whence the lemma.   □

Our goal now is to show that, when we designate the input and output nodes of $\mathcal{B}_n$ as special, we find that $MSBW(\mathcal{B}_n) \geq 2^n$. To this end, we employ $\mathcal{K}_{N,N}$ as an auxiliary graph. First, we note that, if we designate all nodes of $\mathcal{K}_{N,N}$ as special, we obtain the following lower bound on $MSBW(\mathcal{K}_{N,N})$.

Lemma 3.2 $MSBW(\mathcal{K}_{N,N}) = \frac{1}{2}N^2$ *when all nodes of* $\mathcal{K}_{N,N}$ *are special.*

Proof: Consider an arbitrary linearization of the nodes of $\mathcal{K}_{N,N}$. Cut the linearization in half. Say that this cut places K input nodes on one side of the cut, hence $N - K$ on the other. Clearly, the output nodes of $\mathcal{K}_{N,N}$ are cut in exactly complementary proportions. Since $\mathcal{K}_{N,N}$ has an edge between every input and every output, the K inputs and K outputs that are separated by the cut give rise to K^2 edges crossing the cut, while the $N - K$ inputs and $N - K$ outputs that are separated by the cut give rise to $(N - K)^2$ cut edges. We thus have $K^2 + (N - K)^2$ edges of $\mathcal{K}_{N,N}$ crossing the cut. This quantity is minimized when $K = \frac{1}{2}N$, in which case $\frac{1}{2}N^2$ edges cross the cut.   □

Congestion arguments. We employ a technique for bounding unknown $MSBW$'s from known ones, which derives from a technique originated in [8] and refined in [11].

The principle underlying congestion arguments is quite simple. Focus on a graph $\mathcal{H}$ which has k special nodes, whose $MSBW$ we wish to bound from below. Say that we have an auxiliary graph $\mathcal{G}$ which has k special nodes, whose $MSBW$ we know. Say further that we have an embedding ϵ of $\mathcal{G}$ into $\mathcal{H}$, *which maps the special nodes of* $\mathcal{G}$ *onto the special nodes of* $\mathcal{H}$, such that the congestion of ϵ does not exceed C. We claim that $MSBW(\mathcal{H}) \geq (1/C)MSBW(\mathcal{G})$. This inequality holds because the embedding ϵ allows us to view the act of partitioning $\mathcal{H}$ into two disjoint subgraphs having equinumerous sets of special nodes as simultaneously partitioning $\mathcal{G}$ into two disjoint subgraphs having the same partition of special nodes. With this view in mind, one can view the act of

removing any particular edge e of $\mathcal{H}$ as effectively removing all edges of $\mathcal{G}$ that are routed over e by the embedding ϵ. If we know that ϵ never routes more than C edges of $\mathcal{G}$ over any edge of $\mathcal{H}$, which is what our upper bound on the congestion of ϵ means, then we know that cutting an edge of $\mathcal{H}$ simultaneously cuts no more than C edges of $\mathcal{G}$. Since we also know that at least $MSBW(\mathcal{G})$ edges of $\mathcal{G}$ must be cut in order to effect the desired partition of $\mathcal{G}$, we can infer that at least $MSBW(\mathcal{H}) \geq (1/C)MSBW(\mathcal{G})$ edges of $\mathcal{H}$ must be cut in order to effect the desired partition of $\mathcal{H}$. This argument yields:

Lemma 3.3 (The Congestion Lemma)
Let $\mathcal{G}$ and $\mathcal{H}$ be graphs having equal numbers of special nodes. If there is an embedding of $\mathcal{G}$ into $\mathcal{H}$ which maps special nodes to special nodes and which has congestion $\leq C$, then

$$MSBW(\mathcal{H}) \geq (1/C)MSBW(\mathcal{G}).$$

Our lower bound. We obtain our bound via the congestion technique, by letting $N = 2^n$ and analyzing the "natural" embedding of $\mathcal{K}_{N,N}$ (which plays the role of the guest graph $\mathcal{G}$) into $\mathcal{B}_n$ (which plays the role of the host graph $\mathcal{H}$).

Lemma 3.4 *One can embed $\mathcal{K}_{N,N}$ into $\mathcal{B}_n$ with congestion $2^{n-1} = \frac{1}{2}N$, in such a way that the inputs and outputs of $\mathcal{K}_{N,N}$ map, respectively, to the inputs and outputs of $\mathcal{B}_n$.*

Proof: Note that $\mathcal{B}_n$ has the *banyan* property: each input node u is connected to each output node v by exactly one path of length n. Consider any embedding ϵ of $\mathcal{K}_{N,N}$ into $\mathcal{B}_n$ which assigns inputs of $\mathcal{K}_{N,N}$ to inputs of $\mathcal{B}_n$ and outputs of $\mathcal{K}_{N,N}$ to outputs of $\mathcal{B}_n$, and which routes the edges of $\mathcal{K}_{N,N}$ "greedily", via the unique length-n path which connects the two end-points in $\mathcal{B}_n$.

We now analyze the congestion of embedding ϵ. Call a path of $\mathcal{B}_n$ *simple* if it does not visit any level twice. Let e be a level-k edge of $\mathcal{B}_n$. Since $\mathcal{B}_n$ has the banyan property, one endpoint of e reaches precisely 2^{n-k-1} distinct output nodes via simple paths, while the other endpoint of e reaches precisely 2^k distinct input nodes via simple paths. Hence, edge e lies on precisely 2^{n-1} input-to-output simple paths; i.e., its congestion is precisely 2^{n-1}. □

If we now designate all nodes of $\mathcal{K}_{N,N}$ as special and the input and output nodes of $\mathcal{B}_n$ as special, we infer from Lemmas 3.2, 3.3, and 3.4 a lower bound on the $MSBW$ of $\mathcal{B}_n$.

Lemma 3.5 $MSBW(\mathcal{B}_n) \geq 2^n$.

Finally, Lemma 3.5 combines with Lemma 3.1 to yield the desired lower bound, Theorem 3.1, on the area of grid-layouts of $\mathcal{B}_n$. □

13

References

[1] A.V. Aho, J.E. Hopcroft, J.D. Ullman (1974): *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, Mass.

[2] F. Berman and L. Snyder (1987): On mapping parallel algorithms into parallel architectures. *J. Parallel Distr. Comput. 4*, 439-458.

[3] S.N. Bhatt, F.R.K. Chung, J.-W. Hong, F.T. Leighton, B. Obrenić, A.L. Rosenberg, E.J. Schwabe (1996): Optimal emulations by butterfly-like networks. *J. ACM*, to appear.

[4] S.N. Bhatt and F.T. Leighton (1984): A framework for solving VLSI graph layout problems. *J. Comp. Syst. Scis. 28*, 300-343.

[5] S.H. Bokhari (1981): On the mapping problem. *IEEE Trans. Comp., C-30*, 207-214.

[6] Ye. Dinitz (1995): A compact layout of butterfly on the square grid. Tech. Rpt. 873, TheTechnion.

[7] S. Even and A. Litman (1992): Layered cross product — a technique to construct interconnection networks. *4th ACM Symp. on Parallel Algorithms and Architectures*, 60-69.

[8] F.T. Leighton (1983): *Complexity Issues in VLSI: Optimal Layouts for the Shuffle-Exchange Graph and Other Networks*. MIT Press, Cambridge, Mass.

[9] K. Mehlhorn, F.P. Preparata, M. Sarrafzadeh (1986): Channel routing in knock-knee mode: simplified algorithms and proofs. *Algorithmica 1*, 213-221.

[10] R.Y. Pinter (1982): On routing two-point nets across a channel. *19th ACM-IEEE Design Automation Conf.*, 894-902.

[11] A.L. Rosenberg and L.S. Heath (1996): *Graph Separators, with Applications*, in preparation.

[12] L. Snyder (1986): Type architectures, shared memory, and the corollary of modest potential. *Ann. Rev. Computer Science 1*, 289-317.

[13] C.D. Thompson (1980): *A Complexity Theory for VLSI*. Ph.D. Thesis, CMU.

[14] C.D. Thompson (1983): Fourier transforms in VLSI. *IEEE Trans. Comp., C-32*, 1047-1057.

[15] D.S. Wise (1981): Compact layouts of banyan/FFT networks. *VLSI Systems and Computations* (H.T. Kung, B. Sproull, G. Steele, eds.) Computer Science Press, Rockville, Md., 186-195.

A Low-Power Field-Programmable Gate Array Routing Fabric

Mingjie Lin, *Member, IEEE*, and Abbas El Gamal, *Fellow, IEEE*

Abstract—This paper describes a new programmable routing fabric for field-programmable gate arrays (FPGAs). Our results show that an FPGA using this fabric can achieve 1.57 times lower dynamic power consumption and 1.35 times lower average net delays with only 9% reduction in logic density over a baseline island-style FPGA implemented in the same 65-nm CMOS technology. These improvements in power and delay are achieved by 1) using only short interconnect segments to reduce routed net lengths, and 2) reducing interconnect segment loading due to programming overhead relative to the baseline FPGA without compromising routability. The new routing fabric is also well-suited to monolithically stacked 3-D-IC implementation. It is shown that a 3-D-FPGA using this fabric can achieve a 3.3 times improvement in logic density, a 2.51 times improvement in delay, and a 2.93 times improvement in dynamic power consumption over the same baseline 2-D-FPGA.

Index Terms—Field-programmable gate arrays (FPGAs), low-power, performance analysis, routing architecture/fabric.

I. INTRODUCTION

POWER consumption is a primary concern of designing integrated circuits in deeply scaled CMOS technologies. This is especially the case in field-programmable gate arrays (FPGAs) because they are significantly less power efficient than cell-based ASICs [1], [2]. Recent studies have shown that FPGAs are between 9 and 12 times less power efficient than cell-based ASICs [3]–[7]. Today, this power inefficiency problem continues to preclude the use of FPGAs in low power applications, and if not appropriately addressed could undermine future improvements in their logic capacity and delay.

The power consumed in an FPGA core[1] consists of both static and dynamic components. Although static power has been increasing with technology scaling, it constitutes only 10% of the total power consumed in an FPGA today [1], [2]. Furthermore, known techniques for reducing it such as programmable supply voltage, multiple gate oxide thicknesses, multiple transistor threshold voltages, and power gating, have already been applied to FPGAs [1], [2] [8]. Dynamic power, on the other hand, constitutes over 90% of the power consumed in FPGAs and is the main source for their power inefficiency relative to cell-based ASICs. This power inefficiency is caused by the high

[1]In this paper, we do not consider the power consumed in FPGA I/Os.

programming overhead of FPGAs, including the pass-transistor switches, MUXes, buffers, and configuration memory used in the programmable routing fabric, which occupies 50%–80% of the silicon area [3]. This results in much longer routed net lengths relative to cell-based implementations, and hence higher capacitive loading. An even more significant factor than the increase in net length is the loading presented by the programming overhead itself (see Sections II–II-C). As a result of these two factors, 60%–80% of the total FPGA core dynamic power is consumed in the programmable routing fabric [9], [10].

From this discussion, it is clear that reducing interconnect power consumption in an island-style FPGA must involve reducing routed net lengths and/or programming overhead. Routed net lengths can be reduced by using only short interconnect segments in the routing channels. Unfortunately, using only short segments in island-style FPGA only marginally improves power consumption and results in longer delays (see Sections II– II-B). To reduce programming overhead, one way is to decrease the switch box and/or connection box flexibilities. This, however, can decrease routability, increase routed net lengths, and ultimately result in longer delay and higher dynamic power. The main contribution of this paper is showing that significant interconnect power saving can be achieved by re-architecting the programmable routing fabric such that both routed net lengths and programming overhead are reduced without adversely affecting routability or delay.

A. Results Summary and Outline

We present a new FPGA programmable routing fabric that achieves significantly lower power consumption and delay than island-style fabric by using only short interconnect segments and reducing the interconnect loading due to the programming overhead.[2] We show that an FPGA using this fabric, referred to henceforth as *new 2-D-FPGA*, can achieve on average a 1.57 times lower dynamic power consumption and 1.35 times lower geometric average pin-to-pin to delay with only 9% increase in silicon area relative to a baseline island-style FPGA, referred to henceforth as *baseline 2-D-FPGA* (see Sections II–II-A and [12]).

The top level architecture of an FPGA using the new routing fabric is depicted in Fig. 1. It consists of an array of merged *Routing and Logic blocks* with horizontal and vertical routing channel overlay. Only short interconnect segments are used in the routing channels to reduce routed net lengths. The routing block integrates the functions of the connection and switch

[2]A very preliminary version of this paper was partially presented in [11]. This paper provides a more complete discussion of the motivation for developing this fabric and performance comparison for both 2-D and 3-D-FPGA.
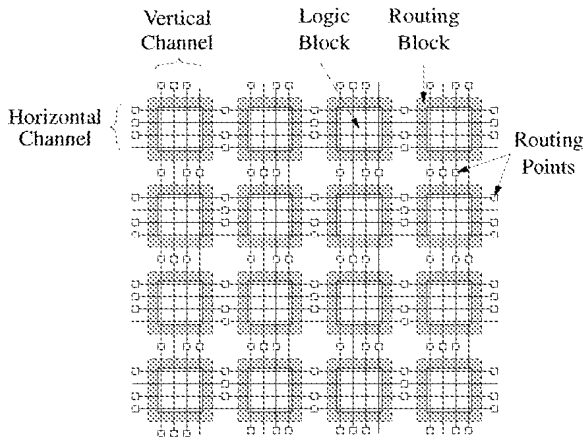
Fig. 1. New routing fabric: Array of logic-routing blocks with horizontal and vertical channel overlay.

boxes in a conventional FPGA. It provides connectivity for logic block inputs and outputs and performs signal bends and fan-outs. The *routing points* are used to: 1) form local connections between neighboring logic blocks (LBs) without going to channels; 2) connect routing block inputs and outputs to channel segments; and 3) chain channel segments together to form longer segments without entering routing blocks. The loading on the interconnect segments is reduced by:

1) using significantly smaller number of pass-transistor switches to connect routing block inputs to logic block inputs than the number used in the connection box of the baseline FPGA;

2) re-architecting the way the logic block outputs connect to the fabric to reduce the loading on logic block output segments and provide direct connectivity between neighboring logic block;

3) eliminating the pass-transistor switches from the interconnect segments in the channels.

As we shall see in Section V, these reductions in programming overhead are achieved without compromising routability.

The architecture of the new routing fabric is also motivated by research on monolithically stacked 3-D-IC, whereby active devices are lithographically built in between metal layers [13], [14]. In addition to achieving higher logic density, vertical stacking reduces interconnect length, hence reducing interconnect delay and power consumption. Considering a 3-D-FPGA where the programming overhead is stacked on top of the LBs (see Fig. 2), a recent study [12] quantified the performance benefits of a monolithically stacked 3-D-FPGA. It was shown that such a 3-D-FPGA can achieve 3.2 times higher logic density, 1.7 times lower critical path delay, and 1.7 times lower dynamic power consumption than a baseline 2-D-FPGA fabricated in the same 65-nm technology node. These improvements were achieved with *no* change in the logic or routing architecture of the baseline 2-D-FPGA. In this paper, we show that a 3-D-FPGA using our new routing fabric, referred to henceforth as *new 3-D-FPGA*, can achieve average improvement of 3.3 times in logic density, 2.38 times in critical path delay, 2.51 times in geometric average pin-to-pin delay, and 2.93 times in dynamic power consumption over the baseline 2-D-FPGA.
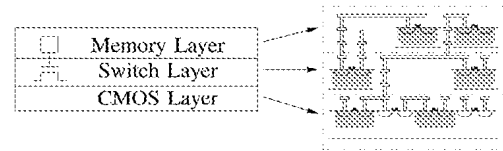

Fig. 2. Monolithically stacked 3-D-FPGA concept.

The rest of the paper is organized as follows. In the following section, we describe the 2-D-FPGA architecture used as a baseline for performance comparison and discuss several factors that motivated the development of the new routing fabric. In Section III, we describe the new routing fabric in details. In Section IV, we describe the CAD flow used for mapping designs into the new FPGA. In Section V, we quantify the performance improvements of the new 2-D-FPGA over the baseline 2-D-FPGA. Finally, in Section VI, we quantify the performance improvements of a monolithically stacked implementation of the new FPGA over the baseline 2-D-FPGA.

II. PRELIMINARIES

We first describe the baseline 2-D-FPGA architecture that we use in explaining the motivation for the new fabric design and in comparing their performance. We then discuss the factors that motivated the development of the new programmable routing fabric.

A. Baseline 2-D-FPGA

We assume an island-style FPGA comprising a 2-D array of LBs interconnected via a programmable routing fabric [15] (see Fig. 3). To ensure the fairness of our performance comparison studies, the architectural parameters of baseline 2-D-FPGA are chosen according to both previous studies [16], [17] and our own investigations using TORCH design tool [18].

- *Logic Block*: We assume a Virtex II style logic block comprising four slices, each consisting of two four-input Lookup Tables (LUTs), two flip-flops (FFs), and programming overhead. Note that the design of our logic block is a simplified version of the Virtex II logic block detailed in [15]. Our choice of cluster and LUT sizes follows the results of [17], where under a fixed routing architecture (using wire segment length 4 and 50% pass transistors and 50% tri-state buffers in routing switches), an island-style FPGA with the cluster size 8 and LUT size 4 consumes the lowest energy. We also assume throughout the paper that the new 2-D and 3-D FPGAs use the same logic block as the baseline 2-D-FPGA.

- *Routing Channel Width*: The routing fabric comprises horizontal and vertical segmented routing channels. The number of routing tracks is determined empirically. In the routability comparison in Sections V–V-A, we use the minimum number of routing tracks needed to successfully route the benchmark circuits in both the new FPGA and the baseline. In the delay and power comparison in Sections V-V-E and V-V-F, we choose the number of routing tracks to be 15% higher than the minimum required to avoid excessive routing congestion [19]–[22].

TABLE I
BASELINE 2-D-FPGA PARAMETER VALUES

Tile Width L	Array Size $N \times N$	LB Buffer Size, b	LB Inputs K_i	LB Outputs K_o	SB Width / Density W, d	CB Flexibility F_c
4100λ	64 or 100	4	32	8	72, 3	0.5

Segment Type	Single	Double	Length-3	Length-6
Number of Tracks	32	36	30	72
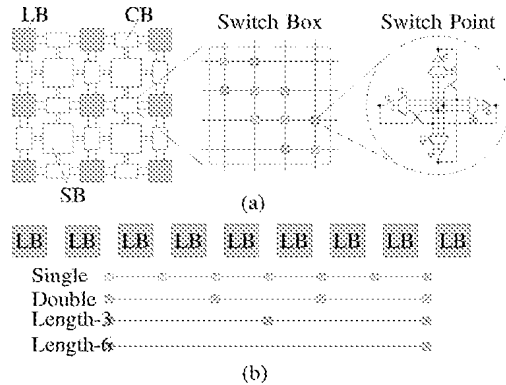
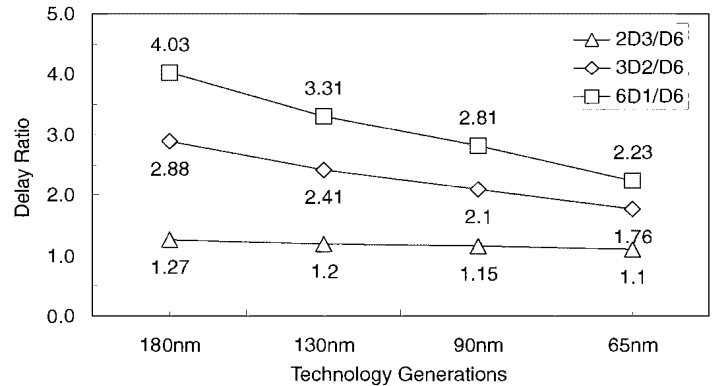Fig. 3. (a) Baseline 2-D-FPGA architecture. (b) Segmented track types used.

Fig. 4. Delay benefits of long interconnect relative to short interconnects of equivalent total length for four technology nodes. Normalized net delays, $6D_1/D_6$, $3 - D_2/D_6$, and $2 - D_3/D_6$, for four technology nodes, where D_i is the delay of a segment of length i tiles, are plotted. The sizes of the buffers for interconnect segments Length-3 and Length-6 are optimized for each technology node as discussed in [12].

• *Routing Channel Segmentation*: The channel segmentation for the baseline FPGA comprises sets of staggered single, double, length-3, and length-6 segmented tracks. Each segment consists of two unidirectional metal wires. The segments can be connected via connection boxes (CBs) only to the inputs and outputs of the first and last LBs it spans. Segments can be connected to each other via switch boxes (SBs). To ensure the quality of routing channel segmentation for the baseline FPGA, we start with a segmentation based on [15] and [23], and use the TORCH routing channel segmentation design tool [18] to optimize it for 65-nm CMOS technology node. The optimization resulted in a 9% improvement in delay-power product over the Virtex II based segmentation.

• *Design of CB and SB*: The design of the CB and SB follow [24] and the design of the switch point is MUX-based as described in [25] [see Fig. 3(a)]. The logic design of a switch point is much more complicated than a pass-transistor switch, which consists of a single transistor controlled by one configuration memory bit. Each LB is assumed to have K_i input pins and K_o output pins that can be connected to routing channel.

Table I lists the key architecture parameter values for the baseline FPGA used in the performance comparisons.

We now discuss various factors that motivated the development of our new routing fabric.

B. *Using Only Short Segments*

FPGA routing fabrics, such as the one used in the baseline FPGA described above, employ a mixture of short and long interconnect segments in order to reduce net delay. Short segments are beneficial to routability but can result in higher net delay. Using long interconnects, however, results in longer

routed net lengths and larger silicon area and hence higher power consumption.

A previous study [12] has argued that the delay advantage of long interconnect segments (Lengths 3 and 6) diminishes as CMOS technology scales below 100 nm due to the much larger increase in wire parasitics relative to transistor parasitics. This observation is demonstrated in Fig. 4, which compares the delay of implementing a net spanning six tiles using 2 Length 3 segments $(2 - D_3)$, 3 Double segments $(3 - D_2)$, or 6 Single segments $(6D_1)$ normalized by the delay of implementing the net in a single Length 6 segment (D_6). Thus, the larger the normalized delay, the greater the benefit of using longer segments. From the figure we see, for example, that the relative delay $6D_1/D_6$ decreases from 4.03 in 180-nm technology to 2.23 in 65-nm technology. As long segments are expensive in area due to large buffers and provide less routing flexibility than shorter segments, their cost effectiveness decreases with technology scaling.

This finding has motivated us to consider routing fabrics with only short segments. In the following routability study, we quantify the improvement in net length of routed circuits using only short segments, in which we consider a baseline 2-D-FPGA with only single and double interconnects, henceforth referred to as *2-D-FPGA(1,2)*. The important point here is that the average segment length is considerably shorter than in the baseline 2-D-FPGA.

In order to make the comparison fair, we equalize the routability of the 2-D-FPGA(1,2) to that of the baseline 2-D-FPGA using the methodology outlined in Sections V–V-B, which results in each channel having 24 single and 96 double
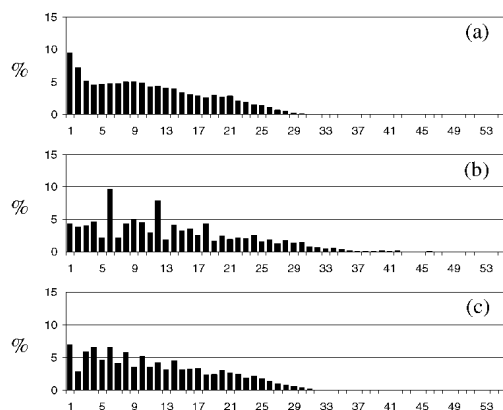
Fig. 5. Routed net length distributions for ALU4 benchmark circuit. (a) Manhattan distance. (b) baseline 2-D-FPGA. (c) 2-D-FPGA(1,2).
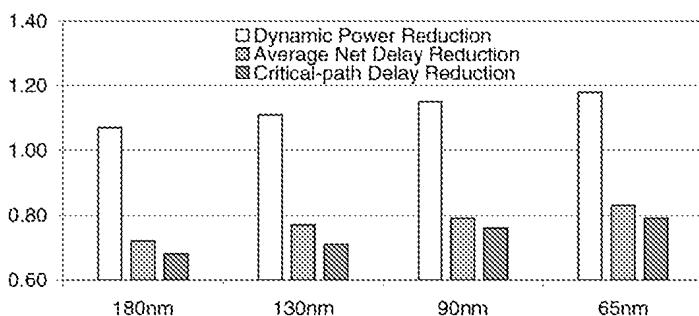

Fig. 6. Average routing fabric power consumption and delay for 2-D-FPGA(1,2) relative to baseline 2-D-FPGA for four technology nodes.

segment tracks. We placed and routed the 20 largest MCNC benchmark circuits [26] in both the baseline 2-D-FPGA and the 2-D-FPGA(1,2). Fig. 5 compares their pin-to-pin net length distributions to the Manhattan distance for the ALU4 benchmark circuit [26]. These results are obtained using timing-driven placement for 2-D-FPGA and separate routing runs for the 2-D-FPGA and the 2-D-FPGA(1,2). Note that the net length distribution in the baseline 2-D-FPGA(1,2) is much closer to that of the Manhattan distance. On average, the net length in the 2-D-FPGA(1,2) is around 16% shorter than that in the baseline 2-D-FPGA. This reduction in net length results in a reduction in dynamic power consumption as illustrated in Fig. 6. Note that the power consumption improves for all technology nodes, and that the improvement factor increases with technology scaling because of the increase in wire parasitics. The improvement in dynamic power, however, comes at the expense of an increase in delay as shown in Fig. 6. As we shall see, the new routing fabric achieves significant improvements in both power and delay relative to the baseline 2-D-FPGA(1,2).

C. Reducing Loading due to Programming Overhead

Interconnect loading in FPGAs is dominated by loading due to the programming overhead. To illustrate this, Fig. 7 plots the fraction of capacitive loading due to the switch point (SP), connection box pass-transistor switches (PTS), and metal wire (MW) in Single and Double segments in the baseline 2-D-FPGA for four technology nodes. Note that even though the fraction

of loading due to metal wire has been increasing with technology scaling, the loading due to the pass-transistor switches and switch point still dominates. In fact, the fraction due to the pass-transistor switches has increased with technology scaling. As such, the design of the new routing fabric focuses on reducing the loading due to the programming overhead as quantified in Sections V–V-D.

D. Improving Routability

While using only short segments should improve routability, reducing programming overhead could have an adverse effect on routability, which could in turn hurt power and delay. The new routing fabric reduces programming overhead without reducing routability. This is achieved through judicious design of the routing point and the routing block, including the extended switching capability of the routing block, as detailed in the Section III.

E. Compatibility With 3-D Implementation

Our routing fabric is also motivated by monolithic stacking. Because the programmable routing (switches and configuration memory) is stacked on top of the logic blocks and buffers and the LB inputs and outputs "come up" to the routing fabric, it is natural to integrate the functionalities of the interconnect and switch boxes associated with each LB into a single routing block. This allows for sharing of resources and further reduction in interconnect loading.

III. NEW ROUTING FABRIC

As discussed in the introduction (see Fig. 1), the new fabric consists of an array of routing blocks (RBs) with horizontal and vertical routing channel overlay.

A. Routing Block

The routing block integrates the functions of both the connection and switch boxes of the island-style FPGA. Fig. 8 shows how the routing block implements the function of a switch box. As in a conventional switch box, the RB function is parameterized by the width W, which is the number of input ports on each side, and the switching width d, which is the number of possible connections for each port [24]. A routing block input line entering at one side connects to inputs of d different MUXes on each of the two perpendicular sides. For example in Fig. 8(b), the input line $i_{1,l}$ enters the left side of the routing block and connects to the inputs of MUXes with outputs $O_{1,t}$, $O_{2,t}$, $O_{3,t}$ on the top side and three MUXes with outputs $O_{1,b}$, $O_{2,b}$, and $O_{3,b}$ on the bottom side. Note that unlike conventional switch box design, our routing block does not include straight connections. Such connections are provided by the interconnect segments in the routing channel overlay as shown in Fig. 12. Each routing block MUX output drives an interconnect segment and/or an input line to a neighboring routing block through a routing point.

Figs. 9 and 10 show how the routing block implements the function of a connection box. Each routing block input can connect to n_i LB inputs via pass-transistor switches as shown in Fig. 9. The value of n_i is chosen so that each LB input can connect to the same number of routing block inputs as the number of interconnect segments an LB can connect to in the baseline
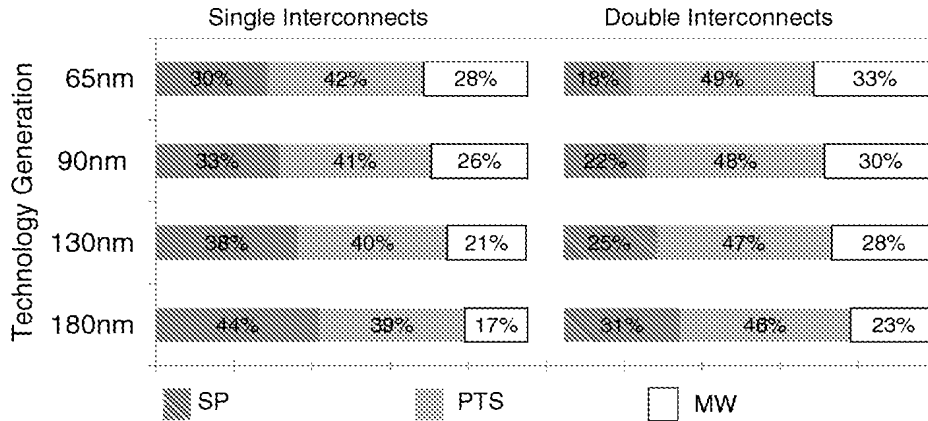
Fig. 7. Delay breakdown of Single and Double interconnect delays between connection box pass-transistor switches (PTS), switch points (SP), and metal wire (MW) for baseline 2-D-FPGA.
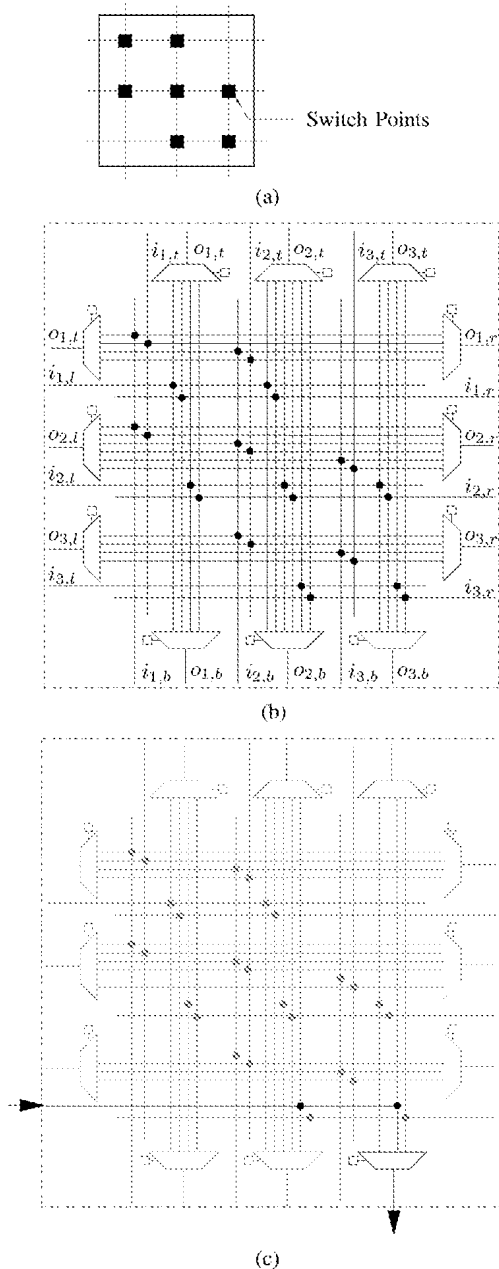

Fig. 8. (a) Switching capability of a routing block. (b) Routing block with $W = 3$ and $d = 3$ (connections to LB inputs and outputs not shown). (c) Example signal turn.
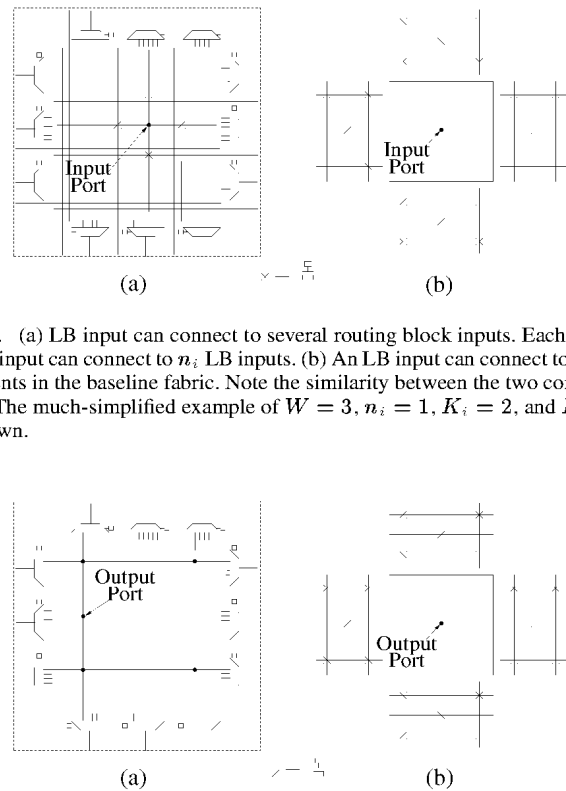

Fig. 9. (a) LB input can connect to several routing block inputs. Each routing block input can connect to n_i LB inputs. (b) An LB input can connect to several segments in the baseline fabric. Note the similarity between the two connectivities. The much-simplified example of $W = 3$, $n_i = 1$, $K_i = 2$, and $K_o = 1$ is shown.

Fig. 10. (a) LB output can connect to n_o routing block MUX inputs on each side. (b) An LB output can connect to segments in the baseline fabric. Note the large difference between the two connectivities. The much-simplified example of $W = 3$, $n_o = 2$, $K_i = 2$, and $K_o = 1$ is shown.

fabric. However, and *bypass.* the loading on a routing block input segment is significantly lower than that on an interconnect segment in the baseline fabric. Thus, we maintain the same connection box flexibility for the LB inputs in our fabric, but with significantly lower segment loading. In Sections V–V-D, we provide a detailed comparison between the loading on a routing block input and on a Single segment in the baseline fabric.

In addition to the connection through switch points, the architecture of the routing block allows for *extended* switching width. As shown in Fig. 11, a signal entering a routing block can loop back twice into it and exit to a perpendicular direction
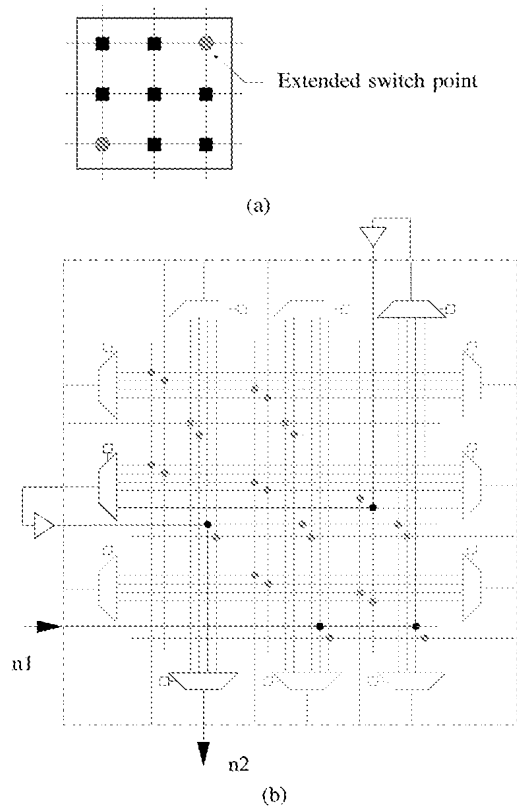
(a)

(b)

Fig. 11. (a) Extended routing capability of routing block. (b) Example of an extended signal connection.

if it cannot do so directly. As we demonstrate later, such extended switching significantly improves routability.

B. Routing Channel Overlay

Each routing channel comprises only Single and Double segmented tracks. Each segment consists of two unidirectional wires. The inputs and outputs of the routing blocks and the channel segments can be programmably connected using the routing points as depicted in Fig. 12. Segments can be chained together to form longer segments, henceforth referred to as *bypass interconnect*, by appropriately setting the states of the routing points without entering routing blocks. The segments can also be connected via routing points to routing blocks to connect to LB inputs and outputs, make bends, or fan-out.

C. Connections Between Routing Block and Routing Channel Overlay

Fig. 10 shows how an LB output is connected to the fabric. Each LB output is connected to n_o different MUX inputs on each side of the routing block. This is quite different from the way LB outputs are connected in the baseline fabric and is a key factor in achieving lower interconnect loading. Note that n_o is chosen to be significantly lower than the number of segments an LB output can connect to in the baseline fabric. This results in a significant reduction in loading on the LB output segment as quantified in Sections V–V-D. Interestingly, this reduction in LB output connectivity does not adversely affect routability (see Sections V–V-A for a more detailed discussion of this key point).
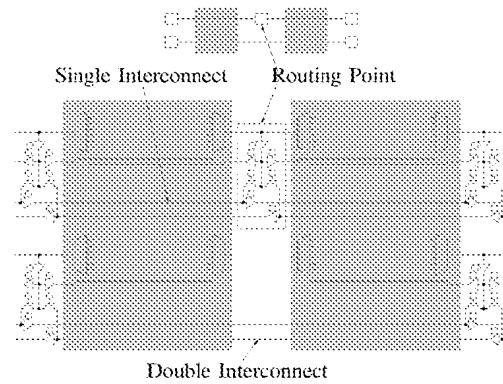

Fig. 12. Single and double interconnects and their connections through routing points to the routing block. Top: symbolic representation from Fig. 1. Bottom: detailed schematic with only horizontal channels shown.
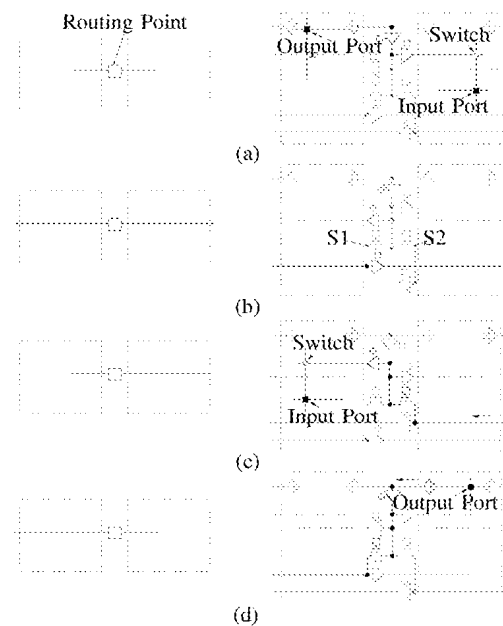

Fig. 13. (a) Connection from LB output to neighboring LB input. (b) Bypass interconnect. (c) Connection from a segment to an LB input. (d) Connection from an LB output to a segment.

The routing fabric allows for two types of net connections: *local* A local connection is used to directly route one output of an LB to an input of its neighboring LB without using routing channel segments. A bypass connection is used to route a longer net without entering intermediate routing blocks. Fig. 13(a) shows how an output of an LB can be directly connected to the input of a neighboring LB without using channel segments. Fig. 13(b) shows how a bypass interconnect is implemented. Note that by turning off the two pass transistor switches S1 and S2, a local connection can be simultaneously made between the two routing blocks. This routing point resource sharing improves routability. Turning off these pass transistors also reduces the loading on the bypass interconnect. Fig. 13(c) and (d), respectively, shows how an LB input and output can be connected to a segment. Note that only buffers that are needed to establish the connection are turned on. This again helps reduce the loading on the connection, thus reducing its delay and power consumption (see Sections V–V-D).
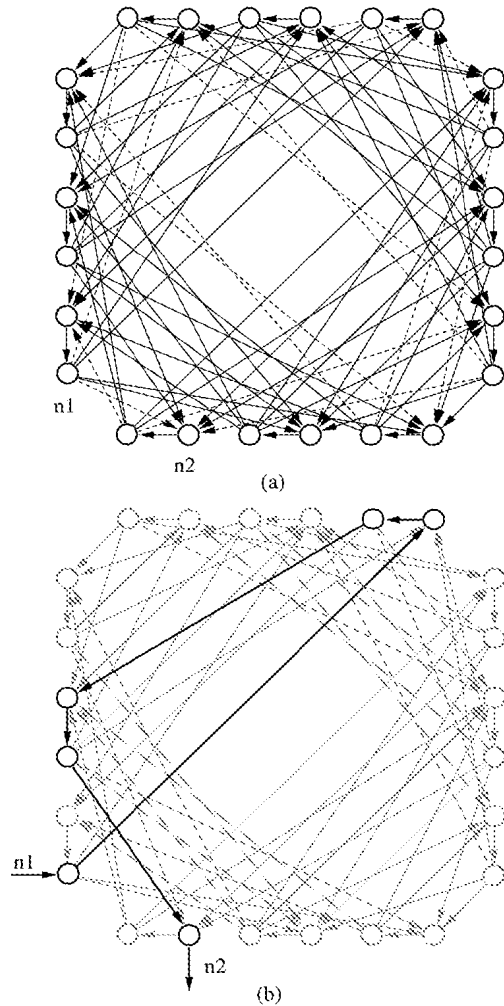
Fig. 14. (a) Routing graph for routing block with $W = 3$ and $d = 3$. (b) Extended connection between n_1 and n_2.

IV. CAD TOOLS

To map designs into the new FPGA, we use the logic packing and placement modules of VPR [27]. We modify the VPR router [27], [28] to accommodate the differences between the routing architecture of our new fabric and the island-style fabric.

Fig. 14(a) shows an example routing graph for a routing block with width $W = 3$ and switching with $d = 3$. Note that d is the number of possible connections from an input port to each of the routing block sides it connects to, which plays a similar role to the parameter F_s in the baseline FPGA [24]. Each routing block input and output is represented by a node (so there are $2W$ nodes on each side). Solid edges correspond to direct connections while dashed edges correspond to extended connections. Fig. 14(b) shows how an extended connection from input node n_1 to output node n_2 is implemented using direct connections. The example corresponds to Fig. 11.

The routing algorithm used is based on the Pathfinder negotiated congestion algorithm in [28]. It is described in Algorithm 1, where:

- A_{ij} is the criticality of the connection from the source of net i to one of its sinks j;

- d_n is the intrinsic delay of routing node n;
- p_n is the present congestion cost of node n.

Initially, nets are routed one at a time using the shortest path it can find without considering interconnect segment or logic block pin overuse. Each iteration of the router consists of sequential net rip-up and reroute according to the lowest cost path available. The cost of using a routing resource is a function of its current overuse and any overuse that occurred in prior routing iterations. By gradually increasing the cost of an oversubscribed routing resource, the algorithm forces nets with alternative routes to avoid using that resource, leaving it to the net that most needs it.

The main difference between this algorithm and VPR is that we keep track of visited nodes during the breadth-first-search to improve the run time. This is described in lines 11, 12, and 22. This modification is needed because the extended switching capability of the routing block explained earlier results in many local cycles in the routing graph.

Algorithm 1 Congestion/Delay Routing Algorithm

1: $A_{ij} \leftarrow 1$ for each signal net i and each sink j
2: **while** shared routing nodes exist **do**
3:   **for all** nets i **do**
4:     rip up routing tree RT_i
5:     initialize the queue PQ
6:     **for all** sinks t_{ij} **do**
7:       enqueue each node n in RT_i at costs $A_{ij}d_n$ to PQ
8:       **while** t_{ij} is not found **do**
9:         dequeue node m with the lowest cost from PQ
10:         **for all** fanout node n of m **do**
11:           **if** node n is unseen **then**
12:             mark node n as seen
13:             enqueue n to PQ with the cost of
              $A_{ij}d_n + (1 - A_{ij})d_np_n$
14:           **end if**
15:         **end for**
16:         **for all** node n in the routed path t_{ij} to s_j **do**
17:           update the cost of node n
18:           add n to RT_i
19:         **end for**
20:       **end while**
21:     **end for**
22:     mark all nodes in PQ as unseen
23:     update A_{ij} for net i
24:   **end for**
25: **end while**

V. NEW 2-D-FPGA VERSUS BASELINE 2-D-FPGA

In this section, we compare a 2-D-FPGA using the new routing fabric to the baseline 2-D-FPGA in terms of routability, programming overhead, logic density, delay, and power consumption. We assume a 65-nm CMOS technology and the Berkeley Predictive Technology Model (BPTM) for devices and interconnect.

A. Routability

To compare the routability of the new fabric to that of the baseline 2-D-FPGA, we placed and routed the 20 largest MCNC benchmark circuits in both architectures. We varied the routing channel width and found the minimum track count $T_{\min}$ for each design mapped to each architecture. Note that, in general, T is greater or equal to the routing block width W. For example, in our baseline FPGA routing channel, we have 32 Single, 36 Double, 30 Length-3, and 72 Length-6 tracks, which corresponds to $T = 170$ and $W = 32 + 36/2 + 30/3 + 72/6 = 72$. In this section, we choose $T_{\min}$ to compare routability between baseline the new and baseline FPGAs. For the new 2-D-FPGA, we assumed that each routing channel contains 28 Single, 64 Double, and 36 Length-3 routing tracks. In varying the channel width we maintained the same fractions of each interconnect segment type. For the baseline, we use a fraction of 0.19 for Single, 0.21 for Double, 0.18 for Length-3, and 0.42 for Length-6 segment tracks. For the new 2-D-FPGA, we maintained a ratio of 0.22, 0.5, and 0.28 for the number of Single, Double, and Length-3 segment tracks, respectively. Table II compares 1) the minimum channel width $T_{\min}$ in segmented tracks for the baseline 2-D-FPGA and the new 2-D-FPGA, 2) the geometric average of pin-to-pin net lengths $\overline{L}$ where the pin-to-pin net length is the sum of segment lengths used in its routing, and 3) the geometric average of the number of bends $\overline{S}$ used to route each pin-to-pin net segment. Note that on average, the new routing fabric requires 50% fewer tracks per channel than the baseline 2-D-FPGA. This is because of the use of only short segments and the additional switching capability of the routing block. These $T_{\min}$ values correspond to a reduction in average routing block width of around 20% over the switch box width in the baseline 2-D-FPGA. The new routing fabric also reduces the average pin-to-pin net segment length by around 16%. Finally, note that the average number of bends $\overline{S}$ increases slightly in the new 2-D-FPGA due to the use of only short segments.

To investigate the improvement in routability due to the extended switching capability of the routing block, we disabled this feature and rerouted the benchmark circuits. We found that the saving in minimum channel width drops from 57% to 34%.

B. Programming Overhead

In this section, we compare the programming overhead of the new 2-D-FPGA to that of the baseline 2-D-FPGA. For the baseline 2-D-FPGA, we assume the architecture parameter values given in Table I. For the new 2-D-FPGA, we assumed that each routing channel contains 28 Single, 64 Double, and 36 Length-3 routing tracks (so $W = 28 + 64/2 + 36/3 = 72$). This choice of routing channel segmentation was obtained using TORCH [18]. Each routing block input line connects to inputs of $d = 3$ MUXes in each of the two perpendicular directions and can be programmably connected to $n_i = 8$ LB inputs through pass transistor switches (see Fig. 9). Equivalently, each LB input can connect to $8 \times 72 \times 4/16/4 = 36$ input lines from each side of the routing block, which is the same as the number of segments an LB can connect to in the baseline 2-D-FPGA. Each LB output can connect to two MUX inputs ($n_o = 2$) on each of the four sides of the routing block (see Fig. 10). As Table II shows, the

TABLE II
ROUTABILITY COMPARISON BETWEEN THE NEW 2-D-FPGA (NEW) AND THE BASELINE 2-D-FPGA (BL). $T_{\min}$ IS THE MINIMUM NUMBER OF SEGMENTED TRACKS NEEDED, $\overline{L}$ IS THE GEOMETRIC AVERAGE OF PIN-TO-PIN NET LENGTHS MEASURED IN THE NUMBER OF LOGIC BLOCKS THE NET SPANS, AND $\overline{S}$ IS THE GEOMETRIC AVERAGE OF THE NUMBER OF BENDS

Circuit	$T_{\min}$		$\overline{L}$		$\overline{S}$	
	BL	NEW	BL	NEW	BL	NEW
alu4	56	24	12.79	11.54	3.61	3.88
apex2	58	31	12.23	10.43	3.23	3.28
apex4	53	36	11.57	9.19	2.79	2.69
bigkey	37	21	19.98	18.22	4.41	5.55
clma	76	42	21.11	17.51	4.37	5.23
des	40	14	18.35	15.63	3.42	3.94
diffeq	39	27	9.13	7.81	2.99	3.01
dsip	32	14	20.77	18.23	4.59	5.11
elliptic	76	36	17.02	13.77	3.34	4.42
ex1010	83	47	13.95	14.56	3.71	4.03
ex5p	75	38	10.19	9.51	2.59	3.07
frisc	83	46	14.91	12.09	3.53	3.87
misex3	65	33	12.01	9.12	3.21	3.32
pdc	112	49	18.98	16.83	3.72	4.17
s298	43	26	14.03	11.03	3.33	3.87
s38417	75	33	10.09	8.23	2.27	3.02
s38584	59	33	12.31	9.79	2.79	3.13
seq	73	35	12.21	10.51	3.21	3.69
spla	94	52	17.72	14.43	3.89	3.81
tseng	43	27	10.78	9.51	3.13	3.41

$T_{\min}$ values for the new 2-D-FPGA correspond to a reduction in average routing block width of around 20% over the switch box width in the baseline 2-D-FPGA. Therefore, $W = 72$ for the new 2-D-FPGA achieves equal or better routability than the assumed $W = 72$ for the baseline 2-D-FPGA.

Table III list the programming overhead per tile for the new 2-D-FPGA versus the baseline 2-D-FPGA. Note that the programming overhead of the routing block, which implements the functions of two connection boxes and a switch box, is lower than that of the switch box by itself. This is because, in the new 2-D-FPGA, 1) the number of pass-transistor switches on an LB input is much lower than that on an interconnect segment in the baseline 2-D-FPGA, 2) there are no pass-transistor switches on LB output segments, and 3) there are no pass-transistor switches on the interconnect segments in the channels.

An important point here is that the number of interconnect segments an LB output can connect to in the new 2-D-FPGA is reduced to 8 from 36 (16 Single, 9 Double, 5 Length-3 and 6 Length-6) in the baseline 2-D-FPGA. However, the number of RB inputs that an LB input can connect to is the same for the new and baseline 2-D-FPGA (14 Single, 16 Double, and 6 Length-3 in the new 2-D-FPGA and 16 Single, 9 Double, 5 Length-3, and 6 Length-6 in the baseline 2-D-FPGA). The fact that an LB input can connect to many more short segments in the new 2-D-FPGA and the extended switching capability of the routing block appear to be more than enough to compensate for the reduction in the LB output connectivity. To demonstrate that a similar reduction in LB output connectivity would hurt routability in the baseline 2-D-FPGA, we decreased the LB output connectivity to the connection box in the baseline 2-D-FPGA from 36 to 8. We placed and routed the MCNC benchmark circuits and found

1489

TABLE III
COMPARISON OF PROGRAMMING OVERHEAD PER TILE FOR BASELINE
2-D-FPGA AND THE NEW 2-D-FPGA. S: SWITCHES. T: TRI-STATE
BUFFERS. I: INVERTERS. M: MEMORY BITS

Baseline 2D-FPGA		New 2D-FPGA	
Logic Block	M: 1049	Logic Block	M: 1049
Interconnects	T: 96	Interconnects	T: 288
	M: 96		M: 288
2 Connection Boxes	S: 1440	Routing Block	S: 2880
	M: 1440		T: 432
1 Switch Box	S: 3456		M: 3312
	T: 864		
	I: 1728		
	M: 2592		
Total	S: 4896	Total	S: 2880
	T: 960		T: 720
	I: 1728		M: 4649
	M: 5177		

that this reduction in LB output connectivity results in a 7% increase in the minimum channel width required to successfully route the designs. Furthermore, the average power consumption increased by 9% and delay increased by 13% as a result of this decrease in LB output connectivity.

C. Logic Density

We use the Cadence GSCLib3.0 technology-independent library and Virtuoso to estimate the layout area for the logic block and buffers in the same manner as in [12]. The routing block area is estimated using custom layout. For the baseline 2-D-FPGA, we assume the architecture parameter values in Section II and the interconnect buffer sizes used in [12]. The 2-D-FPGA(1,2) has the same architecture as the baseline except that it has 24 Single and 48 Double segment tracks in each routing channel. For the new 2-D-FPGA, we assume the architecture parameter values in the previous subsection and buffer size 4 for Single interconnects, 6 for Double interconnects, 8 for buffers driving the routing block input, and 6 for shared MUX output buffer. The MUXes and the pass-transistor switches that connect segments to routing blocks use size 4 transistors.

The estimated size of a single tile in the new 2-D-FPGA is $4280\lambda \times 4280\lambda$ compared to $3846\lambda \times 3846\lambda$ for the 2-D-FPGA(1.2) and $4100\lambda \times 4100\lambda$ for the baseline 2-D-FPGA [12]. The slight increase of 9% in the tile area of the new 2-D-FPGA over the baseline 2-D-FPGA is due to the use of larger pass-transistor switch sizes to improve interconnect delay.

D. Interconnect Loading

To quantify the reduction in interconnect loading in the new fabric relative to that in the baseline 2-D-FPGA(1,2), we consider three scenarios, a local connection between two neighboring LBs (see Fig. 15), a connection spanning 3 LBs (see Fig. 16), and a connection spanning 6 LBs with one signal bend (see Fig. 17). Table IV compares the capacitive loading for the three scenarios. To help understand where the reduction in loading comes from, the loading in each case is divided into several components as shown in the figures. From the table, we see that the total loading in the new fabric is 55% lower than that in the baseline fabric for the local connection, 60% lower for
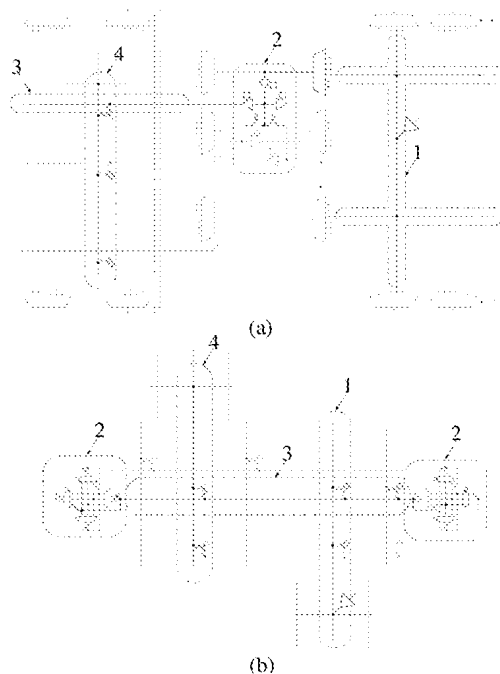

(a)

(b)
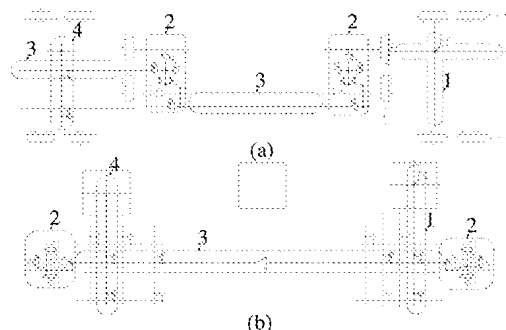
Fig. 15. Local connection in new fabric (a) and in baseline fabric (b). The numbers correspond to (1) output segment, (2) routing point/ switching point, (3) interconnect segment, and (4) input segment.

the length 3 connection, and 39% for the length 6 connection. For the local connection (Fig. 15), the loading savings from the output segment and the interconnect segment are about the same. For the other second scenario (Fig. 16), the largest source of reduction in the loading due to the interconnect segment (component 3). While the Single segment in the baseline fabric with CB connectivity $F_c = 0.5$ has 20 ($= (16 + 4) \times 0.5 \times 2$) pass transistors connected to it from the input and output segments of the two neighboring LBs, the routing block input has only eight pass transistors in addition to six MUX inputs connected to it. The second largest source of loading reduction is the output segment (component 1). The output segment in the baseline with $W = 72$ has 36 pass transistors connected to, while in the new fabric it has only eight MUX inputs connected to it. In addition, there is a small reduction in the loading due to the lower loading of the routing point in the new fabric relative to the switch point in the baseline fabric. For the third scenario (Fig. 17), the largest source of reduction in the loading is the interconnect segment (component 3). However, as shown in Table IV, the parasitic loading due to the signal bend (component 5) in the new routing fabric is significantly larger than that in the baseline fabric. This is because in the new fabric a bend results in high wire loading (see Fig. 8) relative to the baseline. As a result of the high signal bend cost, the loading reduction in the third scenario 3 is only 39%.

E. System Delay

To compare the system delay of the new 2-D-FPGA to that of the baseline 2-D-FPGA, we use two metrics; the improvement in the geometric average of the pin-to-pin net delays, and the improvement in critical path delay, which includes the LB delays along the path. By improvement here we mean the ratio of the

TABLE IV
COMPARISON OF CAPACITIVE LOADING FOR DIRECT CONNECTION AND LENGTH-3 CONNECTION IN 2-D-FPGA(1,2) AND NEW 2-D-FPGA

Component	Direct ($\times 100$ fF)		Length-3 ($\times 100$ fF)		Length-9 ($\times 100$ fF)	
	Baseline	New	Baseline	New	Baseline	New
Output Segment (1)	44.9	23.0	44.9	23.0	44.9	23.0
SP/RP (2)	12.8	5.7	12.8	11.4	51.2	21.0
Interconnect Segment (3)	36.6	19.1	96.9	47.6	193.9	95.1
Input Segment (4)	43.4	41.1	43.4	41.1	43.4	41.1
Signal Bend (5)	-	-	-	-	30.2	80.3
Total	137.7	88.9	198.0	123.1	363.5	260.6

Fig. 16. Length 3 net connection in new fabric (a) and in baseline fabric (b). The numbers correspond to (1) output segment, (2) routing point/ switching point, (3) interconnect segment, and (4) input segment..
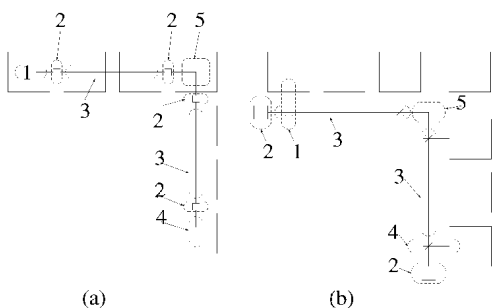
Fig. 17. Length 6 net connection with a bend in new fabric (a) and in baseline fabric (b). The numbers correspond to (1) output segment, (2) routing point/ switching point, (3) interconnect segment, (4) input segment, and (5) signal bend.
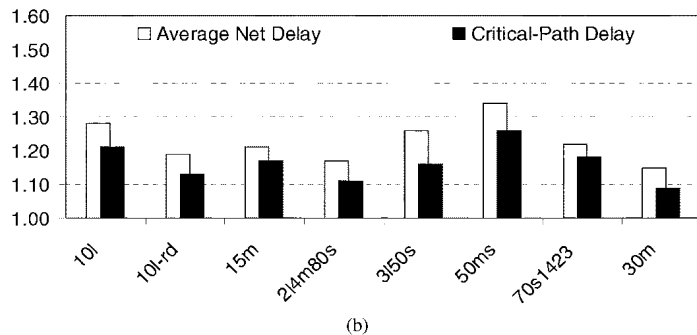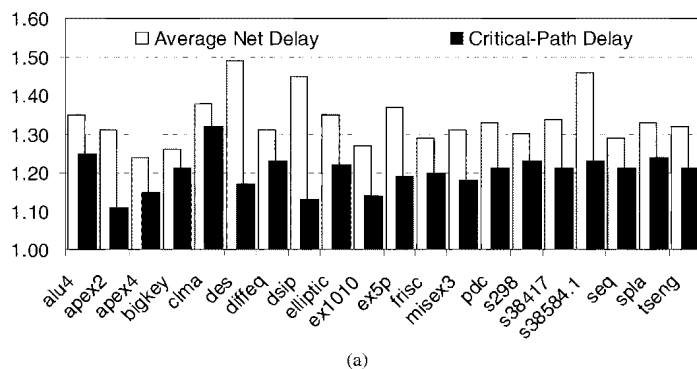

(a)

(b)

Fig. 18. Delay improvements of new 2-D-FPGA over baseline 2-D-FPGA for (a) MCNC benchmark circuits and (b) benchmark designs synthesized with QUIP toolkit from Altera.

delay in the baseline 2-D-FPGA to that in the new 2-D-FPGA. We use two sets of benchmark designs, the 20 largest MCNC benchmark circuits and 8 designs synthesized by QUIP toolkit from Altera.[3] The circuits in the second set are around 4 times larger than the largest MCNC circuit. Results for both benchmark sets are plotted in Fig. 18. Note that for the MCNC benchmarks, the improvements over the baseline 2-D-FPGA range from 1.24 to 1.49 times for the geometric average pin-to-pin delay and from 1.11 to 1.32 times for the critical path delay. On average, pin-to-pin net delay improves by 1.34 times and critical path delay improves by 1.21 times over the baseline 2-D-FPGA. The improvement for the QUIP benchmarks ranges from 1.15 to 1.34 times for the geometric average pin-to-pin delay and from 1.09 to 1.26 times for the critical path delay. On average, pin-to-pin net delay improves by 1.23 times and critical path

[3]Quartus II University Interface Program. 2005.

delay improves by 1.16 times over the baseline 2-D-FPGA. The reason for the lower improvements in the large designs can be attributed to the choice of segmentation. We believe that larger improvements can be achieved by optimizing the segmentation in the new fabric, for example, using TORCH [18].

F. Dynamic Power

Dynamic power consumption consists of three components, the power consumed in the logic blocks P_{LB}, the power consumed in the interconnects P_{int}, and the power consumed in the clock networks P_{clk}. In this paper, we assume that the new 2-D-FPGA uses the same logic block as the baseline 2-D-FPGA. Although potentially the logic block in the new FPGA can have different glitching characteristics that can cause changes in its power consumption, in our study, we assume the component of dynamic power consumed by the logic block in the new 2-D-FPGA is the same as that in the baseline 2-D-FPGA.

We quantify the improvement in the total dynamic power consumption, ξ, between the baseline 2-D-FPGA and the new

2-D-FPGA both implemented in 65-nm technology using the equation (see detailed derivation in [12])

$$\xi = \frac{1}{\left(\phi_{LB} + \frac{\phi_{int}}{\xi_{int}} + \frac{\phi_{clk}}{\xi_{clk}}\right)} \tag{1}$$

where ϕ_{LB}, ϕ_{int}, and ϕ_{clk} are the fraction of dynamic power consumed in the logic blocks, the interconnect, and the clock network of the baseline 2-D-FPGA, respectively ($\phi_{LB} + \phi_{int} + \phi_{clk} = 1$), and $\xi_{int} \geq 1$ is the ratio of the dynamic power consumed by the interconnects in the 2-D-FPGA to that in the new 2-D-FPGA for a particular benchmark circuit in MCNC suite.

We choose $\phi_{LB} = 0.15$, $\phi_{int} = 0.65$, and $\phi_{clk} = 0.2$ to be consistent with recent studies [9], [10]. Because the component of dynamic power consumed in the interconnects is proportional to the total capacitance of all signal nets for a fixed activity factor, we set ξ_{int} equal to the ratio of the total signal net capacitance of the baseline 2-D-FPGA to that in the new 2-D-FPGA for each placed and routed benchmark circuit. We use the same procedure to estimate the dynamic power improvement factor for the clock network ξ_{clk}. We assume the H-tree clock distribution network with distributed buffering [16], [29], [30]. The change of the power consumption in clock network is mainly due to the area change in the new FPGA. We compute the dynamic power improvement ξ for each of the 20 MCNC benchmark circuits assuming a 64 × 64 LB array for both the baseline 2-D-FPGA and the new 2-D-FPGA implemented in 65-nm technology. The results in Fig. 19(a) show a 1.49 to 1.85 times improvement in total dynamic power with an average improvement of 1.56 times. The same procedure is repeated to compute the dynamic power improvement ξ for each of the eight benchmark circuits synthesized with QUIP toolkit assuming a 100 × 100 LB array for both the baseline 2-D-FPGA and the new 2-D-FPGA implemented in 65-nm technology. The results in Fig. 19(b) show a 1.31 to 1.56 times improvement in total dynamic power with an average improvement of 1.47 times. Again, the reason for the reduction in improvement versus the MCNC designs is the choice of segmentation.

G. Performance Comparison Summary

Table V summarizes the results of Sections V-V-C, V-E, and V-F. While the 2-D-FPGA(1,2) has better power consumption and logic density than the baseline 2-D-FPGA, it has worse delay. The new 2-D-FPGA achieves significantly better power consumption than the 2-D-FPGA(1,2), while improving delay over the baseline 2-D-FPGA with only a small penalty in logic density.

VI. NEW 3-D-FPGA VERSUS BASELINE 2-D-FPGA

In this section, we compare the performance of a monolithically stacked 3-D-FPGA using the new fabric to the baseline 2-D-FPGA. First, we discuss how the 3-D-FPGA may be implemented.

A. 3-D Implementation Feasibility

The new 3-D-FPGA can be realized by stacking three active layers on top of a standard CMOS layer with a total of 12 metal
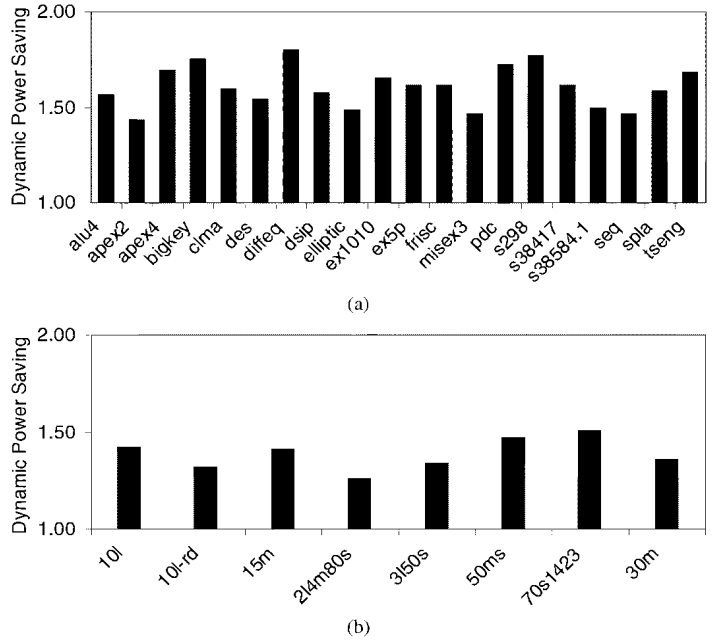

Fig. 19. Dynamic power saving of new 2-D-FPGA over baseline 2-D-FPGA for (a) MCNC benchmark circuits and (b) QUIP toolkit benchmark designs.

TABLE V
PERFORMANCE IMPROVEMENTS

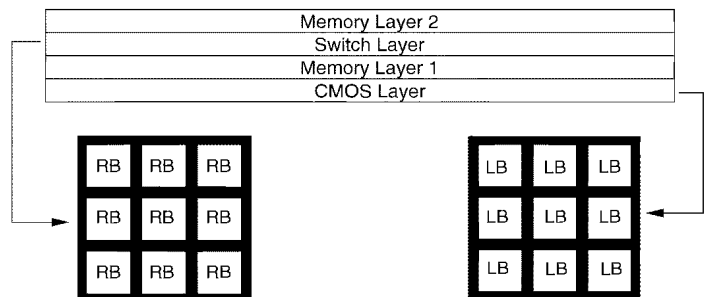|	Logic Density	Pin-to-Pin Delay (Critical Path Delay)	Dynamic Power
Baseline	1.00	1.00 (1.00)	1.00
Baseline (1,2)	1.14	0.91 (0.83)	1.18
New	0.92	1.31 (1.19)	1.54

Fig. 20. Active layers of a 3-D-FPGA using proposed routing fabric. (RB: Routing block, LB: Logic block.

layers interspersed between them (see Fig. 20). The stacked active layers consist of: 1) a first configuration memory layer to program the logic blocks and tri-state buffers, 2) a switch layer comprised only of NMOS devices, and 3) a second configuration memory layer for programming the switches in the switch layer. Because the switch layer has only NMOS device, all buffers are located in the bottom CMOS layer. The use of two memory layers, instead of one as assumed in [12], provides better local vertical connectivity and relaxes the requirement on memory cell size. The 3-D-FPGA is completely tileable, so we focus on the implementation of a single tile consisting of a stack of one logic block and interconnect tri-state buffers, one routing block, and their configuration memory.
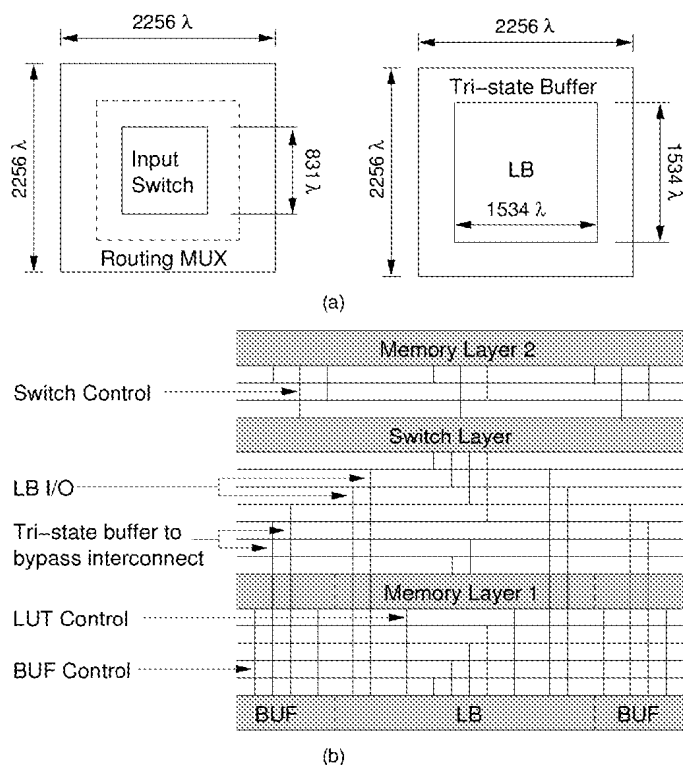
Fig. 21. (a) Tile size and (b) Layer and metal assignment in the new 3-D-FPGA.
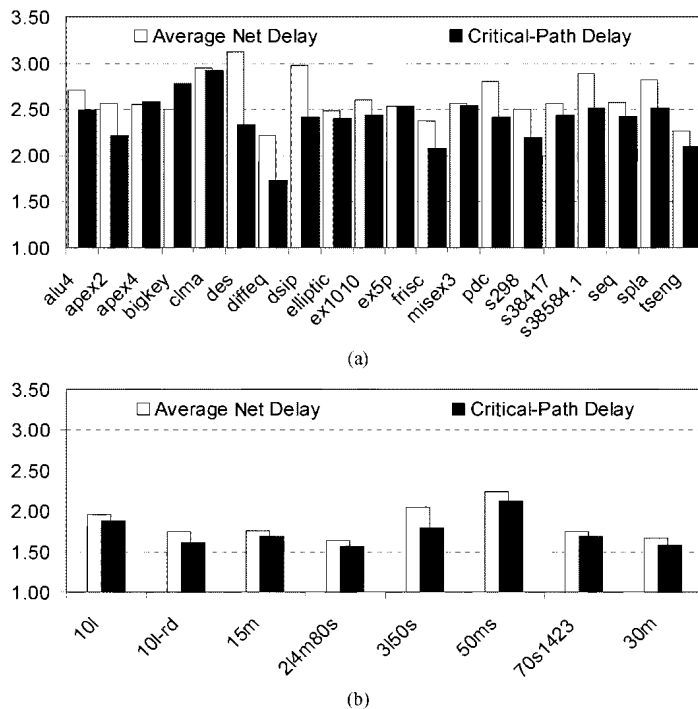

Fig. 22. Delay improvements of new 3-D-FPGA over baseline 2-D-FPGA for MCNC benchmark circuits. (a) Geometric average pin-to-pin delay. (b) Critical path delay.

Fig. 21(b) shows how the metal layers are allocated between the active layers. The bottom layer is a standard CMOS layer with both NMOS and PMOS devices available and is used to implement the logic block and buffers. A minimum of four layers of metal are assumed for signal and power and ground connections. The second layer is the configuration memory layer for the logic block and buffers. Two layers of metal are assumed for this layer. The third layer is the switch layer, where the routing block and interconnect segments are implemented. We assume four layers of metal for this layer. The top layer is the configuration memory layer for the switch layer, which requires two layers of metal. Thus, a minimum of 12 layers of metal are needed to implement this architecture. The vertical lines in Fig. 21(b) depict the vertical connections between different layers. Note that a connection that spans more than two layers is implemented using multiple vias between consecutive active layers. To estimate the tile area of the new 3-D-FPGA, we use the same methodology as in Section V-C. Our estimated footprint size of the new 3-D-FPGA tile is around $2256\lambda \times 2256\lambda$ [Fig. 21(a)]. The logic block and buffer tile and the routing block in the new 3-D-FPGA have similar areas. This is compared to a tile size of $4100\lambda \times 4100\lambda$ for the baseline 2-D-FPGA [12] and corresponds to an improvement in logic density of around 3.3 times, which is slightly better than reported in the previous study [12].

The reason for using two configuration memory layers versus one as in [12] is to relax the area requirement for the configuration memory cell and to provide greater and simpler vertical connectivity.

B. Performance Comparison

As in Section V, we compare the system delay performance using two metrics, the improvement in the geometric average of the pin-to-pin net delays, and the improvement in critical path delay, which includes the LB delays along the path. Results for the largest 20 MCNC benchmark circuits are plotted in Fig. 22. Note that the improvements over the baseline 2-D-FPGA range from 2.31 times to 3.19 times for the geometric average pin-to-pin delay and from 1.82 times and 3.01 times for the critical path delay. On average, there is a 2.59 times delay improvement in pin-to-pin net delay and 2.51 times delay improvement in critical path delay over the baseline 2-D-FPGA. The improvement for the QUIP benchmarks ranges from 1.72 to 2.31 times for the geometric average pin-to-pin delay and from 1.63 to 2.23 times for the critical path delay. On average, pin-to-pin net delay improves by 2.03 times and critical path delay improves by 1.92 times over the baseline 2-D-FPGA.

To obtain the power consumption improvement of new 3-D-FPGA over the baseline 2-D-FPGA, we used the same estimation method as detailed in Section V-F. Again, we computed the dynamic power improvement ξ for each of the 20 MCNC benchmark circuits assuming a 64×64 LB array for both the baseline 2-D-FPGA and the new 3-D-FPGA implemented in 65-nm technology. Our results in Fig. 23 show a 2.59 times to 3.24 times improvement in total dynamic power with an average improvement of 2.91 times. The same procedure is repeated to compute the dynamic power improvement ξ for each of the seven benchmark circuits synthesized with QUIP toolkit assuming a 100×100 LB array for both the baseline 2-D-FPGA and the new 3-D-FPGA implemented in 65-nm technology. The results

LIN AND EL GAMAL: LOW-POWER FPGA ROUTING FABRIC   1493
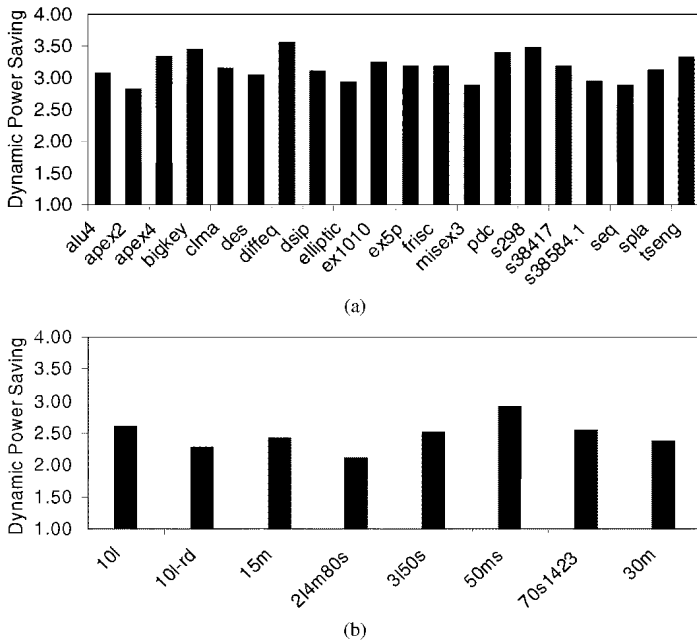

(a)

(b)

Fig. 23. Dynamic power saving of new 3-D-FPGA over baseline 2-D-FPGA for MCNC benchmark circuits.

in Fig. 23(b) show a 2.33 to 3.02 times improvement in total dynamic power with an average improvement of 2.61 times.

VII. CONCLUSION

The power inefficiency of FPGAs relative to cell-based ASICs is a major impediment to their adoption in a broad range of applications and to the continuing improvement in their logic density and performance. In an effort to address this major problem, we proposed a new FPGA routing fabric and showed that an FPGA that uses this fabric can achieve 1.57 times reduction in the overall dynamic power consumption and 1.35 times improvement in average net delays with only 9% reduction in logic density over an island-style baseline 2-D-FPGA implemented in the same 65-nm CMOS technology. The improvements in delay and dynamic power consumption are achieved by using only Single and Double interconnect segments to reduce routed net lengths and by reducing the component of interconnect loading due to programming overhead without adversely affecting routability.

The improvement results are for specific choice of segmentation and transistor and buffer sizes. Further improvements can be achieved by optimizing segmentation using TORCH [18]. This would be particularly important for large designs. Power and logic density can be further improved at the expense of extra delay by using smaller transistor and buffer sizes. To illustrate this, we reduced the MUX transistor sizes from 4 to 3, the pass transistor switch sizes from 4 to 3, and the LB input/output buffer sizes by a factor of 2. This resulted in 11% reduction in silicon area and 7% reduction in power consumption with only 4% increase in delay. An area of improvement in the new fabric is to reduce the high loading incurred by a signal bend. Although the cost of a signal bend can be factored in the placement and routing cost, it would be interesting to investigate architectural ways to reduce it.

The new fabric is well-suited to monolithically stacked 3-D implementation. We showed that a 3-D-FPGA using the new routing fabric can be implemented using a four-layer stack consisting of a CMOS layer, a switch transistor layer, and two memory layers with 12 metal layers between them. This new 3-D-FPGA achieves a 3.3 times improvement in logic density, a 2.51 times improvement in delay, and a 2.93 times improvement in dynamic power consumption over the same baseline 2-D-FPGA. Much work remains to verify the general validity of these performance improvements. However, the fact that the performance improvements presented here are indeed very large warrants continued investigation.

REFERENCES

[1] "Power Consumption in 65 nm FPGAs." White Paper Xilinx, Inc., 2006.
[2] "Stratix III Programmable Power." White Paper Altera, Inc., 2006.
[3] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," in *Proc. 2006 ACM/SIGDA 10th Int. Symp. Field-Programmable Gate Arrays*, 2006, pp. 21–30.
[4] F. Li, Y. Lin, L. He, and J. Cong, "Low-power FPGA using pre-defined dual-Vdd/dual-Vt fabrics," in *Proc. 2004 ACM/SIGDA 12th Int. Symp. Field-Programmable Gate Arrays*, 2004, pp. 42–50.
[5] Y. Lin, F. Li, and L. He, "Circuits and architectures for field programmable gate array with configurable supply voltage," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 13, no. 9, pp. 1035–1047, Sep. 2005.
[6] J. H. Anderson and F. N. Najm, "Low-power programmable routing circuitry for FPGAs," in *Proc. IEEE Int. Conf. Comput.-Aided Design*, 2004, pp. 602–609.
[7] F. Li, D. Chen, L. He, and J. Cong, "Architecture analysis and automation: Architecture evaluation for power-efficient FPGAs," in *Proc. 2003 ACM/SIGDA 10th Int. Symp. Field-Programmable Gate Arrays*, 2003, pp. 175–184.
[8] T. Tuan, S. Kao, A. Rahman, S. Das, and S. Trimberger, "A 90 nm low-power FPGA for battery-powered applications," in *Proc. 2006 ACM/SIGDA 10th Int. Symp. Field-Programmable Gate Arrays*, 2006, pp. 3–11.
[9] L. Shang, A. S. Kaviani, and K. Bathala, "Dynamic power consumption in Virtex-II FPGA family," in *Proc. 2002 ACM/SIGDA 10th Int. Symp. Field-Programmable Gate Arrays*, 2002, pp. 157–164.
[10] V. George, "Low energy field-programmable gate array," Ph.D. dissertation, Univ. of California, Berkeley, 2000.
[11] M. Lin and A. El Gamal, "A routing fabric for monolithically stacked 3-D-FPGAs," in *Proc. 2007 ACM/SIGDA 10th Int. Symp. Field-Programmable Gate Arrays*, 2007, pp. 1–10.
[12] M. Lin, A. El Gamal, Y.-C. Lu, and S. Wong, "Performance benefits of monolithically stacked 3-D-FPGA," in *Proc. 2006 ACM/SIGDA 10th Int. Symp. Field-Programmable Gate Arrays*, 2006, pp. 113–122.
[13] A. R. Joshi and K. C. Saraswat, "Nickel induced crystallization of a-Si gate electrode at 500C and MOS capacitor reliability," *IEEE Trans. Electron Devices*, vol. 50, no. 4, pp. 1058–1062, Apr. 2003.
[14] S. M. Jung, J. Jang, W. Cho, J. Moon, K. Kwak, B. Choi, B. Hwang, H. Lim, J. Jeong, J. Kim, and K. Kim, "The revolutionary and truly 3-dimensional 25F2 SRAM cell technology with the smallest S3 (Stacked single-crystal Si) cell, 0.16 μm^2, and SSTFT(Stacked single-crystal thin film transistor) for ultra high density SRAM," in *Tech. Dig. 2004 VLSI Technol. Symp.*, 2004, pp. 228–229.
[15] "Virtex-II Cpmplete Datasheet (All Four Modules)," Xilinx, Inc., 2007.
[16] F. Li, L. He, and J. Cong, "Architecture evaluation for power-efficient FPGAs," in *Proc. 2003 ACM/SIGDA 11th Int. Symp. Field-Programmable Gate Arrays*, 2003, pp. 175–184.
[17] F. Li, Y. Lin, L. He, D. Chen, and J. Cong, "Power modeling and characteristics of field programmable gate arrays," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 24, no. 11, pp. 1712–1724, Nov. 2005.

[18] "TORCH: A tool for segmented routing channel design in FPGAs, "A routing fabric for monolithically stacked 3-D-FPGAs"," in *Proc. 2008 ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays*, 2008, pp. 131–138.

[19] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*.   Norwell, MA: Kluwer, 1990.

[20] V. Betz and J. Rose, "Directional bias and non-uniformity in FPGA global routing architectures," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design. ICCAD-96. Dig. Tech. Papers.*, , Nov. 10–14, 1996, pp. 652–659.

[21] V. Betz and J. Rose, "Effect of the prefabricated routing track distribution on FPGA area-efficiency," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 6, no. 9, pp. 445–456, Sep. 1998.

[22] V. Betz and J. Rose, "FPGA routing architecture: Segmentation and buffering to optimize speed and density," in *Proc. 1999 ACM/SIGDA 7th Int. Symp. Field Programmable Gate Arrays*, 1999, pp. 59–68.

[23] W. Xu, VPR for Virtex [Online]. Available: http://www-unix.ecs.umass.edu/~wxu/jbits/VPR_for_Virtex.htm.

[24] V. Betz, J. Rose, and A. Marquardt, Eds., Architecture and CAD for Deep-Submicron FPGAs   Norwell, MA, Kluwer, 1999.

[25] G. Lemieux and D. Lewis, "Circuit design of routing switches," in *Proc. ACM/SIGDA 10th Int. Symp. Field-Programmable Gate Arrays*, 2002, pp. 19–28.

[26] S. Yang, "Logic Synthesis and Optimization Benchmarks," Tech. Rep. Microelectron. Center of North Carolina, 1991, 3.0.

[27] V. Betz and J. Rose, "VPR: A new packing, placement and routing tool for FPGA research," in *Proc. 7th Int. Workshop Field-Programmable Logic Applicat.*, 1997, pp. 213–222.

[28] C. Ebeling, L. McMurchie, S. Hauck, and S. Burns, "Placement and routing tools for the Triptych FPGA," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 3, no. 4, pp. 473–482, Apr. 1995.

[29] D. Liu and C. Svensson, "Power consumption estimation in CMOS VLSI chips," *IEEE J. Solid-State Circuits*, vol. 29, no. 6, pp. 663–670, Jun. 1994.

[30] E. Friedman, "Clock distribution design in VLSI circuits—An overview," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 1993, pp. 1475–1478.

Mingjie Lin (S'01–M'08) received the B.S. degree in engineering from Xi'an Jiaotong University, Xi'an, China, in 1993, the M.S. degree in mechanical engineering and electrical engineering from Clemson University, Clemson, SC, in 2001, and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, in 2008.

His current research interests include next-generation FPGA architectures and reconfigurable parallel many-core computer architecture.

Abbas El Gamal (S'71–M'73–SM'83–F'00) received the B.Sc. degree in electrical engineering from Cairo University, Cairo, Egypt, in 1972, and the M.S. degree in statistics and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, in 1977 and 1978, respectively.

From 1978 to 1980, he was an Assistant Professor of electrical engineering with the University of Southern California, Los Angeles. Since 1981, he has been with Stanford University, where he is currently a Professor of electrical engineering and the Director of the Information Systems Laboratory. From 1984 to 1986, he served as the Director of the LSI Logic Research Laboratory. In 1986, he cofounded Actel Corporation and served as its Chief Scientist. In 1990, he cofounded Silicon Architects and served as its Chief Technical Officer until it was acquired by Synopsys in 1995. From 1997 to 2003, he was a Principal Investigator on the Stanford Programmable Digital Camera project. His research contributions have spanned several areas in information theory, digital imaging, and integrated circuit design and design automation. He has authored and coauthored more than 180 papers and holds 30 patents in these areas. He has served on the board of directors and advisory boards of several silicon valley companies.

Multilayer VLSI Layout for Interconnection Networks

Chi-Hsiang Yeh

Dept. of Electrical & Computer Engineering
Queen's University
Kingston, Ontario, K7L 3N6, Canada
chi-hsiang.yeh@ece.queensu.ca

Emmanouel A. Varvarigos and Behrooz Parhami

Dept. of Electrical and Computer Engineering
University of California,
Santa Barbara, CA 93106-9560, USA
{manos, parhami}@ece.ucsb.edu

Abstract

Current VLSI technology allows more than two wiring layers and the number is expected to rise in future. In this paper, we show that, by designing VLSI layouts directly for an L-layer model, the layout area for a variety of networks can be reduced by a factor of about $(L/2)^2$ compared to the layout area required under a 2-layer model, and the volume and maximum wire length can be reduced by a factor of about $L/2$, leading to considerably lower cost and/or higher performance. The proposed layouts for k-ary n-cubes, hypercubes, butterfly networks, cube-connected cycles (CCC), folded hypercubes, generalized hypercubes, k-ary n-cube cluster-c, hierarchical hypercube networks, reduced hypercubes, hierarchical swap networks, and indirect swap networks, are the best layouts reported for these networks thus far and are optimal within a small constant factor under both the Thompson model and the multilayer grid model. All of our layouts are optimally scalable in that we can allow each network node to occupy the largest possible area (e.g., $o(N/L^2)$ for hypercubes) without increasing the leading constant of the layout area, volume, or maximum wire length.

1. Introduction

Twenty years ago many researchers believed that parallel processing would move to the mainstream of computation due to rapid advance in VLSI technologies. A variety of famous papers, theses, and books considered the VLSI layout of interconnection networks for parallel processing [7, 17, 19, 20, 22, 23, 24, 25, 27]. However, the revolution did not materialize at that time; rather, the increased VLSI density was used to build more complex single processors whose performance has improved by two orders of magnitude since then. As recently pointed out by Dally and Lacy [9], the number of transistors per chip will likely increase by another three orders of magnitude in the next two decades and few efficient alternatives to explicit parallelism exist for exploiting the increased number of transistors and grid points. Therefore, the expected revolution may begin soon and the mainstream computing community may shift from serial computers to parallel and distributed systems. The layout of interconnection networks has important cost

and performance implications for single-chip multiprocessors and parallel/distributed systems based on such components. Thus, there is currently renewed interest in finding efficient VLSI layouts for various interconnection networks [3, 8, 10, 12, 13, 21, 28, 30, 31, 32, 35].

VLSI layout of interconnection networks is usually derived under the Thompson model, where two layers of wires are assumed. However, the assumption of two wiring layers cease to be realistic as more and more layers of wires become available in VLSI chips at reasonable cost. When the numbers of wiring layers and active layers (for network nodes) are both increased by a factor of t, the area of a layout designed for the Thompson model can be reduced by a factor of about t by folding the layout, while the volume and maximum wire length remain approximately the same. In this paper, we introduce the *multilayer 2-D grid* and *multilayer 3-D grid models* for VLSI layout of networks. We show that, for a wide variety of networks, including k-ary n-cubes, hypercubes, butterfly networks, cube-connected cycles (CCC) [22, 18], folded hypercubes [1], generalized hypercubes [5, 14], k-ary n-cube cluster-c [4], hierarchical hypercube networks (HHNs) [36], reduced hypercubes [37], hierarchical swap networks (HSNs) [33, 34], indirect swap networks (ISNs) [35], designing layouts under the multilayer 2-D grid model leads to the following advantages:

(1) the area of the layout can be reduced by a factor of approximately t^2 when we use $L = 2t$ layers of wires instead of two layers of wires as in the Thompson model

(2) the volume of the layout can be reduced by a factor of approximately t

(3) the maximum length of wires can be reduced by a factor of approximately t

(4) the maximum total length of wires along the routing path between any source-destination pair can be reduced by a factor of approximately t

For many other networks, including star graphs [2], transposition networks [16, 18], pancake graphs [2], bubble-sort graphs [2], and star-connected cycles (SCC) [15], the preceding arguments are still true, leading to lower cost and/or higher performance for most of the architectures considered thus far for parallel computation. The proposed layouts for butterfly networks, generalized hypercubes, HSNs,

and ISNs are optimal within a factor of $1 + o(1)$ under the Thompson model, and are optimal within a factor of $2 + o(1)$ from a trivial lower bound under the multilayer grid model. These layouts and the proposed layouts for hypercubes, CCCs, folded hypercubes, reduced hypercubes, HHNs, and enhanced cubes constitute the best results reported in the literature for these networks, under both the Thompson model and the multilayer grid model.

The organization of the remainder of the paper is as follows. In Section 2, we discuss existing VLSI layout models, introduce the multilayer grid models that we propose, and propose several layout schemes. In Section 3 we present efficient multilayer layout for k-ary n-cubes, product networks, and related networks. In Section 4 we present efficient multilayer layouts for butterfly networks, generalized hypercubes, and related networks. In Section 5 we present efficient multilayer layouts for hypercubes, CCC, folded hypercube, and related networks. In Section 6 we present our conclusions.

2. VLSI layout models and layout schemes

In this section, we describe several models for VLSI layout of interconnection networks.

2.1. The Thompson model

In the Thompson model [23], a network is viewed as a graph whose nodes correspond to processing elements and edges correspond to wires. The graph is then embedded in a 2-D grid, where wires have unit width and a node of degree d occupies a square of side d. The wires can run either horizontally or vertically along grid lines. Two wires can cross each other at a grid point, but cannot overlap or bend at the same grid point, which would form a knock-knee [6].

The area of a layout is defined as the area of the smallest rectangle that contains all the nodes and wires. (In this paper we only consider upright rectangles for this purpose.) When there are two layers of wires and a node can be laid out in a square of area d^2, it is guaranteed that we can lay out the network within the area of that rectangle. More precisely, we can use one layer of wires to lay out all the horizontal segments of wires and the other layer to lay out all the vertical segments. When a wire makes a turn, its horizontal and vertical parts in different layers are connected by an inter-layer connector known as a via.

Note that some authors have assumed that a node occupies a square of side 1 in the layout model they use. Some such layouts cannot be extended to the Thompson model without a nonnegligible increase in area, while layouts under the Thompson model can usually be extended to the former model using comparable area.

2.2. The multilayer grid model

In the multilayer grid model, a network is viewed as a graph whose nodes correspond to processing elements and edges correspond to wires. The nodes and edges of the graph are then embedded in a 3-D grid, where edges have unit width, can run along grid lines, but cannot cross or overlap with each other (i.e., the paths for embedding these edges must be edge- and node-disjoint). The area A of a layout is defined as the area of the smallest upright rectangle along the x-y directions that contains all the nodes and wires. The

volume of a layout is equal to the number L of layers times its area A.

In the multilayer 2-D grid model, the nodes of the graph are embedded in the 2-D grid of the first layer (i.e., $z = 1$). The range of actual node sizes must be specified explicitly in this model, and is usually taken to be between the minimum size required to implement a node (e.g., a square of side d, $d/4$, or $\frac{d}{4L}$ for a degree-d node in some technologies) and the maximum allowable size without affecting the leading constants for area, volume, and maximum wire length. A network with area A under the Thompson model can be laid out with area no larger than A under the multilayer 2-D grid model with $L = 2$ layers, so the former can be viewed as a special case of the latter. Note, however, that we may derive layouts under the two-layer 2-D grid model with area smaller than the Thompson model. In the multilayer 3-D grid model, the nodes of the graph are embedded in L_A layers of the 3-D grid. These L_A layers are called "active layers" and do not need to be consecutive layers. The range of actual node sizes is also required to be specified explicitly, which is usually between the minimum size required to implement a node (e.g., a cuboid with sides at least $d/h \times d/h \times h$, $1 \leq h \leq L_A \leq L$, for a degree-d node in some technologies) and the maximum allowable size without affecting the leading constants for area, volume, and maximum wire length. The multilayer 2-D grid model is a special case of the multilayer 3-D grid model with $L_A = 1$ active layer. Note that a $d/h \times d/h \times h$ cuboid node requires h active layers for its implementation, while a $d \times d \times 1$ cuboid node requires only 1 active layer. The cost of a layout under the multilayer grid model is a function of A, L, and L_A, as well as other parameters.

The motivations for using multilayer layout models include the significant reduction achieved in the layout area, volume, and maximum wire length required, leading to considerable improvements in both hardware cost and performance. When we use L layers, the number of tracks in the x and y directions may both be reduced by a factor of about $L/2$ in many networks, leading to a factor of about $L^2/4$ reduction in its area compared to the layout under the Thompson model, and a factor of about $L/2$ reduction in its volume (since the number of layers is only increased by a factor of $L/2$). Hence, the cost of the resultant layout can be significantly reduced, or the performance can be significantly improved with the same hardware cost. As a point of comparison, if we fold a layout derived for the Thompson model in order to use all the available layers, the area can be reduced by a factor of only $L/2$, while the volume is unaffected; if we extend the collinear layout model to its multilayer counterpart, the volume will not change either since the area can only be reduced by a factor of at most $L/2$ when L layers are used. The maximum wire length in many networks is approximately proportional to the number of tracks in the x or y direction (or to their sum). Therefore, if the numbers of tracks in the x and y directions are both reduced by a factor of about $L/2$, the maximum wire length can also be reduced by a factor of approximately $L/2$, leading to significant improvement in performance. As a point of comparison, the maximum wire length in a collinear layout using L layers, or in a layout obtained by folding the layout derived using the Thompson model, is not significantly affected in most cases. These arguments will become clear after examining

the multilayer layouts derived in the following subsections.

We can extend the multilayer grid model to the *multilayer layout model* by allowing nodes and edges to run in other specified directions. Layouts under this model may have smaller area and volume compared with layouts under its multilayer grid model counterpart. Moreover, wires in this model may have different width and cross area, depending on the technology used. For example, wires along the z direction may have larger cross area in PCB. In the remainder of the paper, we focus on the multilayer 2-D grid model. When the number L of layers is equal to 2, the multilayer layouts presented in this paper become layouts under the Thompson model. Note that, in general, a multilayer layout with $L = 2$ is not necessarily a layout under the grid model. Layouts under other models, such as the multilayer 3-D grid model and other multilayer layout models, will be reported in the near future.

2.3. The recursive grid layout scheme

In [28, 32], we have proposed the *recursive grid layout scheme* for simple and efficient 2-D layout of interconnection networks. In this subsection, we extend the scheme to the 3-D layout model and briefly present this generally applicable layout scheme.

To lay out an l-level hierarchical network, we first place nodes belonging to the same level-l cluster within a block, which we call a *level-l block*. We arrange the blocks as a 2-D grid for the 2-D layout model or as a 3-D grid for the 3-D layout model, where neighboring rows (or columns) are separated by a sufficient number of horizontal tracks (or vertical tracks, respectively) (see Fig. 1). We then lay out level-l inter-cluster links (i.e., links connecting nodes in different level-l clusters) outside the blocks. Note that we will eventually connect each of the level-l inter-cluster links incident to a level-l block to a certain node within the block. We can then continue to lay out each level-l cluster, including the M_{l-1} level-$(l-1)$ blocks within it and the links connecting these level-$(l-1)$ blocks, within a level-l block. This process is repeated recursively until each block contains a node or until the number of nodes within a block to be laid out is small. Then we use any viable method to lay out all these small clusters.

2.4. The orthogonal multilayer layout scheme

In this subsection, we propose a special case of the recursive grid layout scheme for multilayer layout of general interconnection networks.

In the *orthogonal multilayer layout scheme* we first partition network nodes into clusters and then arrange these clusters as a 2-D grid for the multilayer 2-D layout model or as a 3-D grid for the multilayer 3-D layout model. The partition and arrangement should be carefully performed so that (most of the) inter-cluster links only connect clusters belonging to the same row or column. Note that we in general prefer to make the clusters small if possible in order to reduce the additional area required to lay out these clusters, and a cluster may consist of a single node. We then lay out the inter-cluster links assuming two layers of wires (e.g., under the Thompson model, with one layer for horizontal tracks and the other for vertical tracks) so that the layout area, volume, and/or other cost/performance criteria (such
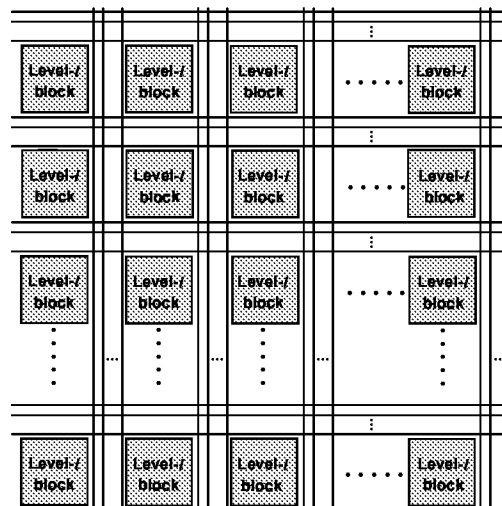
Figure 1. Top-view of a layout based on the recursive grid layout scheme. Level-l blocks are arranged as a 2-D grid.

as maximum wire length) are optimized. We refer to a 2-D layout as an *orthogonal layout* if all of its inter-cluster links connect clusters belonging to the same row or column. As will be shown in what follows, we can always transform an orthogonal layout to an efficient multilayer layout.

Assume that the number of horizontal tracks required above row i of clusters is h_i, the number of vertical tracks required to the right of column j of clusters is w_j, and there are L layers of wires available. We partition the h_i horizontal tracks into $\lceil L/2 \rceil$ groups, each having at most $\lceil h_i / \lceil L/2 \rceil \rceil$ tracks, and the w_j vertical tracks into $\lfloor L/2 \rfloor$ groups, each having at most $\lceil w_i / \lfloor L/2 \rfloor \rceil$ tracks. To obtain an L-layer layout, we assign each group to a certain layer. For example, we can assign the groups for horizontal tracks to layers 1, 3, 5, ..., $2\lceil L/2 \rceil - 1$, and the groups for vertical tracks to layers 2, 4, 6, ..., $2\lfloor L/2 \rfloor$. If the cluster is small enough, which is the typical case for the layout of most interconnection networks, the layout area/volume is dominated by these inter-cluster links. The multilayer layout has then been mostly derived since it is easy to lay out small clusters without increasing the leading constant of layout area/volume. Let A be the layout area using two layers of wires. We can see that when L layers of wires are available, the area of the multilayer layout can be reduced by a factor of about $L^2/4$ and the volume can be reduced by a factor of about $L/2$.

If the cluster is very large and the number L of layers is not small, then we may have to lay out these intra-cluster links carefully. A possible method is to lay out these intra-cluster links recursively using the above method. The details are omitted in this paper.

3. Multilayer layout for k-ary n-cubes, product networks, and PN clusters

In this section, we present multilayer layout of k-ary n-cubes as an example to illustrate the multilayer grid model and the associated orthogonal multilayer layout scheme. We then extend the layout method to arbitrary product networks (also called Cartesian product graphs), k-ary n-cube cluster-c, and product network clusters (PN clusters).

3.1. Multilayer layout for k-ary n-cubes

To apply the orthogonal multilayer layout scheme to a k-ary n-cube, we first place node $(i_{n-1}, i_{n-2}, ..., i_0)$ at position (i, j) of a 2-D grid (i.e., each node is viewed as a cluster in the scheme), where

$$i = i_{n-1}k^{\lceil n/2 \rceil - 1} + i_{n-2}k^{\lceil n/2 \rceil - 2} + \cdots + i_{\lfloor n/2 \rfloor + 1}k + i_{\lfloor n/2 \rfloor},$$

$$j = i_{\lfloor n/2 \rfloor - 1}k^{\lfloor n/2 \rfloor - 1} + i_{\lfloor n/2 \rfloor - 2}k^{\lfloor n/2 \rfloor - 2} + \cdots + i_1 k + i_0.$$

Then all links connect nodes belonging to the same row or column. It can be seen that each row is now connected as a k-ary $\lceil n/2 \rceil$-cube, and each column is connected as a k-ary $\lfloor n/2 \rfloor$-cube, so the 2-D layout problem is reduced to finding collinear layout of k-ary n-cubes, where a collinear layout is a layout derived by first placing all nodes along a line.

To describe the 2-layer collinear layout for a k-ary n-cube, we use a bottom-up approach, starting with a k-node ring (i.e., a k-ary 1-cube), and inductively moving to k-ary n-cubes of higher dimensions n. A collinear layout of a ring can be obtained by placing the k nodes along a row, connecting neighboring nodes through wires in the first track, and then connecting node 0 with node $k - 1$, through a wire in the second track. Clearly, this layout requires 2 tracks. Assume that we have a collinear layout for a k-ary n-cube that requires $f_k(n)$ tracks. To obtain the collinear layout of a k-ary $(n + 1)$-cube, we start with k copies of the layout of a k-ary n-cube. By increasing the horizontal space by a factor of k, we can place the ith node of the jth copy adjacent (from the right) to the ith node of the $(j - 1)^{th}$ copy, $i, j = 0, 1, ..., k - 1$. We also increase the number of tracks (i.e., vertical space) to accommodate the $kf_k(n)$ tracks of the k collinear layout copies. Moreover, to connect the k copies of the k-ary n-cube into a k-ary $(n + 1)$-cube, we need two extra tracks, one containing links between adjacent nodes (i.e., the ith nodes of the k copies) and the other containing a wire connecting the ending nodes of the ring (i.e., the ith nodes of the 0th and $(k - 1)^{th}$ copies). Figure 2 illustrates a resultant collinear layout for a 3-ary 2-cube. Therefore, the number of tracks required for the collinear layout of the k-ary $(n + 1)$-cube is $f_k(n + 1) = kf_k(n) + 2$. Since $f_k(1) = 2$, we have

$$f_k(n) = kf_k(n-1) + 2k^0 = k^2 f_k(n-2) + 2k^1 + 2k^0 = \cdots$$

$$= 2(k^{n-1} + k^{n-2} + \cdots + k^1 + k^0) = \frac{2(k^n - 1)}{k - 1} = \frac{2(N - 1)}{k - 1}.$$

If we connect nodes belonging to the same row (or column) as a 2-layer collinear layout of a k-ary n_1-cube (or k-ary n_2-cube, respectively), we obtain a 2-layer 2-D layout of a k-ary $(n_1 + n_2)$-cube.
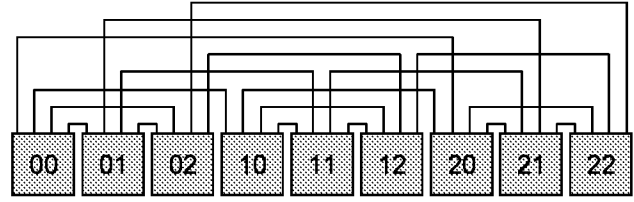

Figure 2. Collinear layout for a 3-ary 2-cube.

We then use the approach described for the orthogonal multilayer layout scheme to transform the 2-layer orthogonal layout to obtain an L-layer layout. When L is even, the number of tracks per layer above a row is $\lceil \frac{4(k^{\lfloor n/2 \rfloor} - 1)}{L(k-1)} \rceil$ and there are $k^{\lceil n/2 \rceil}$ rows; the number of tracks per layer to the right of a column is $\lceil \frac{4(k^{\lceil n/2 \rceil} - 1)}{L(k-1)} \rceil$ and there are $k^{\lfloor n/2 \rfloor}$ columns. Therefore, the area of the L-layer k-ary n-cube layout becomes

$$\frac{16N^2}{L^2 k^2} + o\left(\frac{N^2}{L^2 k^2}\right),$$

and the volume becomes

$$\frac{16N^2}{Lk^2} + o\left(\frac{N^2}{Lk^2}\right),$$

assuming that k is not a constant. To reduce the maximum wire length, we fold each row and column and the resultant maximum wire length becomes

$$O(\frac{N}{Lk^2}).$$

The area for an L-layer k-ary n-cube layout with odd L is

$$\frac{16N^2}{(L^2 - 1)k^2} + o\left(\frac{N^2}{L^2 k^2}\right)$$

and the volume is

$$\frac{16N^2 L}{(L^2 - 1)k^2} + o\left(\frac{N^2}{Lk^2}\right).$$

3.2. Multilayer layout for product networks, PN clusters, and k-ary n-cube cluster-c

The preceding multilayer layout method can be easily extended to general meshes and tori, and can also be further generalized to all product networks [11]. More precisely, for a product network $G = A \times B$, we can use the collinear layouts for the factor graphs A and B to lay out G. To do so, we simply arrange network nodes as a 2-D grid, and connect nodes belonging to the same row as a collinear layout of the factor graph A and nodes belonging to the same column as a collinear layout of the factor graph B. We then obtain a 2-layer orthogonal layout of the product network G, which can then be transformed to an L-layer layout using

the techniques described for the orthogonal multilayer layout scheme. Clearly, this layout method is applicable to binary hypercubes and generalized hypercubes, special cases of product networks, as will be demonstrated in the following sections.

A network obtained by replacing each node of a product network with a cluster is referred to as a *product network cluster (PN cluster)*. In other words, the quotient graph obtained by shrinking each cluster of a PN cluster into a supernode will become a product network. In what follows we further extend the layout method to PN clusters. We can lay out such networks by first deriving an L-layer layout for the quotient graph, and then using the recursive grid layout method (Subsection 2.3) to lay out the clusters. More precisely, we expand each node (which corresponds to a supernode of the PN cluster) in the layout of the quotient graph into a rectangular block and arrange these blocks as a 2-D grid, where neighboring rows (or columns) are separated by a sufficient number of horizontal (or vertical, resp.) tracks (see Fig. 1). We then lay out the cluster within each of the blocks, and connect incident inter-cluster links from outside a block to network nodes within the block in the way specified by the topology. If the area increase due to the expansion of nodes in the quotient graph into rectangles (to lay out the clusters) does not dominate the area of the resultant layout, then the area of the PN cluster remains asymptotically the same as that of the quotient PN layout. Since the clusters and the nodes within the clusters are arranged as 2-D grids, a network node can occupy $o(\frac{\text{Layout Area}}{N})$ area without increasing the leading constants of the layout area, volume, and maximum wire length. For example, a hypercube node can occupy an area as large as $o(N)$ and a k-ary n-cube node can occupy an area $o(N/k^2)$ when L is a constant, instead of areas $\log_2^2 N$ and $4n^2 = 4\log_k^2 N$, respectively, as assumed in most previous papers. Such layouts (including all the layouts proposed in this paper) are optimally scalable in terms of node size since the leading constant of the layout area must become larger when network nodes are larger (i.e., with area $\Omega(\frac{\text{Layout Area}}{N})$).

Let us now consider a k-ary n-cube cluster-c [4] as an example of PN clusters. Assume that the clusters in the k-ary n-cube cluster-c are c-node hypercubes. Then a block with area $O(c^2/L^2)$ is sufficient to accommodate the c-node cluster and its inter-cluster links (see Section 5 or [31]). Since these blocks are arranged as a $k^{n/2} \times k^{n/2}$ grid, the increase in area is negligible as long as the number c of nodes in a cluster is not very large; that is, $c = o(k^{n/2-1})$ so that $\frac{k^{n/2}c}{L} = o(\frac{k^n}{L(k-1)})$ (or $o(\frac{k^{n-1}}{L})$ when k is not a constant), which is the case except when c is large and/or n is small. Clearly, this conclusion applies to any k-ary n-cube cluster-c whose cluster is at most as dense as hyercubes. Similarly, we can show that even if the clusters are complete graphs, a k-ary n-cube cluster-c still has asymptotically the same area (witin a factor of $1 + o(1)$) as a k-ary n-cube as long as $c = o(k^{n/4-1})$.

4. Multilayer layout for generalized hypercubes, butterflies, and related networks

In this section we present efficient multilayer layouts for generalized hypercubes, butterfly networks, HSNs, HHNs, and ISNs.
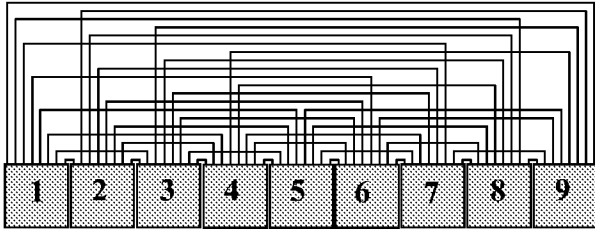
4.1. Multilayer layout for generalized hypercubes

A generalized hypercube [5] is the Cartesian product of two smaller generalized hypercubes. Therefore, we can use the layout method for product networks introduced in Subsection 3.2 to lay out generalized hypercubes.

To describe the 2-layer collinear layout for an n-dimensional radix-$(r_{n-1}, r_{n-2}, ..., r_1, r_0)$ generalized hypercube, we use a bottom-up approach similar to that for laying out k-ary n-cubes. We start with an r_0-node complete graph (i.e., a 1-dimensional radix-r_0 generalized hypercube), and inductively moving to generalized hypercubes of higher dimensions n. We have proposed a strictly optimal collinear layout for N-node complete graphs that requires $\lfloor N^2/4 \rfloor$ tracks (see Fig. 3 for an example) [30, 35]. Assume that we have a collinear layout for an n-dimensional generalized hypercube that requires $f_r(n)$ tracks. To obtain the collinear layout of an $(n+1)$-dimensional generalized hypercube, we start with r_n copies of the layout for an n-dimensional generalized hypercube. By increasing the horizontal space by a factor of r_n, we can place the ith node of the jth copy adjacent (from the right) to the ith node of the $(j-1)^{th}$ copy, $i, j = 0, 1, ..., r_n - 1$. We also increase the number of tracks (i.e., vertical space) to accommodate the $r_n f_k(n)$ tracks of the r_n collinear layout copies. Moreover, to connect the r_n copies of the n-dimensional generalized hypercube into an $(n+1)$-dimensional generalized hypercube, we need $\lfloor r_n^2/4 \rfloor$ extra tracks to connect r_n adjacent nodes (i.e., the ith nodes of the r_n copies) as a complete graph. Therefore, the number of tracks required for the collinear layout of the $(n+1)$-dimensional generalized hypercube is $f_r(n+1) = r_n f_r(n) + \lfloor r_n^2/4 \rfloor$, where $f_r(1) = \lfloor r_0^2/4 \rfloor$. It is easy to solve the recursive function $f_r(n)$ once the mixed-radix $(r_{n-1}, r_{n-2}, ..., r_1, r_0)$ is knowdni. For a radix-r generalized hypercube (i.e., $r_{n-1} = r_{n-2} = \cdots = r_0 = r$), we have

$$f_r(n) = r f_r(n-1) + r^0 \lfloor r^2/4 \rfloor = r^2 f_r(n-2) + (r^1 + r^0) \lfloor r^2/4 \rfloor$$

$$= \cdots = (r^{n-1} + r^{n-2} + \cdots + r^1 + r^0) \lfloor r^2/4 \rfloor$$

$$= \frac{(r^n - 1) \lfloor r^2/4 \rfloor}{r-1} = \frac{(N-1) \lfloor r^2/4 \rfloor}{r-1}$$

By connecting each row as an m-dimensional radix-$(r_{m-1}, r_{m-2}, ..., r_0)$ generalized hypercube and each column as an $(n-m)$-dimensional radix-$(r_{n-1}, r_{n-2}, ..., r_m)$ generalized hypercube using the preceding collinear layouts, we obtain a 2-layer orthogonal layout for an n-dimensional radix-$(r_{n-1}, r_{n-2}, ..., r_0)$ generalized hypercube. We then use the approach described for the orthogonal multilayer layout scheme to transform the 2-layer orthogonal layout to obtain an L-layer layout. For an n-dimensional radix-r generalized hypercube with an even number L of wiring layers, the number of tracks per layer above a row is $\lceil \frac{2\lfloor r^2/4 \rfloor (r^{\lfloor n/2 \rfloor} - 1)}{L(r-1)} \rceil$ and

Figure 3. Collinear layout for a 9-node complete graph.

there are $r^{\lceil n/2 \rceil}$ rows; the number of tracks per layer to the right of a column is $\lceil \frac{2\lfloor r^2/4 \rfloor (r^{\lceil n/2 \rceil} - 1)}{L(r-1)} \rceil$ and there are $r^{\lfloor n/2 \rfloor}$ columns. Therefore, the area of the L-layer generalized-hypercube layout becomes

$$\frac{r^2 N^2}{4L^2} + o\left(\frac{r^2 N^2}{L^2}\right).$$

The volume of the layout is

$$\frac{r^2 N^2}{4L} + o\left(\frac{r^2 N^2}{L^2}\right).$$

The maximum wire length is

$$\frac{rN}{2L} + o\left(\frac{rN}{L}\right) \; and$$

the maximum total length of wires along a shortest routing path is

$$\frac{rN}{L} + o\left(\frac{rN}{L}\right).$$

When L is odd, the area of the resultant generalized-hypercube layout is

$$\frac{r^2 N^2}{4(L^2 - 1)} + o\left(\frac{r^2 N^2}{L^2}\right),$$

assuming that r is not a constant.

4.2. Multilayer layout for butterfly networks

In [35], we have shown that by appropriately partitioning a butterfly network into clusters, these clusters can be connected as a generalized hypercube with multiple links. It is interesting, therefore, that butterfly network can be viewed as a PN cluster (i.e., a generalized hypercube cluster) and laid out using the approach introduced in Subsection 3.2. More precisely, we can partition an $R \times R$ butterfly network into $r(\log_2 R + 1)$-node clusters so that these clusters are connected as an $\frac{R}{r \log_2 R}$-node generalized hypercube where each pair of neighboring clusters are connected by 4 links, where $N = R \log_2 R$. Since a cluster only contains several

copies of small butterfly networks, the layout area is dominated by inter-cluster links and is 16 times that of the area of an $\frac{R}{r \log_2 R}$-node generalized hypercube. Therefore, when L is even, the area of the resultant L-layer butterfly layout is

$$16 \times \frac{r^2 (\frac{N}{r \log_2 N})^2}{4L^2} + o\left(\frac{r^2 (\frac{N}{r \log N})^2}{L^2}\right)$$

$$= \frac{4N^2}{L^2 \log_2^2 N} + o\left(\frac{N^2}{L^2 \log^2 N}\right),$$

the volume of the L-layer layout is

$$\frac{4N^2}{L \log_2^2 N} + o\left(\frac{N^2}{L \log^2 N}\right),$$

and the maximum wire length is

$$\frac{2N}{L \log_2 N} + o\left(\frac{N}{L \log N}\right).$$

When the number L of wiring layers is odd, an N-node butterfly network can be laid out using area

$$\frac{4N^2}{(L^2 - 1) \log_2^2 N} + o\left(\frac{N^2}{L^2 \log^2 N}\right).$$

More details can be found in [35].

4.3. Multilayer layout for hierarchical swap networks and related networks

An l-level hierarchical swap network (HSN) based on r-node nucleus graphs [33, 34] can be derived by replacing each node of an $(l-1)$-dimensional radix-r generalized hypercube with an r-node nucleus graph. Therefore, we can derive multilayer layout for HSNs using our layout for generalized hypercubes. When L is even, the area of the resultant L-layer layout for an N-node HSN is

$$\frac{r^2 (N/r)^2}{4L^2} + o\left(\frac{r^2 (N/r)^2}{L^2}\right) = \frac{N^2}{4L^2} + o\left(\frac{N^2}{L^2}\right),$$

the volume is

$$\frac{N^2}{4L} + o\left(\frac{N^2}{L}\right),$$

the maximum wire length is

$$\frac{N}{2L} + o\left(\frac{N}{L}\right),$$

and the maximum total length of wires along a shortest routing path is

$$\frac{N}{L} + o\left(\frac{N}{L}\right),$$

assuming that r is not a constant. When L is odd, the area of the resultant HSN layout is

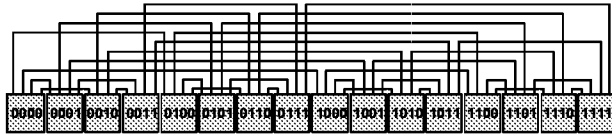$$\frac{N^2}{4(L^2 - 1)} + o\left(\frac{N^2}{L^2}\right).$$

Figure 4. Collinear layout for a 4-cube.

The HSN layout can be easily generalized to general swap networks. Since hierarchical hypercube networks (HHNs) [36] are a special case of HSNs where the basic modules are hypercubes, they can be laid out in asymptotically the same area, volume, and maximum wire length (within a factor of $1 + o(1)$).

Similar to butterfly networks, we can partition an $R \times R$ indirect swap network (ISN) [35] into $r(\log_2 R + o(\log R))$-node clusters so that these clusters are connected as an $(\frac{N}{r \log_2 N} + o(\frac{N}{r \log n}))$-node generalized hypercube with two links connecting each pair of neighboring clusters. Therefore, we can show that the multilayer layout for an ISN has area and volume smaller than those of a similar-size butterfly network by a factor of approximately 4, and the maximum wire length and the maximum total length of wires along a shortest routing path are smaller than those of a similar-size butterfly network by a factor of approximately 2.

We can use similar strategies to obtain efficient multilayer layouts for star graphs and other Cayley graphs [2], such as transposition networks [16, 18], pancake graphs [2], bubble-sort graphs [2], macro stars [29], and star-connected cycles (SCC) [15]. The details will be reported in the near future.

5. Multilayer layout for hypercubes, CCCs, and related networks

In this section, we present multilayer layouts for hypercubes, folded hypercubes, CCC, and reduced hypercubes.

5.1. Multilayer layout for hypercubes

An n-dimensional hypercube is the Cartesian product of two smaller hypercubes of dimensions $\lceil n/2 \rceil$ and $\lfloor n/2 \rfloor$. Therefore, we can use the layout method for product networks introduced in Subsection 3.2 to lay out hypercubes. We have proposed an efficient collinear layout that requires $\lfloor 2N/3 \rfloor$ tracks, derived by a bottom-up approach as that for k-ary n-cubes (Subsection 3.1) and generalized hypercubes (Subsection 4.1). The layout is based on a 2-track collinear layout for 2-cubes (rather than the 1-track collinear layout for 1-cubes) as the basic building block (see Fig. 4) [28, 31]. Therefore, we can show that the area of the resultant L-layer hypercube layout is

$$\frac{16N^2}{9L^2} + o\left(\frac{N^2}{L^2}\right);$$

the volume is

$$\frac{16N^2}{9L^2} + o\left(\frac{N^2}{L^2}\right);$$

and the maximum wire length is

$$\frac{2N}{3L} + o\left(\frac{N}{L}\right).$$

More details can be found in [31].

5.2. Multilayer layout for CCC and reduced hypercubes

An n-dimensional cube-connected cycles (CCC) graph is obtained by replacing each node in an n-cube with an n-node cycle [22]. A reduced hypercube, $RH(\log_2 n, \log_2 n)$ [37], can be obtained by replacing each n-node cycle in a CCC with a $\log_2 n$-dimensional hypercube. Clearly, both of them can be viewed as PN clusters (i.e., hypercube clusters) and laid out using the method presented in Subsection 3.2.

To lay out a CCC, we first lay out an n-cube using the 2-D layout introduced in Subsection 5.1, and then lay out the n-node cycles within each of the hypercube nodes using the recursive grid layout scheme. Since the size of an n-dimensional CCC is $N = n2^n$ and its area is dominated by its hypercube links, which require $\frac{2^{n+4}}{9L^2} + o(2^n/L^2)$ area, an N-node CCC can be laid out in

$$\frac{16N^2}{9L^2 \log_2^2 N} + o\left(\frac{N^2}{L^2 \log^2 N}\right)$$

area when L is even. The layout area is better than that of a recently proposed CCC layout [8]. Using the same layout method, the reduced hypercube can be laid out in asymptotically the same area.

5.3. Multilayer layout for folded hypercubes and related networks

An enhanced-cube is a hypercube with one additional outgoing link per node leading to a random node [26]. A folded hypercube [1] is a hypercube with one additional link per node, where each node S has a link connecting it to the node whose label is the bitwise complement of S.

By adding additional links to our hypercube layout we can lay out folded hypercubes efficiently. More precisely, we first lay out an N-node hypercube in a square of side $\frac{2N}{3L} + o(N/L)$. To lay out an additional link, we need at most one additional vertical track and one additional horizontal track, besides the two ending segments connecting the link to two nodes. Since there are $N/2$ additional links in a folded hypercube, we need at most $N/2$ extra vertical and horizontal tracks to accommodate all the diameter links. These links can be partitioned into $L/2$ groups easily and laid out using L layers. Therefore, the area for the layout of a folded hypercube is

$$\left(\frac{7N}{3L} + o(\frac{N}{L})\right) \times \left(\frac{7N}{3L} + o(\frac{N}{L})\right) = \frac{49N^2}{9L^2} + o(N^2/L^2),$$

when L is even. Since there are N additional links in an enhanced-cube, we need at most N vertical and horizontal tracks to accommodate all the additional links. Therefore, the area for the layout of an enhanced-cube is $\frac{100N^2}{9L^2} + o(N^2/L^2)$. Some of these additional links may be placed in the same tracks so that the layout areas may be reduced.

6. Conclusion

In this paper, we introduced the multilayer grid model and showed that, for a variety of networks, the area can be reduced by a factor of $L^2/4$, and the volume and the maximum wire length can be reduced by a factor of $L/2$, relative to layouts using two layers of wires. The proposed layouts are the best layouts reported for these networks thus far and are optimal within a small constant factor under the multilayer grid model. The techniques introduced in this paper can also be applied to a variety of other networks.

References

[1] Adams, G.B. and H.G. Siegel, "The extra stage cube: a fault-tolerant interconnection network for supersystems," *IEEE Trans. Comput.*, vol. 31, no. 5, May. 1982, pp. 443-454.

[2] Akers, S.B. and B. Krishnamurthy, "A group-theoretic model for symmetric interconnection networks," *IEEE Trans. Comput.*, Vol. 38, Apr. 1989, pp. 555-565.

[3] Avior, A., T. Calamoneri, S. Even, A. Litman, and A. Rosenberg, "A tight layout of the butterfly network," *Theory Comput. Sys.*, vol. 31, no. 4, 1998, pp. 475-488.

[4] Basak, D. and D.K. Panda, "Designing clustered multiprocessor systems under packaging and technological advancements," *IEEE Trans. Parallel Distrib. Sys.*, vol. 7, no. 9, Sep. 1996, pp. 962-978.

[5] Bhuyan, L.N. and D.P. Agrawal, "Generalized hypercube and hyperbus structures for a computer network," *IEEE Trans. Comput.*, vol. 33, no. 4, Apr. 1984, pp. 323-333.

[6] Brady, M.L. and M. Sarrafzadeh, "Stretching a knock-knee layout for multilayer wiring," *IEEE Trans. Computers*, vol. 39, no. 1, Jan. 1990, pp. 148-151.

[7] Brebner, G., "Relating routing graphs and two-dimensional grids," *VLSI: Algorithms and Architectures*, 1985, pp. 221-231.

[8] Chen, G. and F.C.M. Lau, "Tighter layouts of cube-connected cycles," *IEEE Trans. Parallel Distrib. Sys, IEEE Trans. Parallel Distrib. Sys.*, vol. 11, no. 2, Feb. 2000, pp. 182-191.

[9] Dally, W.J. and S. Lacy, "VLSI architecture: past, present, and future," *Proc. Advanced Research in VLSI Conf.*, 1999, pp. 232-241.

[10] Dinitz, Y., S. Even, R. Kupershtok, and M. Zapolotsky, "Some compact layouts of the butterfly," *Proc. ACM Symp. Parallel Algorithms and Architectures*, Jun. 1999, pp. 54-63.

[11] Efe, K. and A. Fernandez, "Products of networks with logarithmic diameter and fixed degree," *IEEE Trans. Parallel Distrib. Sys.*, vol. 6, no. 9, Sep. 1995, pp. 963-975.

[12] Even, S., S. Muthukrishnan, M.S. Paterson, and S. Cenk Sahinalp, "Layout of the Batcher bitonic sorter," *Proc. ACM Symp. Parallel Algorithms and Architectures*, 1998, pp. 172-181.

[13] Fernández, A. and K. Efe, "Efficient VLSI layouts for homogeneous product networks," *IEEE Trans. Computer*, vol. 46, no. 10, Oct. 1997, pp. 1070-1082.

[14] Lakshmivarahan, S. and S.K. Dhall, "A new hierarchy of hypercube interconnection schemes for parallel computers," *J. Supercomputing*, vol. 2, 1988, pp. 81-108.

[15] Latifi, S., M.M. de Azevedo, and N. Bagherzadeh, "The star connected cycles: a fixed-degree network for parallel processing," *Proc. Int'l Conf. Parallel Processing*, Vol. I, 1993, pp. 91-95.

[16] Latifi, S. and P.K. Srimani, "Transposition networks as a class of fault-tolerant robust networks," *IEEE Trans. Parallel Distrib. Sys.*, Vol. 45, no. 2, Feb. 1996, pp. 230-238.

[17] Leighton, F.T., *Complexity Issues in VLSI : Optimal Layouts for the Shuffle-exchange Graph and Other Networks* Cambridge, Mass., MIT Press, 1983.

[18] Leighton, F.T., *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes,* Morgan-Kaufman, San Mateo, CA, 1992.

[19] Leiserson, C.E., *Area-Efficient VLSI Computation*, Cambridge, MA, MIT Press, 1983.

[20] Leiserson, C.E., "Fat-trees: universal networks for hardware-efficient supercomputing," *IEEE Trans. Computers*, vol. C-34, no. 10, Oct. 1985, pp. 892-901.

[21] Muthukrishnan, S., M.S. Paterson, S. Cenk Sahinalp, and T. Suel, "Compact grid layouts of some multi-level networks," *Proc. ACM Symp. Theory of Computing*, 1999, to appear.

[22] Preparata, F.P. and J.E. Vuillemin, "The cube-connected cycles: a versatile network for parallel computation," *Commun. ACM*, vol. 24, No. 5, pp. 300-309, May 1981.

[23] Thompson, C.D., "Area-time complexity for VLSI," *Proc. ACM Symp. Theory of Computing*, 1979, pp. 81-88.

[24] Thompson, C.D., "A complexity theory for VLSI," Ph.D. dissertation, Dept. of Computer Science, Carnegie-Mellon Univ., Pittsburgh, PA, 1980.

[25] Ullman, J.D., *Computational Aspects of VLSI*, Rockville, MD., Computer Science Press, 1984.

[26] Varvarigos, E.A., "Static and dynamic communication in parallel computing," Ph.D. dissertation, Dept. Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1992.

[27] Wise, D.S., "Compact layouts of banyan/FFT networks," *VLSI Systems and Computations*, Computer Science Press, 1981, pp. 186-195.

[28] Yeh, C.-H., "Efficient low-degree interconnection networks for parallel processing: topologies, algorithms, VLSI layouts, and fault tolerance," Ph.D. dissertation, Dept. Electrical & Computer Engineering, Univ. of California, Santa Barbara, Mar. 1998.

[29] Yeh, C.-H. and E.A. Varvarigos, "Macro-star networks: efficient low-degree alternatives to star graphs," *IEEE Trans. Parallel Distrib. Sys.*, Vol. 9, no. 10, Oct. 1998, pp. 987-1003.

[30] Yeh, C.-H. and B. Parhami, "VLSI layouts of complete graphs and star graphs," *Information Processing Letters*, Vol. 68, Oct. 1998, pp. 39-45.

[31] Yeh, C.-H., E.A. Varvarigos, and B. Parhami, "Efficient VLSI layouts of hypercubic networks," *Proc. Symp. Frontiers of Massively Parallel Computation*, Feb. 1999, pp. 98-105.

[32] Yeh, C.-H., B. Parhami, and E.A. Varvarigos, "The recursive grid layout scheme for VLSI layout of hierarchical networks," *Proc. Merged Int'l Parallel Processing Symp. & Symp. Parallel and Distributed Processing*, Apr. 1999, pp. 441-445.

[33] Yeh, C.-H. and B. Parhami, "The index-permutation graph model for hierarchical interconnection networks," *Proc. Int'l Conf. Parallel Processing*, Sep. 1999, pp. 48-55.

[34] Yeh, C.-H. and B. Parhami, "A unified model for hierarchical networks based on an extension of Cayley graphs," *IEEE Trans. Parallel Distrib. Sys.*, to appear.

[35] Yeh, C.-H., B. Parhami, E.A. Varvarigos, and H. Lee, "VLSI layout and packaging of butterfly networks," *Proc. ACM Symp. Parallel Algorithms and Architectures*, 2000, to appear.

[36] Yun S.-K. and K.H. Park, "Hierarchical hypercube networks (HHN) for massively parallel computers," *J. Parallel Distrib. Comput.*, vol. 37, no. 2, Sep. 1996, pp. 194-199.

[37] Ziavras, S.G., "RH: a versatile family of reduced hypercube interconnection networks," *IEEE Trans. Parallel Distrib. Sys.*, vol. 5, no. 11, Nov. 1994, pp. 1210-1220.

Evaluation of MP-SoC Interconnect Architectures: a Case Study

Partha Pratim Pande, Cristian Grecu, Michael Jones, André Ivanov, and Res Saleh

SOC Research Lab
Department of Electrical and Computer Engineering
University of British Columbia
2356 Main Mall Vancouver, BC, V6T 1Z4 Canada
Email: {parthap, grecuc, michaelj, ivanov, res}@ece.ubc.ca

Abstract

Multi-Processor (MP-SoC) platforms are emerging as the latest trend in SoC design. These MP-SoCs consist of a large number of IP blocks in the form of functionally heterogeneous embedded processors. In this new design paradigm, IP blocks need to be integrated using a structured interconnect template, for example, according to high-performance parallel computing architectures. A formal evaluation process is required before adopting a specific parallel architecture to SoC domain. Here, we propose an evaluation methodology based on performance metrics that include latency, throughput and silicon area requirements. As a case study, we present the results of such an evaluation for two MP-SoC interconnect topologies, i.e., the MESH and the Butterfly Fat-Tree (BFT). This evaluation methodology can be extended to any other SoC interconnect topology without loss of generality.

Keywords: MP-SoC, Mesh, BFT, Latency, Throughput.

1. Introduction

According to ITRS [1], the realization of complex Systems on a Chip (SoCs) consisting of billions of transistors fabricated in technologies characterized by 65 nm feature size and less will soon be reality. The emergence of SoC platforms consisting of large, heterogeneous sets of embedded processors is imminent. A key element of such multiprocessor SoC (MP-SoC) platforms [2] is the *interconnect topology*. We specifically propose that the on-chip interconnect topology should resemble the interconnect architecture of high-performance parallel computing systems. The common characteristic of these kinds of architectures is that the functional IP blocks communicate with each other through the help of intelligent switches. As such, the switches can be considered as *infrastructure* IPs (I^2Ps) [3] providing a robust data transfer medium for the functional IP modules.

There are different possible interconnect architectures well documented in the parallel processing domain. However, only a few of them would be suitable for SoC implementation. One of these architectures is the Butterfly Fat Tree (BFT) [4]. Another possible interconnect architecture for SoC design is the MESH as proposed in [5] [6].

In this paper, our main contribution is the establishment of a performance/cost evaluation methodology for MP-SoC interconnects. As a case study this methodology is applied and experimental results are shown for the two different MP-SoC architectures, mentioned above.

2. Related Work

A few on-chip micro-network proposals for SoC integration can be found in the literature. Sonic's Silicon Backplane [7] is one example. Kumar [5] and Dally [6] have proposed mesh-based interconnect architectures. In this case, the number of switches is equal to the number of functional IPs. Guerrier and Greiner [8] proposed the use of a tree- based interconnect (SPIN) and addressed system level design issues.

In [4] and [9], we have described an interconnect architecture for a networked SoC based on a BFT architecture, as well as the associated design of required switches and addressing mechanisms.

The precise focus of this paper is to present and discuss a formal evaluation methodology allowing for objective comparisons of different architectures. Such an analysis can prove invaluable for SoC integrators faced with having to select a suitable MP-SoC interconnect template.

3. Performance Metrics

In a communication-centric design methodology, data is transferred among the IP blocks in the form of packets. The need for storing entire packets in a switch makes the buffer requirement high in the case of conventional packet switching. In wormhole switching, the packets are divided into fixed-length flow control units (flits) and the input and output buffers should be able to store only a few flits. As a result, the buffer space requirement in the switches can be small compared to that generally required for conventional packet switching. One drawback of this simple wormhole switching method is that the transmission of distinct messages cannot be interleaved or multiplexed over a physical channel. This will decrease channel utilization if a flit from a given packet is blocked in a buffer. By introducing virtual channels in the input and output ports, channel utility can be increased considerably [9]. If a flit belonging to a particular packet is blocked in one of the virtual channels, then flits of alternate packets can use the other virtual channel buffers, and hence, ultimately, the physical channel.

It is desirable that a MP-SoC interconnect architecture exhibits low latency and high throughput. The additional area overhead due to the infrastructure IPs should be reasonably small.

3.1 Latency

Latency is defined as the time (in clock cycles) that elapses from between the occurrence of a message header injection into the network at the source node and the occurrence of a tail flit reception at the destination node [10].

In order to reach the destination node, the flits have to travel through a path consisting of a set of *stages*. Depending on the source/destination pair and the routing algorithm, each message may have a different latency. We use the average latency as a performance metric in our evaluation methodology.

Let P be the total number of messages reaching their destination IPs, and let L_i be the latency of each message, where i

varies from 1 to P. The average latency, *Lat,* is then calculated according to the following:

$$Lat = \frac{\sum_{1}^{P} L_i}{P}$$

3.2 Throughput

We define *throughput* [TP] as follows:

$$TP = \frac{(Total\ messages\ completed) \times (Message\ length)}{(Number\ of\ IP\ blocks) \times (Total\ time)}$$

Total messages completed refers to the number of whole messages that successfully arrive at their destination IPs, *Message length* is measured in flits, *Number of IP blocks* is the number of functional IP blocks, and *Total time* is the time (in clock cycles) that elapses between the occurrence of the first message generation and the last message reception. Thus, throughput is measured as the fraction of the maximum load the network is capable of physically handling. A throughput $TP = 1$ corresponds to all end nodes receiving one flit every cycle. Realistically, $TP < 1$ since it is improbable that all possible destinations be active at each cycle. Accordingly, throughput is measured in flits/second.

3.4 Area Requirements

To evaluate the feasibility of these interconnect schemes we need to study their silicon area requirements. As the switches form an integral part of the active components of these infrastructures, it is important to determine the amount of relative silicon area they consume. The switches have two main components: the storage buffer, and logic to implement routing and flow control. The storage buffers are the FIFOs at the inputs and outputs of the switch. Another source of silicon area overhead is the area for the inter-switch wires, which, depending on their lengths may have to be buffered through repeater insertion to keep the inter-switch delay within one clock cycle [11]. Consequently, this additional buffer area has to be accounted for.

4. Evaluation Methodology

The communication-centric parameters that need to be considered when evaluating a MP-SoC interconnect are latency and throughput. The area overhead due to the I^{2}Ps should be considered as well since their sole purpose is to transfer data among functional IP blocks.

We developed a flit-level event-driven wormhole routing simulator to study the characteristics of the communication-centric parameters of the interconnect infrastructure.

To estimate the silicon area consumed by the switches, we developed their VHDL models and synthesized them using a fully static, standard cell-based, CMOS 0.13 μm technology.

5. Interconnect Architectures

In [4], we first proposed a novel interconnect template to integrate numerous IPs of an SoC following a butterfly fat-tree architecture. In our network, the IPs are placed at the leaves and switches are placed at the vertices. Figure 1 illustrates the physical placement of a butterfly fat tree with *64* IPs. In the limit, the number of switches converges to *N/2* for a system size of *N,* as *N* grows arbitrarily large. In the case of 64 IPs the number of switches is 28 as shown in Figure 1.
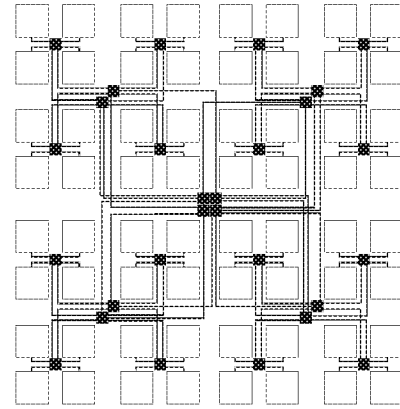

Figure 1: Floorplan of a 64-IP-BFT network.

In a mesh topology, IP blocks are arranged in a two dimensional array structure where each switch (infrastructure IP) is connected to four neighboring switches and a functional IP module. Figure 2 shows the physical placement of 64 functional IP blocks in a mesh based on [5] [6]. In both Figs.1 and 2, the darker blocks denote the switches, and the white squares represent the functional IPs. It is clear that the number of switches in a mesh is equal to the number of interconnected functional IP blocks.
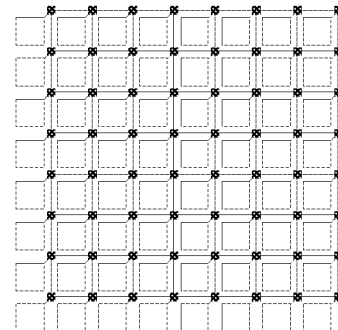

Figure 2: Floorplan of a 64-IP-MESH network.

7. Case Study

We applied our evaluation methodology to two different MP-SoC interconnect platforms, e.g. MESH [6] and BFT [4].

Data were obtained by using a 2500 lines event-driven wormhole routing simulator written in C++.

Uniform, localized, and bit-reversal traffic patterns can be chosen as well. When injections occur, a source IP is selected randomly, according to a uniform distribution. The mean time between packet injections can be specified to alter the overall load on the network. The destination IP selection depends on the traffic pattern adopted. All packets are assumed to have a constant length.

Switch parameters can also be specified. These include input/output port buffer depths (in flits), number of ports, and the number of virtual channels per switch port.

All resource contention is handled without bias such that granting of resources to packets is done on a first come, first-serve basis. The routing algorithms employed are LCA (Least Common Ancestor) determination for the BFT and X-Y routing for the MESH [10].
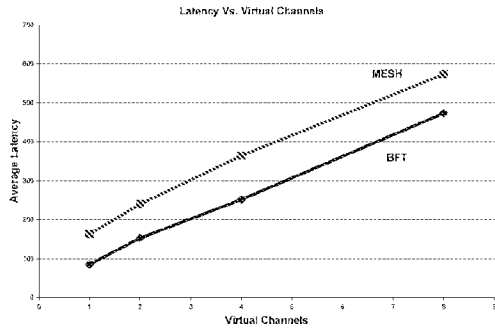
7.1 Communication-centric Parameters

Latency: In Figure 3 we have plotted the average latency, varying the injection load and the number of virtual channels. Figure 3 indicates a better performance of the BFT in terms of latency till the network saturates.
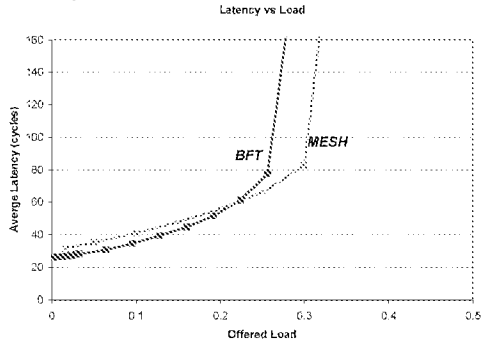

Figure 3 (a): Latency comparison (varying injection load)

**Figure 3 (b): Latency comparison (varying virtual channels)**

Throughput: The throughput of the communication infrastructure generally depends on the traffic pattern. Measuring throughput under random traffic with uniform distribution assumptions is an accepted metric [10] for evaluating parallel systems. Figure 4(a) shows the variation of throughput with the number of virtual channels for both the BFT and MESH topologies, under the stated distribution assumptions.
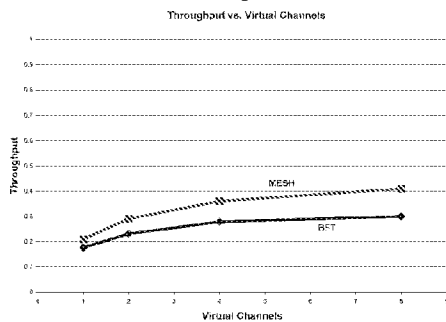
Figure 4(a): Throughput comparison under uniform traffic.

It turns out however, that this uniform traffic assumption is not very realistic in an SoC environment, since different functions will be mapped to different parts of the SoC and they will exhibit highly localized patterns. Hence, we studied the effect of traffic

localization on throughput. We considered a very simple case of localization where local messages travel from source to the set of the nearest destinations. In the case of BFT localized traffic is constrained within a single sub-tree while in the case of MESH it is constrained within the four destinations placed at the shortest Manhattan distance [10]. We define the fraction of localization as the ratio between the local traffic and the total traffic. Fig 4(b) shows the effect of traffic localization on throughput for both topologies.
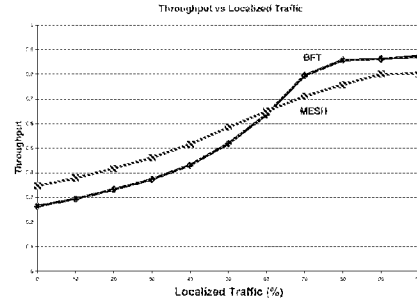

Figure 4(b): Throughput comparison under localized traffic.

From Figure 4(b), we conclude that though the BFT has a lower throughput under uniform traffic, in the presence of localized traffic, its throughput is higher if the fraction of localization is beyond 60%.

7.2 Area Requirements

From our initial implementation, we deduce that within a switch the buffer area significantly dominates over the logic [9]. The buffer area, in turn, largely depends on the number of virtual channels and the flit length. The proposed packets consist of a header flit, one or more data flits, and a tail flit. The header, data, and tail flit structures are as shown in Figure 5.
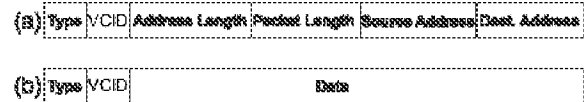
Figure 5: Packet structure (a) Header flit; (b) Data and Tail flit.

The first field denotes the flit type, namely *header, data* or *tail*. The second field contains the virtual channel identifier (VCID). The third field denotes the address length, which is dependent on the number of SoC IP blocks. The fourth field contains packet length information, i.e., the number of flits in the corresponding packet. The next two fields give source and destination addresses.

From Figure 4(a), if the number of virtual channels is increased beyond four, then there is a trend towards throughput saturation. A system with four virtual channels strikes an appropriate balance between high throughput and conservation of silicon area.

The number of IPs in a single SoC varies from one technology node to another, and consequently, the number of bits required to address the IPs will also vary. We consider that each functional IP block consists of 100K gates [11]. In a networked SoC, IPs can be divided into two groups, *functional* and *infrastructure* IPs (switches). Through RTL level design and synthesis, we found that the switches consist of approximately 30K

gates. Consequently, the distribution of functional and I²Ps depends on their respective sizes and interconnect topology. Letting A_{FIP} denote the area of the 100K gates functional IP blocks and A_{I^2P} denote the area of the switches then

$$Area_{chip} = N_1 \cdot A_{FIP} + N_2 \cdot A_{I^2P}$$

where N_1 and N_2 are the number of functional and infrastructure IPs, respectively, and Area$_{chip}$ is the total area of the SoC under consideration. For a BFT, $N_1 = 2N_2$, and for a MESH, $N_1=N_2$. These help us in determining the distribution of functional and infrastructure IP blocks in a SoC.

Functional IPs govern the number of bits required to denote each of address length, source address and destination address fields. We have considered each packet to be consisting of 16 flits [10]. Two bits are needed to specify each of *Flit Type* and *VCID,* and four bits will be sufficient to denote the packet length in each technology node, irrespective of the topologies. The number of bits required to denote the flit type, VCID, and the packet length are topology- and technology-independent. As the number of functional IP blocks varies with interconnect topologies and technology nodes, the size of the address length, source and destination address fields will vary accordingly. Table 1 shows the number of bits in the header flit in the BFT and MESH architectures. The lengths of the body flits (data or tail) are kept equal to the header flit length.

Table 1: Flit lengths in different technology nodes.

Tech. node	Address Length		Source Address		Destination Address		Header Flit Length	
	BFT	MESH	BFT	MESH	BFT	MESH	BFT	MESH
130 nm	4	4	9	9	9	9	30	30
90 nm	4	4	10	10	10	10	32	32
65 nm	4	4	12	11	12	11	36	34
45 nm	4	4	13	13	13	13	38	38
32 nm	4	4	14	13	14	13	40	38

The other contributing factors to the area overhead are the inter-switch repeaters. The wire length between switches in the BFT architecture depends on the levels of the switches. Consequently to keep the inter-switch wire delay within one clock cycle some of them need to be buffered [12]. In a MESH, all the inter-switch wires are of equal length and their delay is always within one clock cycle [12]. Consequently, no repeater insertion is required.

The inter-switch channel width is assumed to be equal to the size of the flit, specified in the last column of Table 1.

The silicon area overhead in different technology nodes can be estimated for both the BFT and MESH interconnect architectures, as the sum of the area due to the switches ($Area_{I^2P}$) and repeaters ($Area_{repeaters}$).

$$Area_{overhead} = Area_{I^2P} + Area_{repeaters}$$

Figure 6 indicates the silicon area overhead across different technology nodes for both platforms.

In terms of silicon requirements, both platforms require a reasonably low area (from 9% in 130 nm node to 14.2% in 32 nm node); however the BFT interconnect architecture offers a more efficient alternative than MESH (20% less area on the average).
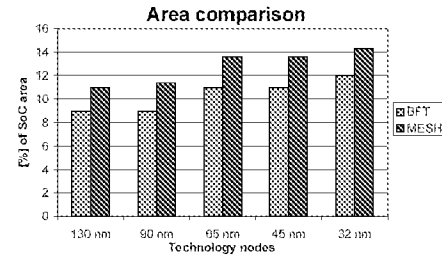
Figure 6: Infrastructure IP area overhead comparison.

8. Conclusions

In this paper, we establish a baseline procedure for performance/cost evaluation of the interconnect architectures to be adopted by the evolving MP-SoC platforms. The interconnect topologies need to be assessed in terms of two types of metrics, communication-centric parameters and the silicon area overhead due to the infrastructures. We presented a case study that illustrates the evaluation of two different topologies, namely the BFT and the MESH.

9. Acknowledgments

10. References

[1] ITRS 2003 Documents
http://public.itrs.net/Files/2003ITRS/Home2003.htm
[2] P. Magarshack, P.G. Paulin, "System-on-Chip Beyond the Nanometer Wall", *Proceedings of DAC '03,* June 2-6,2003, Anaheim, USA.
[3] Y. Zorian, "Guest editor's introduction: what is infrastructure IP?" *IEEE Design & Test of Computers*, Volume: 19 Issue: 3 , May-June 2002 pp. 3 -5.
[4] P. P. Pande, C. Grecu, A. Ivanov, R. Saleh, "Design of a Switch for Network on Chip Applications", Proceedings *of ISCAS*, Bangkok, May 2003 Vol. V, pp. 217-220.
[5] S. Kumar, et al, " A Network on Chip Architecture and Design Methodology," *Proceedings of ISVLSI,* pp. 117-124, 2002.
[6] W. J. Dally, and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks", *Proceedings of DAC 2001,* pp. 683-689, Las Vegas, Nevada, USA, June 18-22, 2001.
[7] D. Wingard, " MicroNetwork-Based Integration for SoCs", *Proc. DAC 2001,* pp. 673-677, Las Vegas, Nevada, USA, June 18-22, 2001.
[8] P. Guerrier, A. Greiner, "A generic architecture for on-chip packet-switched interconnections", *Proceedings of Design, Automation and Test in Europe Conference and Exhibition 2000*, pp. 250 –256.
[9] P. P. Pande, C. Grecu, A. Ivanov, R. Saleh, "High-Throughput Switch-Based Interconnect for Future SoCs", *Proceedings of 3rd IEEE International Workshop on SoC for Real Time Applications*, 2003, pp. 304-310, Calgary, Canada.
[10] J. Duato, S. Yalamanchili, L. Ni, *Interconnection Networks – An Engineering Approach*, Morgan Kaufmann, 2002.
[11] D. Sylvester, K. Keutzer, "Impact of Small Process Geometries on Microarchitectures in Systems on a Chip", *Proceedings of the IEEE,* Vol. 89, No. 4, April 2001, pp. 467-489.
[12] Cristian Grecu, Partha Pratim Pande, Andre Ivanov, Res Saleh, "A Scalable Communication-Centric SoC Interconnect Architecture," IEEE International Symposium on Quality Electronic Design, ISQED 2004 San Jose, California, USA, 22-24 March, 2004.

k-ary n-trees: High Performance Networks for Massively Parallel Architectures

Fabrizio Petrini and Marco Vanneschi

Dipartimento di Informatica, Università di Pisa,
Corso Italia 40, 56125 Pisa, Italy,
tel +39 50 887228, fax +39 50 887226
e-mail: {petrini,vannesch}@di.unipi.it

Abstract

The past few years have seen a rise in popularity of massively parallel architectures that use fat-trees as their interconnection networks. In this paper we study the communication performance of a parametric family of fat-trees, the k-ary n-trees, built with constant arity switches interconnected in a regular topology. Through simulation on a 4-ary 4-tree with 256 nodes, we analyze some variants of an adaptive algorithm that utilize wormhole routing with one, two and four virtual channels. The experimental results show that the uniform, bit reversal and transpose traffic patterns are very sensitive to the flow control strategy. In all these cases, the saturation points are between 35−40% of the network capacity with one virtual channel, 55–60% with two virtual channels and around 75% with four virtual channels. The complement traffic, a representative of the class of the congestion-free communication patterns, reaches an optimal performance, with a saturation point at 97% of the capacity for all flow control strategies.

Keywords: Interconnection networks, fat-trees, k-ary n-trees, routing, virtual channels, flow control, congestion-free patterns.

1   Introduction

The fat-tree is an indirect interconnection network based on a complete binary tree [9]. Unlike traditional trees in computer science, fat-trees resemble real trees, because they get thicker near the root. A set of processors is located at the leaves of the fat-tree and each edge of the underlying tree corresponds to a bidirectional channel between parent and child. A channel consists of a bundle of wires and the number of wires in a channel is called its capacity. These capacities are determined by how much hardware we can afford: this means that a fat-tree is parameterized not only in the number of processors, but also in the communication bandwidth it can support.

Routing on a fat-tree is relatively easy, since there is a unique minimal path between a pair of processors i and j: a message going from i to j goes up the internal switches of the tree until it finds the nearest common ancestor and then down to j. In the presence of channels with multiple capacity, the flow control algorithm can pick one of these wires in order to evenly distribute the messages and to minimize congestion.

Fat-trees have many nice theoretical properties. Leiserson [9] proved that, for an arbitrary message set M, *off-line* scheduling can be done optimally within a logarithmic factor of the number of processors. That is, M can be scheduled in a group of delivery cycles $M_1, M_2, \ldots M_d$ such that $d = O(\lambda \log N)$, where N is the number of processors and λ is the *load factor*. Also, fat-trees are hardware-efficient networks. The *universality theorem* from Leiserson states that a universal fat-tree of a given volume can simulate any other interconnection network of equal volume with only a polylogarithmic factor increase in the time required. These results have been extended to the more interesting *on-line* case [5]: if a set of messages has load factor λ on a fat-tree with N processors, the number of delivery cycles required is $O(\lambda + \log N \log \log N)$ with probability $1 - O(1/N)$.

The arity of the internal switches of the fat-tree increases as we go closer to the root: this makes the physical implementation of these switches unfeasible. For this reason some alternative constructions have been proposed that use building blocks with fixed arity [11]. These solutions trade connectivity with simplicity: incoming messages at a given switch in a "full" fat-tree may have more choices in the routing decision than in a corresponding network with fixed-arity switches. DeHon [3] provided an in-depth study of implementation problems, as wiring and packaging complexity and fault-tolerance. Orthogonal fat-trees are an alternative formulation that uses constant size elements. The elegant recursive construction developed by Valerio *et al.* [16] tries to maximize the number of processors when the degree of

the internal switches and the diameter of the network are physically constrained.

Fat-trees have been adopted by several parallel computers as the Connection Machine CM-5 [10], the Data Diffusion Machine [15] and the Meiko CS-2 [14].

Unfortunately, not much is known on the communication performance of the fat-trees. Most of the literature deals with the CM-5 and focuses on raw network performance [7] [12] [13]. Typical communication patterns include simple sends and ping-pong between pairs of nodes. Block permutations of data and grid shifts have been shown to have little or no contention on the CM-5. This makes the data network very efficient for regular communication patterns commonly used in numerical algorithms as the FFT. Heller [6] provided an analytical description of a class of such permutations, defined as *congestion-free*.

The remainder of this paper is organized as follows. Section 2 presents a recursive definition of fat-trees, and focuses on a parametric family of networks, the k-ary n-trees. Section 3 presents a detailed simulation model of the interconnection network for studying the impact of wormhole routing and virtual channels. The communication patterns adopted as benchmarks in the experimental evaluation are described in section 4. Using the simulation model and the benchmarks, in section 5 we evaluate the communication performance of a 4-ary 4-tree with 256 nodes. An overview of the experimental results is provided in section 6 and some concluding remarks are given in section 7.

2 k-ary n-trees

In this section we formalize the fat-trees, giving a recursive definition that is general enough to embed many different topologies that are often quoted as fat-trees. We then turn our attention to a particular subclass: the k-ary n-trees. As the k-ary n-cubes and the k-ary n-butterflies [2], the k-ary n-trees are a parametric family of regular topologies that can be built varying the two parameters k and n.

Definition 2.1 *A fat-tree is a collection of vertices connected by edges and is defined recursively as follows.*

- *A single vertex by itself is a fat-tree. This vertex is also the root of the fat-tree.*

- *If $v_1, v_2, \ldots, v_i$ are vertices and $T_1, T_2, \ldots, T_j$ are fat-trees, with $r_1, r_2, \ldots, r_k$ as roots (j and k need not to be equal), a new fat-tree is built by connecting with edges, in any manner, the vertices $v_1, v_2, \ldots, v_i$ to the roots $r_1, r_2, \ldots r_k$. The roots of the new fat-tree are $v_1, v_2, \ldots, v_i$.*

Definition 2.1 is extremely general and covers many existing examples in the literature. It includes ordinary trees,

"full" fat-trees with variable-sized switches and multiple connections between vertices and irregular constructions. The only exception being the orthogonal fat-trees, that allow connections between the roots $v_1, v_2, \ldots, v_i$. Some examples are shown in Figure 1.
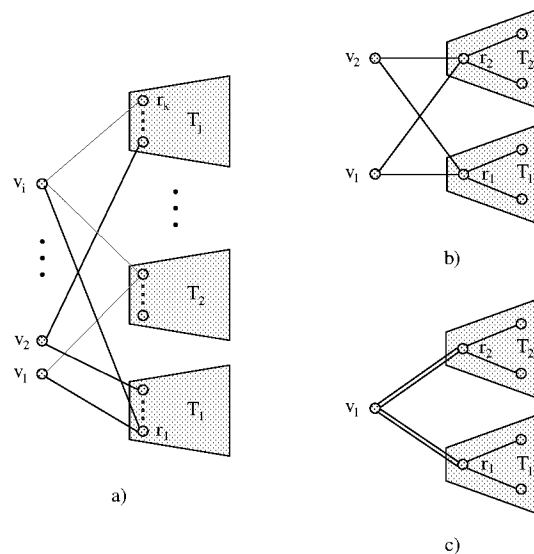
Figure 1. a) A fat-tree is recursively built connecting the new roots $v_1, v_2, \ldots, v_i$ to the roots $r_1, r_2, \ldots, r_k$ of the subtrees. b) A fat-tree with two roots. c) A fat-tree with multiple edges between the root v_1 and the roots of the subtrees T_1 and T_2.

Let's turn our attention to a particular class of fat-trees, the k-ary n-trees. k-ary n-trees borrow from a popular class of multistage interconnection networks, the k-ary n-butterflies [8] (or k-ary n-flies for short), the topology of the internal switches. The k-ary n-fly is a generalization of the butterfly and is useful to model those topologies that use communication switches with arity k greater than two. This topology has a recursive structure: one k-ary n-fly contains k butterflies of dimension $n-1$ as subgraphs. Also, each level 0 switch is linked to any level n switch by a unique path of length n, that is k-ary n-flies are *banyan* networks.

We can define the class of k-ary n-trees in the following way.

Definition 2.2 *A k-ary n-tree is composed of two types of vertices: $N = k^n$ processing nodes and nk^{n-1} $k * k$ communication switches* [1]. *Each node is an n-tuple*

---

[1] A k-ary n-tree of dimension $n = 0$ is composed of a single processing node

$\{0, 1, \ldots, k-1\}^n$, while each switch is defined as an ordered pair $\langle w, l \rangle$, where $w \in \{0, 1, \ldots, k-1\}^{n-1}$ and $l \in \{0, 1, \ldots, n-1\}$.

- Two switches

$$\langle w_0, w_1, \ldots, w_{n-2}, l \rangle \quad and \quad \langle w_0', w_1', \ldots, w_{n-2}', l' \rangle$$

are connected by an edge if and only if $l' = l + 1$ and $w_i = w_i'$ for all $i \neq l$. The edge is labeled with w_l' on the level l vertex and with w_l on the level l' vertex.

- There is an edge between the switch

$$\langle w_0, w_1, \ldots, w_{n-2}, n-1 \rangle$$

and the processing node $p_0, p_1, \ldots, p_{n-1}$ if and only if

$$w_i = p_i \quad for \ all \quad i \in \{0, 1, \ldots, n-2\}.$$

This edge is labeled with p_{n-1} on the level $n-1$ switch.

It can be easily seen that a k-ary n-tree is a fat-tree, according to Definition 2.1. In fact, the level 0 switches $\langle w, 0 \rangle$ are the roots of a k-ary n-tree whose subtrees are $(n-1)$-dimensional k-ary n-trees. Also, the labeling scheme shown in Definition 2.2 makes the k-ary n-tree a delta network [8]: any path starting from a level 0 switch and leading to a given node $p_0, p_1, \ldots, p_{n-1}$ traverses the same sequence of edge labels $(p_0, p_1, \ldots, p_{n-1})$.
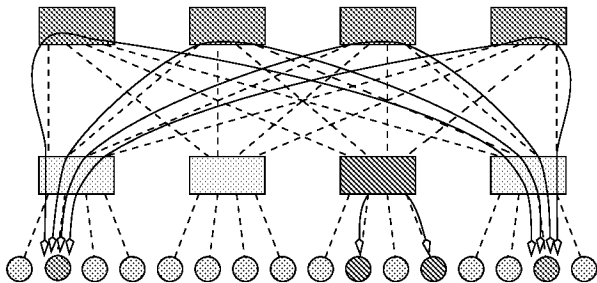

Figure 2. Routing on a 4-ary 2-tree.

As outlined above, minimal routing between a pair nodes on a k-ary n-tree can be slimly accomplished sending the message to one of the nearest common ancestors of both source and destination and from there to the destination. That is, each message experiences two phases, an *ascending* phase to get to a nearest common ancestor, followed by a *descending* phase. An example of 4-ary 2-tree is shown in Figure 2.

3   Relevant details of the network model

This section presents a router model and a simulation environment, that are used in the following sections to analyze the performance of the k-ary n-trees under various traffic loads and flow control strategies.
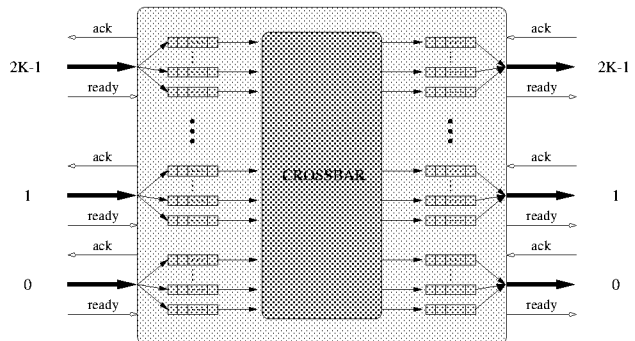

Figure 3. The internal structure of a routing switch.

Figure 3 outlines the internal structure of a $k * k$ routing switch. We can distinguish the external channels or links, the input and the output buffers or lanes that implement the buffer space of the virtual channels and an internal crossbar. The switch has $2k$ bidirectional channels and each channel on the single direction is logically composed of three interfaces: a *data path* that transmits messages on a flit level, the *ready* lines that flag the presence of a flit on the data path and specify the virtual channel where the flit is to be stored and the *ack* lines in the reverse direction that send an acknowledgment every time buffer space is released in the input lanes. The processing nodes have a compatible interface with the same number of virtual channels.

A flit is moved from an output lane to the corresponding input lane in a neighboring node in T_{link} cycles, when there is at least a free buffer position. When multiple lanes are enabled, an arbiter picks one of them according to a fair policy.

When a header flit reaches the top of an input lane, the routing algorithm tries to establish a path in the crossbar with a suitable output lane, that is neither full nor bound to another input lane. This path will remain in action till the transmission of the tail flit of the packet. Our model allows the routing of a single header at a time every $T_{routing}$ cycles. The adaptive routing algorithm studied in this paper, in the ascending phase, picks the channel with the maximum number of free lanes [2].

---

[2] A free lane is an output lane not bound to any input lane.

Although a physical link services in each direction at most one virtual channel every T_{link} cycles, multiple virtual channels can be active at the input and output ports of the crossbar. The internal flit propagation takes $T_{crossbar}$ cycles. Every time a flit is moved from an input lane to the corresponding output lane, a feedback is sent back to the neighboring switch or node to update the counter of free positions.

The experiments in this paper evaluate 4-ary 4-trees with 256 processing nodes. The number of virtual channels is varied between one, two and four and both input and output lanes have two buffer positions. Each node generates 20-flit packets with exponentially distributed inter arrival times; the destinations are distributed uniformly or according a static communication pattern, as explained in more detail in the following section. The simulator collects performance data only after 2000 cycles, to allow the network to reach steady state and each simulation is halted after 20000 cycles.

Chien in [1] has proposed a cost model to make fair comparisons between routing algorithms. It assumes a 0.8 micron CMOS gate array technology for the implementation of the routing chip. Using this model, as done in [4] for a 256 nodes hypercube, we can see that fat-trees are wire limited when we use up to four virtual channels. For this reason, in our experiments the delays $T_{routing}$ $T_{crossbar}$ and T_{link} are equalized to a single clock cycle and the clock cycle is the same for the three flow control strategies.

4   Message generation

In our model each node generates packets independently according to an exponential distribution and the destinations are chosen according to the following traffic patterns. To describe the patterns, let each node $p_0, p_1, \ldots, p_{n-1}$ also be labeled with a number in base k resulting from the concatenation of the p_i. The binary representation of $p_0 p_1 \ldots p_{n-1}$ is $a_0 a_1 \ldots a_{(n \log_2 k)-1}$[3]. Also, let $\overline{0} = 1$, $\overline{1} = 0$ and $m = \frac{n}{2} \log_2 k$.

- *Uniform traffic.* Destinations are chosen at random with equal probability between the processing nodes.

- *Complement traffic.* Each node sends only to the destination given by $\overline{a_0} \overline{a_1} \ldots \overline{a_{(n \log_2 k)-1}}$.

- *Bit reversal.* Each node sends only to the destination given by $a_{(n \log_2 k)-1} a_{(n \log_2 k)-2} \cdots a_0$.

- *Transpose.* Each node sends only to the destination given by $a_m a_{m+1} \ldots a_{2m-1} a_0 a_1 \ldots a_{m-1}$.

These traffic patterns illustrate different features. The uniform one is a standard benchmark used in network rout-

ing studies. This generation pattern can be considered representative of well-balanced shared memory computations. In the complement traffic all the packets cross the bisection of the network and traverse a path whose length is the diameter.

5   Experimental results

The performance of an interconnection network under dynamic load is usually assessed by two quantitative parameters, the *accepted bandwidth* or *throughput* and the *latency*.

Accepted bandwidth is defined as the sustained data delivery rate given some offered bandwidth at the network input. Two important characteristics are the saturation point and the sustained rate after saturation. Saturation is defined as the minimum offered bandwidth where the accepted bandwidth is lower than the global packet creation rate at the source nodes. It is worth noting that, before saturation, offered and accepted bandwidth are the same. The behavior above saturation is important because the network and/or the routing algorithm can become unstable, leading to a sharp performance degradation. We usually expect the accepted bandwidth to remain stable after saturation, both in the presence of bursty applications that require peak performance for a short period of time and applications that operate after saturation in normal conditions, e.g. when executing a global permutation pattern.

The *network* latency is the average delay spent by a packet in the network, from the insertion of the header flit in the injection lane till the reception of the tail flit at the destination. It does not include the source queuing delay. The *end-to-end latency* rises to infinity above saturation and is impossible to gain any information in this case. For this reason, the network latency is often preferred to analyze the network performance.

The experimental results of each traffic pattern are presented according to the Chaos Normal Form[4] (CNF). The CNF uses two graphs, one to display the accepted bandwidth and the other to display the network latency. In both graphs the x-axis corresponds to the offered bandwidth normalized with the unidirectional bandwidth of the links connecting the processing nodes to the network switches. This makes the analysis independent from the link bandwidth and the flit size. Also, it is worth noting that k-ary n-trees are not bisection-bandwidth limited as the k-ary n-cubes, whose CNF is normalized on the bisection bandwidth, the upper bound on the throughput for the uniform traffic.

---

[3] We will assume that k is a power of two and n is even.

[4] See http://www.cs.washington.edu/research/projects/lis/chaos/www/ presentation.html for more details on the presentation of simulation results of network routing studies.

5.1   Uniform traffic

Under uniform traffic the adaptive routing algorithm saturates at 36% of the capacity with one virtual channel, 55% with two virtual channels and 72% with four virtual channels, as shown in Figure 4 a). In all cases the post saturation behavior is stable, with a constant throughput for any offered bandwidth.

These results confirm the importance of the flow control strategy. Wormhole routed networks do not achieve optimal throughput, due to blocking problems. When a packet is stopped at an intermediate switch on the descending phase, several links on the path from the node/switch where the tail flit is stored to the current switch are blocked. Other packets could profitably use these links. In fact, the use of four virtual channels doubles the accepted bandwidth, reaching a considerable 72%. This comes at a price. The condivision of the links between two or more packets slightly increases the network latency for moderate loads. In Figure 4 b) we can see that when the offered load is 20% of the capacity, with one virtual channel the network latency is 45 cycles and 49 cycles with 2 and four virtual channels. When the load is increased at 30% of the capacity, getting closer to the saturation point of the single virtual channel, the use of more virtual channels pays in term of network latency too: in this case we have 62, 56 and 58 cycles, respectively for one, two and four virtual channels.

5.2   Complement traffic

The CNF of the complement traffic shows a surprising behavior, at least at first glance. As can be seen in Figure 4 c), the saturation point is at about 97% of the capacity for all flow control strategies. This permutation pattern doesn't create any congestion in the descending phase. The use of more than a virtual channel is counterproductive in terms of network latency (Figure 4 d): this is mainly due to the link multiplexing, that increases the tail latency. At steady state, there are as many packets in progress as the number of virtual channels in each link. The network latency with one virtual channel remains stable until the offered load is 70% of the capacity and experiences a minor increase of the head latency after this point, which remains under 30 cycles. With two virtual channels the head latency has a similar behavior, while the tail latency converges to the upper bound after 70% of the capacity. The network latency with two and four virtual channels is mainly influenced by the tail latency.

The complement traffic belongs to a wide class of permutations that map a k-ary n-tree into itself. These permutations do not generate any congestion on the descending phase and are called *congestion-free* [6].

5.3   Bit reversal and transpose traffic

Bit reversal and transpose permutations are often generated by numerical programs and are considered as an interesting benchmark for the interconnection network and the routing algorithm. These permutations have a similar distribution of the destinations in terms of distance. It can be easily noted, looking at the numerical representation shown in section 4, that in both cases there are $k^{n/2}$ nodes at distance 0 (that is source and destination are on the same node) and $(k-1)k^{n/2+i-1}$ nodes at distance $n+2i$, for all $i \in \{1,\ldots,n/2\}$. The average distance d_m traversed by a packet is given by

$$d_m = \frac{k-1}{k^{n/2}} \sum_{i=1}^{n/2} (n+2i)k^i. \tag{1}$$

For a 4-ary 4-tree $d_m = 7.125$, which is very close to the network diameter. The performance results of these communication patterns are very similar. From the CNF of the bit reversal shown in Figure 4 e) we can see that the saturation points are at 39%, 60% and 78% of the capacity for one, two and four virtual channels. An analogous behavior for the transpose can be seen in Figure 4 g), the only difference being the saturation point with one virtual channel at 38% of the capacity. After saturation, there is a linear increase of the accepted bandwidth, because there are $k^{n/2}$ nodes sending packets to themselves.

6   Discussion

The congestion-free communication patterns are an important characteristic of the k-ary n-trees. They can be routed reaching optimal performance with a simple routing algorithm and flow control strategy. They are analogous to local communication in direct topologies, as the k-ary n-cubes. The results obtained on the complement traffic generalize to the whole class of congestion-free patterns and are expected to scale with the number of nodes with an accepted bandwidth that approximates the network capacity. Message latency is only influenced by the flow control overhead and can be deterministically estimated with tight upper bounds. Using a sophisticated flow control strategy with several virtual channels is of little help in this case, because increases the network latency.

The remaining communication patterns, uniform, bit reversal and transpose, generate congestion in the descending phase and are very sensitive to the flow control strategy. As shown in Table 6, they all saturates at about 35−40% of the capacity with one virtual channel, 55−60% with two virtual channels and around 75% with four virtual channels. From these results we can argue that the expected performance of different communication patterns is mainly influenced by
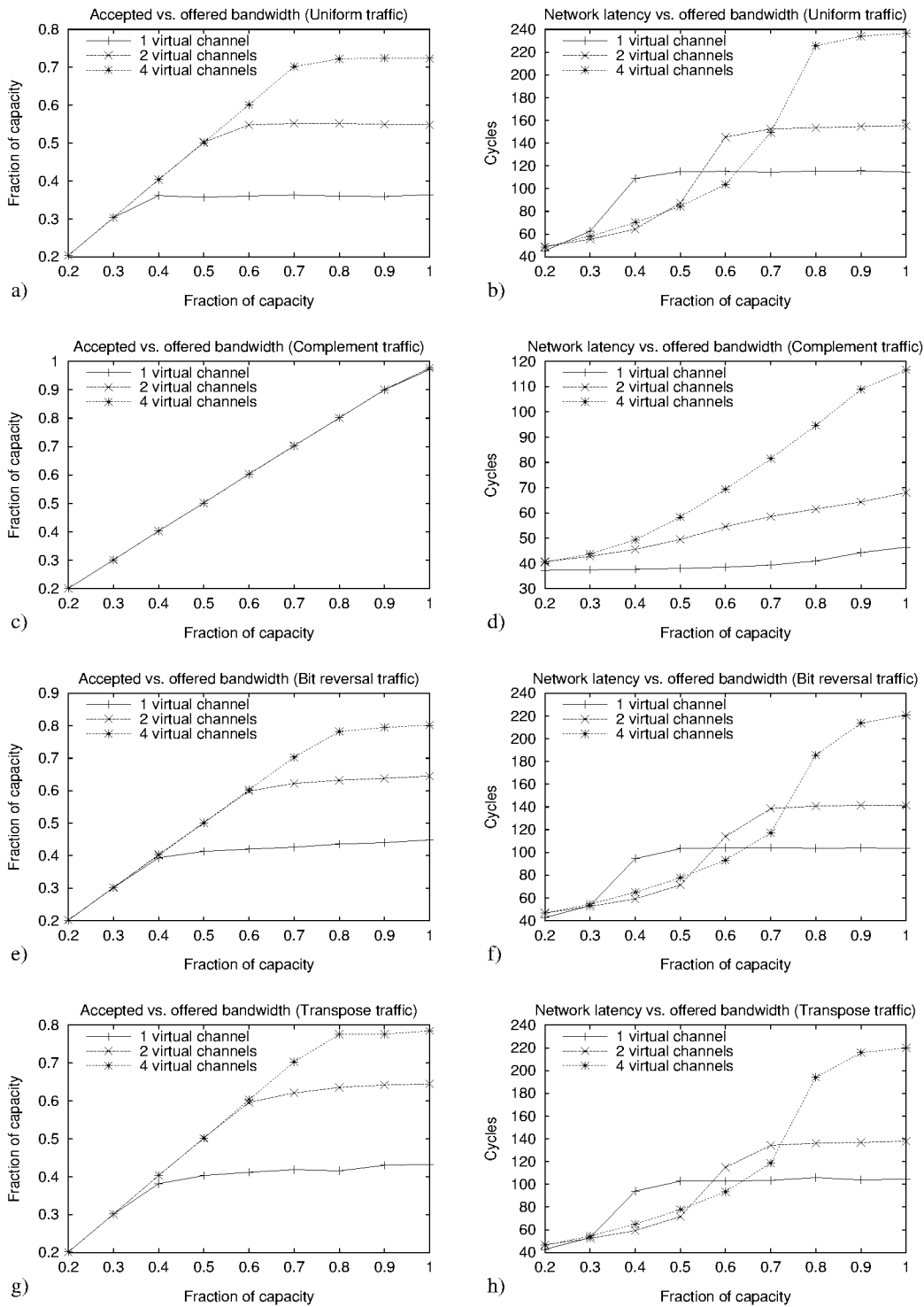
Figure 4. The Chaos Normal Form (CNF) for the uniform traffic (a-b), complement traffic (c-d), bit reversal traffic (e-f) and transpose traffic (g-h).

Traffic pattern	1 vc	2 vc	4 vc
Uniform	36	55	72
Complement	97	97	97
Bit reversal	39	60	78
Transpose	38	60	78

Table 1. Saturation points of the communication patterns.

the flow control strategy. Also, in all these cases, switching from one to four virtual channels doubles the accepted bandwidth. The performance of these patterns is not scalable. In fact, in a 4-ary 3-tree, the saturation points are about 5% above those shown in Table 6 and in a 4-ary 5-tree there is a further decrease of the same amount.

These results provide clear guidelines both for the compilation of parallel programs and for the implementation of the interconnection network. Congestion-free patterns are a powerful tool to reach optimal performance and guaranteed scalability. An important goal for a compiler is to factorize any given application in a collection of congestion-free communication patterns, when possible. In the remaining cases, virtual channels are needed to exploit the network bandwidth and to avoid the blocking problems of wormhole.

7   Conclusion and future work

In this paper we have presented a parametric family of regular topologies, the k-ary n-trees. k-ary n-trees are a particular type of fat-trees, built using processing nodes and constant arity switches interconnected in a butterfly-like topology. Adaptive routing on k-ary n-trees can be easily accomplished in two phases. In the ascending phase, each message reaches one the nearest common ancestors of source and destination and then begins the descending phase following the unique downward path to the destination. The adaptive algorithm has been extensively analyzed on a 4-ary 4-tree with 256 nodes under several traffic patterns, representative of shared memory computations and numerical algorithms. We have compared some variants of the routing algorithm using one, two and four virtual channels and the experimental results have outlined two main classes of communication patterns. The complement traffic, representative of the class of the congestion-free patterns, has reached an optimal performance, with a saturation point at 97% of the capacity for all flow control strategies. The remaining traffic patterns, uniform, bit reversal and transpose, are very sensitive to the flow control strategy.

References

[1] A. A. Chien. A Cost and Speed Model for k-ary n-cube Wormhole Routers. In *Hot Interconnects '93*, Palo Alto, California, August 1993.
[2] W. Dally. Network and Processor Architecture for Message-Driven Computers. In R. Suaja and G. Birthwhistle, editors, *VLSI and Parallel Computers*, chapter 3, pages 140–222. Morgan Kaufmann Publishers, San Mateo, CA, USA, 1991.
[3] A. DeHon. Fat-Tree Routing for Transit. Technical Report 1224, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, February 1990.
[4] J. Duato and P. López. Performance Evaluation of Adaptive Routing Algorithms for k-ary n-cubes. In K. Bolding and L. Snyder, editors, *First International Workshop, PCRCW'94*, volume 853 of *LNCS*, pages 45–59, Seattle, Washington, USA, May 1994.
[5] R. I. Greenberg and C. E. Leiserson. Randomized Routing on Fat-Trees. *Advances in Computing Research*, 5:345–374, 1989.
[6] S. Heller. Congestion-Free Routing on the CM-5 Data Router. In K. Bolding and L. Snyder, editors, *First International Workshop, PCRCW'94*, volume 853 of *LNCS*, pages 176–184, Seattle, Washington, USA, May 1994.
[7] T. T. Kwan, B. K. Tatty, and D. A. Reed. Communication and Computation performance of the CM-5. In *Supercomputing '93*, pages 192–201, November 1993.
[8] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*. Morgan Kaufmann Publishers, San Mateo, CA, USA, 1992.
[9] C. E. Leiserson. Fat-Trees: Universal Networks for Hardware Efficient Supercomputing. *IEEE Transactions on Computers*, C-34(10):892–901, October 1985.
[10] C. E. Leiserson et al. The Network Architecture of the Connection Machine CM-5. In *Proceedings of the 4th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 272–285, June 1992.
[11] C. E. Leiserson and B. M. Maggs. Communication-Efficient Parallel Algorithms for Distributed Random Access Machines. *Algorithmica*, 3:53–77, 1988.
[12] M. Lin, R. Tsang, D. H. C. Du, A. E. Klietz, and S. Saroff. Performance Evaluation of the CM-5 Interconnection Network. Technical Report AHPCRC Preprint 92-111, University of Minnesota AHPCRC, October 1992.
[13] A. Martin and D. Bader. Performance of the CM-5 ENEE 646. unpublished, January 1994.
[14] Meiko World Incorporated. *Computing Surface 2 reference manuals*, preliminary edition, 1993.
[15] H. L. Muller, P. W. A. Stallard, and D. H. D. Warren. An Evaluation Study of a Link-Based Data Diffusion Machine. In *Proceedings of the 8th International Parallel Processing Symposium, IPPS'94*, pages 115–128, Cancun, Mexico, April 1994.
[16] M. Valerio, L. Moser, and P. Melliar-Smith. Recursively Scalable Fat-Trees as Interconnection Networks. In *Proceedings of the Thirteenth IEEE International Phoenix Conference on Computers and Communications*, pages 40–46, Phoenix, AZ, April 1994.

Regenerative Feedback Repeaters for Programmable Interconnections

Ivo Dobbelaere, *Member, IEEE*, Mark Horowitz, *Senior Member, IEEE*, and Abbas El Gamal, *Senior Member, IEEE*

Abstract—The use of regenerative feedback repeaters to reduce the delay in programmable interconnections is described. A static, complementary regenerative feedback (CRF) repeater is proposed. This CRF repeater locally regenerates the new level for a fixed time after a transition has been detected. Design issues and limitations are discussed. It is shown that rising transitions can propagate faster than falling transitions through a chain of overdriven nMOS switches with CRF repeaters. Experimental results from a 1.2 μm CMOS implementation show that the loaded delay through 64 switches for static and dynamic repeaters can be reduced by a factor 1.4–2 over conventional repeaters.

I. INTRODUCTION

FIELD Programmable Gate Arrays (FPGA's) consist of an array of programmable logic cells interspersed by a programmable interconnection network as shown in Fig. 1 and provide a medium for prototyping, emulating, and implementing logic designs. The interconnection network consists of programmable switches that are organized in connection blocks and switch blocks. Several programming technologies are available such as SRAM cells, anti-fuses, and (E)EPROM's. Programmable switches may be implemented as MOS transfer gates controlled by a memory element or as anti-fuses. The performance of FPGA's is mainly limited by the delay of the programmable interconnection network. This delay increases quadratically with the number of series switches and linearly with the number of switches loading each node and is especially a problem when the programmable switches are implemented using MOS transistors since these have an appreciable resistance and capacitance.

Changing the size of the switches cannot reduce the delay much since RC remains relatively constant. The accumulation of quadratic delay can be limited by inserting repeaters as shown in Fig. 2. Conventional repeaters consist of pairs of unidirectional tristate buffers and memory cells. These repeaters have a high area penalty since after programming at least half of the buffers is not used. In addition, they contribute their own propagation delay to the signal delay, which limits the delay reduction that can be obtained.

In this paper, we describe an alternative repeater based on regenerative feedback, which offers a smaller area and delay penalty. In Section II, the principle is explained, and
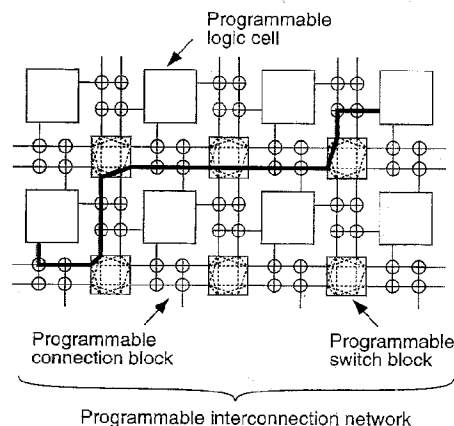
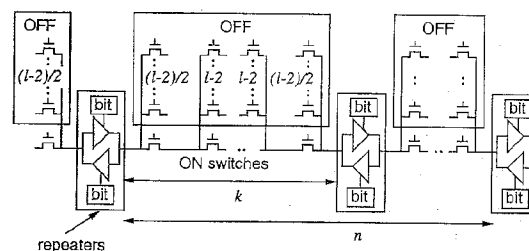Fig. 1. FPGA architecture. An interconnection obtained after programming is indicated.

Fig. 2. Interconnection network using pairs of unidirectional tristate buffers to limit delay.

a number of CMOS implementations are described. In Section III, we discuss the design issues for high performance and show a delay comparison with conventional repeaters. In Section IV, the limitations of regenerative feedback repeaters are addressed. Section V details the design of a test chip, and Section VI contains measurement results.

II. REGENERATIVE FEEDBACK REPEATERS

The area and delay penalty of providing signal amplification in both signal directions along the interconnection paths can be alleviated by using regenerative feedback repeaters. Such repeaters have a single signal terminal on which the potential is monitored. When a level change past a threshold is sensed, the repeater locally regenerates the signal, shorting out the large series resistance that is driving the interconnection. Fig. 3 conceptually shows a network with regenerative feedback repeaters.
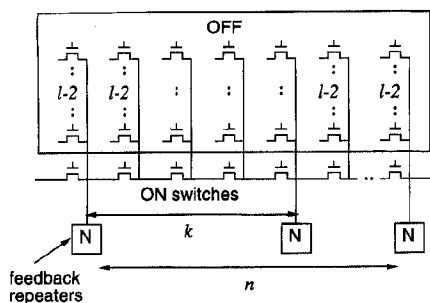
Fig. 3. Interconnection network using regenerative feedback repeaters.

Fig. 5. Waveforms in a chain with regenerative feedback repeaters.
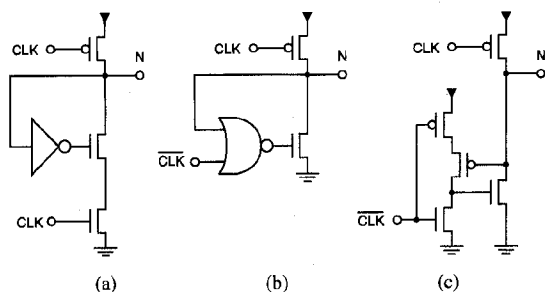

Fig. 4. Precharged regenerative feedback circuits. (a) Domino. (b) improved Domino. (c) NORA.

The principle is similar to the signal propagation in myeli-nated nerve axons [2]. Due to the periodical amplification of the signal along its propagation path, the delay increases linearly rather than quadratically with the number of stages [3]. The deviation from the initial node level that is required to activate the repeater will be referred to as the "relative threshold." The propagation delay decreases as the relative threshold is chosen smaller.

Since regenerative feedback repeaters are not inserted in the signal path, they do not contribute their own propagation delay but only add extra capacitance to the network. As a result, the interconnection delay keeps decreasing further as repeaters are placed more frequently, and the delays can be made smaller than when conventional repeaters are used. The regenerative feedback repeater only needs one buffer per repeater and no memory cells. This allows a substantial area reduction over the conventional repeater.

A. Dynamic Implementation

The simple implementation of regenerative feedback us-ing cross-coupled static inverters does not function due to hard-latching, but this is avoided by using dynamic logic. Precharged regenerative feedback repeaters have been used to reduce the delay of memory word lines [4] and carry chains [5], [6]. Fig. 4(a) shows a Domino buffer [7] with its output connected back to its input. When the precharged node N is pulled below the threshold voltage of the inverter, the low level is enforced locally. When added to a chain of switches as in Fig. 3, these circuits rapidly propagate a falling transition. By using a gated clock, as shown in Fig. 4(b), only a single nMOS is needed in the pull-down path resulting in a lower output impedance and a shorter propagation delay. The delay
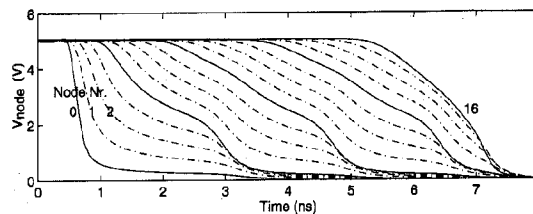
can be minimized by choosing the NOR gate threshold close to V_{DD} while maintaining a low input capacitance. By using a dynamic NOR gate, a small input capacitance can be combined with a relative threshold equal to the pMOS device threshold $|V_{TP}|$. This results in the NORA [8] feedback circuit of Fig. 4(c). The NORA implementation is the fastest but suffers from poor noise margin and poor noise performance.

In Fig. 3, the topology parameters used to describe the regenerative feedback repeater chain are indicated. The re-peater interval k is defined as the number of switches between two nodes that contain repeaters, the chain length n is the total number of series switches, and the node load l is the total number of switches attached to each intermediate node. The corresponding parameters for conventional repeaters are indicated in Fig. 2.

Fig. 5 shows the waveforms at subsequent nodes in a chain of switches with Domino feedback repeaters as in Fig. 4(b), for a repeater interval $k = 4$, chain length $n = 16$, and node load $l = 16$. The input node (Node 0) is driven by a clocked driver. The potentials on the subsequent nodes show an exponentially decaying step response until the potential on the fourth node reaches the threshold of the NOR gate. This activates the local pull-down path and speeds up the falling transition on the further nodes. In turn, the repeaters at the eighth, twelfth, and sixteenth nodes are activated.

The precharged circuit is fast, but it is not compatible with most FPGA's since it requires clocking and monotonic signaling.

B. Static Implementation

To address FPGA's, we propose the self-timed, complemen-tary regenerative feedback (CRF) circuit of Fig. 6. This is the complementary version of the circuit in Fig. 4(b), where the precharge transistor is removed and to which the complement of the pull-down circuitry is added, resulting in the NAND gate and pMOS transistor. The clock input to NOR and NAND gates, which determines when the feedback loop is turned off, is replaced by the delayed, inverted signal of node N.

In steady state, both drivers are off, so N is not actively driven. In the case of a falling transition on node N, both inputs to the NOR gate are temporarily low, and the nMOS driver is turned on for approximately the imposed delay or regeneration time t_d. Complementarily, when a rising transition is detected by the NAND gate, the pMOS driver turns on for a time t_d. Hence, this circuit can propagate both rising and falling transitions, does not require a clock, and, if
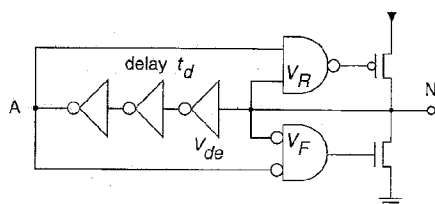
Fig. 6.   Self-timed complementary regenerative feedback (CRF) repeater.

Fig. 7.   Lumped and distributed RC networks with nonlinear resistance R_{nl} and linear capacitance C.

there is sufficient time between transitions, is not limited by ratioed logic constraints.

III. Design for Performance

In this section, the relation between design choices and performance is discussed. First, the switch choice is discussed. Since the rising and falling thresholds can be chosen independently in CRF repeaters, overdriven nMOS switches can propagate a rising transition as fast as a falling transition. The propagation speed further depends on the output impedance and on how fast the feedback turns on, which is determined by the relative threshold, the input capacitance, and the predriver strength. Finally, the delay of regenerative feedback repeaters is compared with that of conventional repeaters.

A. Switch Selection

Due to the higher current driving capability of nMOS transistors, a single nMOS transistor with gate voltage $V_{GG} = V_{DD}$ has a smaller equivalent RC for propagating a falling transition than a similarly sized CMOS transfer gate [9]. By using an overdriven gate voltage $V_{GG} \geq V_{DD} + V_{TN}$ with V_{TN}, the nMOS device threshold, the switch resistance can be further reduced, and the threshold voltage drop for a rising transition can be avoided. Such voltages above V_{DD} can be generated on-chip using a charge pump [10]. With an inverter threshold at 50% V_{DD}, the rising transition propagates roughly 1.5 times slower than the falling transition in an overdriven switch with $V_{GG} = V_{DD}+V_{TN}$. In order to obtain equal rising and falling propagation delays, the transistor sizes should be adjusted such that the gate threshold is around 40% V_{DD}.

When using CRF repeaters, the rising and falling thresholds V_R and V_F may be chosen independently. This allows a rising transition to propagate faster than a falling transition through a chain with overdriven nMOS transistors as will be shown next.

The current through an nMOS transistor with an overdriven voltage of $V_{GG} = V_{DD} + V_{TN}$ can be approximated by neglecting body and short-channel effects

$$I = B\left\{V_{GS} - V_{TN} - \frac{1}{2}V_{DS}\right\} V_{DS} \qquad (1)$$

where $B = \mu C_{ox} W/L$, in which μ is the electron surface mobility, C_{ox} is the gate oxide capacitance per unit area, and W and L are the width and length of the channel, and where V_{GS} and V_{DS} are the gate-source and drain-source potentials, respectively. Let the source and drain potentials be V_1 and V_2 with $V_1 \leq V_2$. This yields

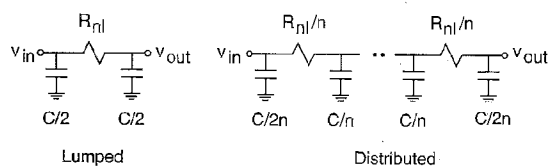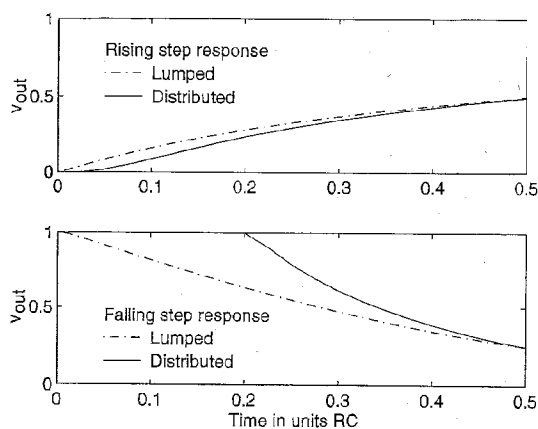$$I \approx B\{V_{DD} - \tfrac{1}{2}(V_1 + V_2)\}(V_2 - V_1). \qquad (2)$$

Fig. 8.   Rising and falling step response through a lumped and a distributed nonlinear RC network modeling overdriven nMOS transistors.

We define the equivalent resistance R as the resistance at $V_1 = V_2 = V_{DD}/2$ or

$$R = \frac{2}{BV_{DD}}. \qquad (3)$$

An overdriven nMOS switch can in first order be modeled as a nonlinear resistance

$$R_{nl}(V_1, V_2) \approx \frac{R}{\left(2 - \dfrac{V_1 + V_2}{V_{DD}}\right)}. \qquad (4)$$

We now consider an n-stage chain consisting of nonlinear series switches as above with equivalent resistance R/n and linear shunt capacitances C/n for large values of n as shown in Fig. 7. For comparison, we also consider a lumped, one-stage network. Fig. 8 shows the step response at the output of the lumped and the distributed network. For the lumped response, the falling transition is always faster than the rising transition for equal relative thresholds, i.e., when $V_R = V_{DD} - V_F$. However, for the distributed network, the falling response stalls initially, while the rising response follows quickly. This can be explained by the fact that in the case of a rising transition, the switches are initially in the region where the conductance is largest, while for the falling transitions the initial switch conductance is zero. For finite switch chains, the response falls in between the lumped and the distributed response after normalization, and for length $n > 1$, the initial rising response is faster than the initial falling response. As a result, a rising transition propagates faster than a falling transition when the relative threshold is chosen below about 30% V_{DD}.
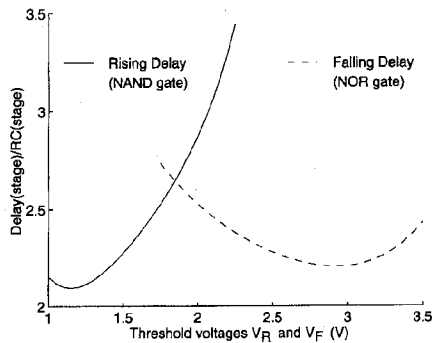
Fig. 9.   Delay per stage versus sense gate threshold voltage.

B. Rising and Falling Thresholds

Due to the asymmetry in the falling and rising step responses, a larger delay reduction is obtained by making the rising threshold V_R of the NAND gate close to GND than by making the falling threshold V_F of the NOR gate close to V_{DD}. In addition, it is difficult to design a NOR gate in CMOS with high threshold without having to resort to long-channel transistors with high input capacitance, while it is relatively easy to design a NAND gate with low threshold and low input capacitance. Both of these factors favor the propagation of the rising transition.

Fig. 9 shows how the rising and falling delay in a CRF chain with overdriven nMOS switches depends on the rising and falling threshold, respectively, for repeater interval $k = 2$. The trade-off between input capacitance and threshold voltage is apparent for the NOR gate. This comparison was obtained by changing the thresholds in a delay-optimized CRF repeater chain while keeping the driver and predriver driving strengths constant. The input capacitance becomes a less important factor as the repeater interval k increases.

C. Driver and Predriver Size

The delay further depends on the size of the driver and predriver transistors. The delay optimimum occurs for large transistor sizes due to the large node capacitances of the interconnection network, and optimization under area constraint is required.

D. Delay Comparison

Figs. 2 and 3 show the topologies that were used for the conventional repeaters and for the regenerative feedback repeaters, respectively, for delay comparison. The topology for the conventional repeaters is less general than the topology for the regenerative feedback repeaters and than the one we considered before in [11], but it is appropriate in a comparison directed toward FPGA's. Fig. 10 shows the circuit schematic of the tristate buffer used in the conventional repeater. The buffer threshold was 2.0 V.

Fig. 11 compares delays per stage, expressed in units of equivalent RC per stage, for simulations of the propagation delay in four chains with variable repeater interval k. The first curve shows the average propagation delay for a chain that uses conventional repeaters, which were optimized for
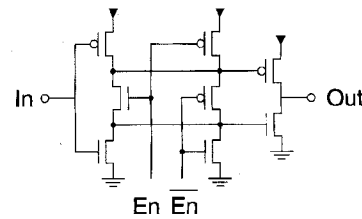

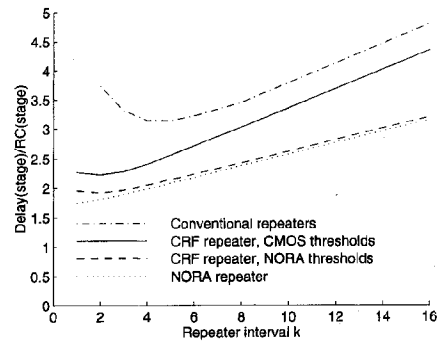Fig. 10.   Tristate buffer for conventional repeater.

Fig. 11.   Delay through 64 loaded stages versus repeater interval.

minimum delay, with equal rising and falling delay, within a given area. The second curve shows the average propagation delay for a chain that uses CRF repeaters with CMOS NAND and NOR gates. These CRF repeaters were optimized for minimum worst case delay for small repeater interval k. The sum of driver and predriver area per buffer was kept equal for both repeater types. The rising threshold was (see Fig. 9) $V_R = 1.3$ V, and the falling threshold was $V_F = 2.6$ V. For large values of the repeater interval k, the delay can be reduced further by changing the threshold of the NAND and NOR gates since the larger input capacitance has a smaller influence on the delay. A lower bound on the delay is obtained by replacing the CMOS NAND and NOR gates by NORA-like gates with relative thresholds equal to the pMOS and nMOS device thresholds $|V_{TP}|$ and V_{TN}, respectively. This yields the third curve of Fig. 11. The fourth curve shows the falling delay for a chain that used dynamic, NORA regenerative feedback repeaters, with a threshold of $|V_{TP}|$. This delay is smaller than that of the CRF repeater due to the smaller capacitance contributed by the repeater. While the noise margin of a NORA implementation is probably too small to be used on an FPGA, the ideal gate will be one with skewed thresholds in the sense gates. In this experiment, the NORA chain was 1.8 times faster than the conventional approach. The CRF chain with NORA-like sense gates was 1.6 times faster, and the CRF chain with CMOS sense gates was 1.4 times faster. A sparser placement of the smaller CRF repeaters can still yield a smaller delay than the conventional approach. Other topologies were considered as well, and speed-ups of over 2 were obtained.

IV. LIMITATIONS

In this section, we discuss the limitations and constraints that must be addressed when designing interconnections with CRF repeaters. The most obvious constraint is that the regeneration
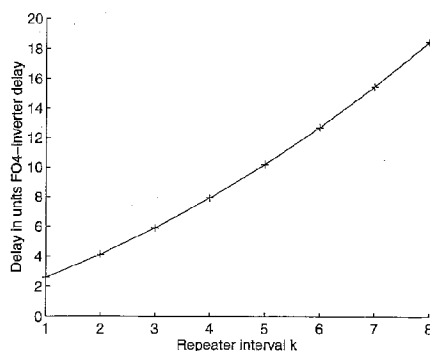
Fig. 12. Minimum delay t_d, measured at the typical process, that still ensures proper operation over all process corners.
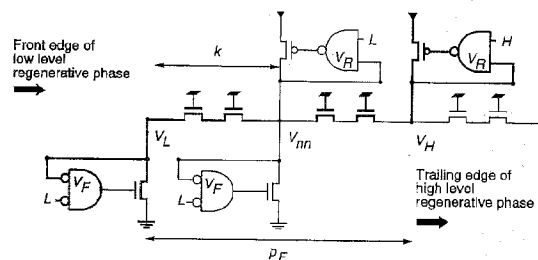

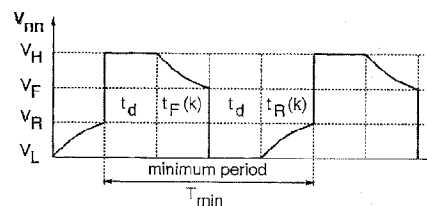Fig. 13. Minimum number of stages between the trailing edge of a rising regeneration and an oncoming falling transition.

Fig. 14. The minimum period $T_{\min}$ is determined by the minimum regeneration time t_d and the time to reach V_R or V_F.

time t_d must be chosen longer than the propagation delay between subsequent repeaters. This regeneration time also limits the maximum rate at which a signal can change. The minimum pulse width that is propagated without any disturbance is the sum of the regeneration time and the propagation delay between opposite levels. Shorter pulses are stretched out to this minimum pulse width. Sufficiently short pulses are filtered out. There is a range of pulse widths at the boundary of propagation for which the repeaters are left in a metastable state. This metastability is the main limitation of CRF repeaters because it prevents the use in an environment with glitches. The adverse effect of metastability can be limited by ensuring that the positive feedback is turned off after the regeneration time t_d.

A. Minimum Regeneration Time t_d

The regeneration time t_d of each repeater must be long enough such that the repeater can activate the next repeater before the local current path is turned off. Otherwise, stored charge may initiate the reverse transition, resulting in oscillations. This requirement must be satisfied over all process corners and operating conditions. The worst situation occurs when there is a large discrepancy between the interconnect network delay, which we implemented using only nMOS transistors, and the delay of the delay element, for which we used both nMOS and pMOS transistors. Fig. 12 shows the delay of the delay element, measured at the typical process and conditions, that ensures sufficiently long delay over all processes and conditions, for both rising and falling transitions. The delay was implemented as an odd chain of inverters and as a single inverter for $k \leq 2$. The constraint on t_d can be relieved by using a Schmitt-trigger to replace the first inverter in the delay element. In that case, some of the dependency of the driver impedance on process and operating parameters can be absorbed by the fact that the delay t_d only starts counting when the node potential reaches a level close to the new level.

B. Minimum Pulse Width

After a CRF repeater detects a transition, the corresponding new level is locally regenerated for a regeneration time t_d. Hence, a propagating rising transition is followed by a propagating high level regenerative phase of duration t_d, and

a propagating falling transition is followed by a propagating low level regenerative phase of duration t_d. A minimum phase distance p_F exists between the trailing edge of a high level regenerative phase and a subsequent falling transition. This number of stages p_F is determined by the ratioed logic constraint, as shown in Fig. 13, for which the potential on the next repeater node V_{nn} can be pulled below the falling threshold V_F by the first repeater that regenerates the oncoming falling transition against the pull-up from the last repeater that regenerates the previous rising transition. The minimum phase distance between two opposite regenerative phases for rising and falling transitions p_R and p_F may be different due to the asymmetry of the switches. By proper choice of the design parameters, we can ensure this phase distance is minimum: $p_F = p_R = 2k$. The minimum period of the input signal $T_{\min}$ can then be estimated as (Fig. 14)

$$T_{\min} \approx 2t_d + t_R(k) + t_F(k) \qquad (5)$$

where $t_R(k)$ and $t_F(k)$ are the rising and falling propagation delay, respectively, through k switches in the chain of length $2k$ between two opposite regenerative phases.

C. Glitches

The minimum signal pulse width $t_{\min}$ for undisturbed propagation is approximately $T_{\min}/2$. Pulses that are shorter than the minimum pulse width $t_{\min}$ are either propagated and stretched out to duration $t_{\min}$, or, if they are sufficiently short, are not propagated. There is a range of pulse widths around the boundary of propagation for which the repeaters are left in a metastable state as shown in Fig. 15. For example, immediately after a glitch to the low level, the circuit consists of two cross-coupled inverters. The first inverter, from V_y to V_x, consists of the pMOS driver MP4 with a load of nMOS transistors MN3-MN2-MN1 since the gate of MN1 is kept high after the glitch. The second inverter consists of the NAND gate, which
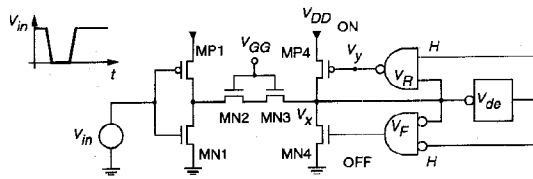
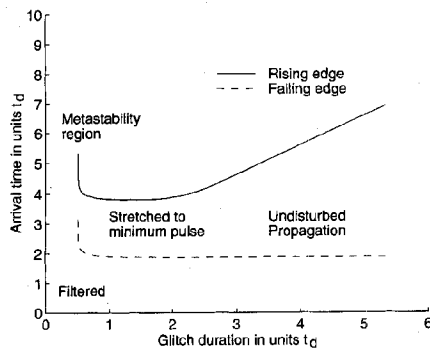Fig. 15.   Metastable state may be induced by a glitch.

Fig. 17.   CRF repeater with limited metastable state duration.

Fig. 16.   Arrival time of rising and falling glitch edges versus glitch duration for a glitch to the low level.

Fig. 18.   Chip block diagram.

acts as an inverter from V_x to V_y as long as its input from the delay element remains high. A critical glitch duration exists, for which the amount of charge placed on the network exactly puts V_x and V_y at the metastable point of these two inverters.

Fig. 16 shows the arrival times of the falling and rising edge of a low level glitch, at the output of a chain of length $n = 8$ with repeater interval $k = 2$, versus the glitch duration. Both the arrival times and the glitch duration are measured in units regeneration time t_d. Glitches shorter than $t_{min} \approx 2t_d$ are propagated but are stretched out to t_{min}. Glitches of $\approx 0.5t_d$ induce a metastable state with a long resolve time [12].

Such metastability is a fundamental property of complementary circuits that use regenerative feedback. As a result, a small range of pulse widths exists that cannot be propagated. This pulse width range can be avoided by requiring glitch-free signaling. Unfortunately, CRF repeaters are not readily compatible with conventional FPGA architectures since the output of a static programmable logic cell can produce a glitch when two inputs change within a short time interval. Glitch-free signaling can be obtained, e.g., by using self-timed design throughout the FPGA or by using clocking at the interconnect drivers. Such architectures are more complex than conventional FPGA's, and their complexity must be compared with that of using simple precharged repeaters (Fig. 4) in a dynamic FPGA.

Finally, by using separate delay elements for NAND and NOR gate, as shown in Fig. 17, with thresholds $V_{de,R} < V_R$ and $V_{de,F} > V_F$, respectively, it can be guaranteed that the positive feedback is turned off, even when in the metastable state, after delay t_d. The worst case propagation delay is now determined by a glitch that moves each repeater in turn into the metastable state for about t_d. This circuit limits the adverse effect of metastability but may leave some of the repeaters unable to fire after signals with long transition times.
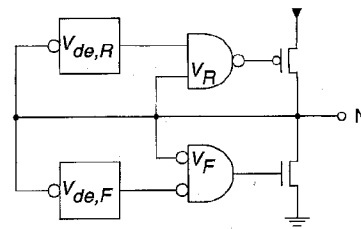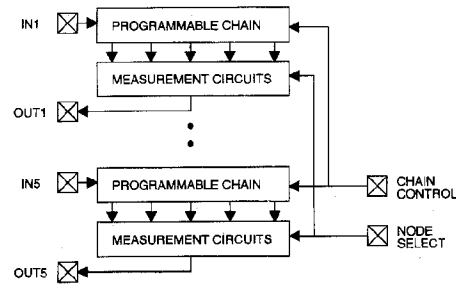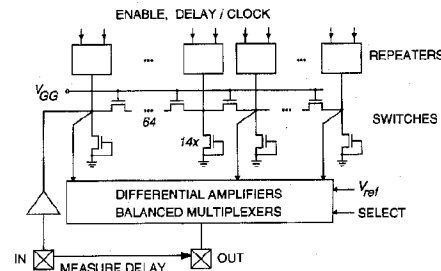
Fig. 19.   Programmable chain.

V. TEST CHIP

In order to evaluate regenerative feedback repeaters, a test chip was fabricated in a MOSIS 1.2 μm n-well CMOS technology. Fig. 18 shows a block diagram of the chip. We implemented several chains each consisting of 64 MOS switches in series. Each intermediate node is loaded by 14 switches in the off state, representing unused switches in a typical programmable interconnection of an FPGA. Each chain employs a different combination of feedback circuits and switch types. The switches are either nMOS transistors with an external gate voltage V_{GG} (Fig. 19) or CMOS transfer gates.

The voltage V_{GG} can be set above V_{DD} for lower resistance and to avoid voltage drop. In order to evaluate different repeater intervals k, NOR and NAND gates were added at the clock input or in the delay loop of the repeaters such that the feedback can be enabled externally.

The delay t_d of the CRF repeaters is controlled by an external supply. At certain nodes, the signal is observed by a measurement circuit (Fig. 20) consisting of two pMOS and two nMOS differential pairs for waveform measurements in the 0–2.5 V and the 2.5–5 V ranges, respectively. The outputs are connected to a balanced network of multiplexers
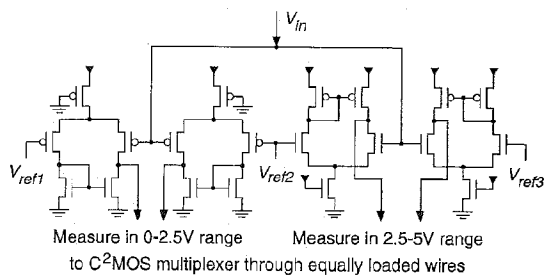
Fig. 20.   Each measurement circuit consists of a two pMOS and two nMOS differential pairs for waveform measurements in the 0–2.5 V and 2.5–5 V ranges, respectively. The reference voltages are externally supplied.

Fig. 21.   Chip micrograph.

Fig. 22.   Measured delay of dynamic regenerative feedback repeaters.
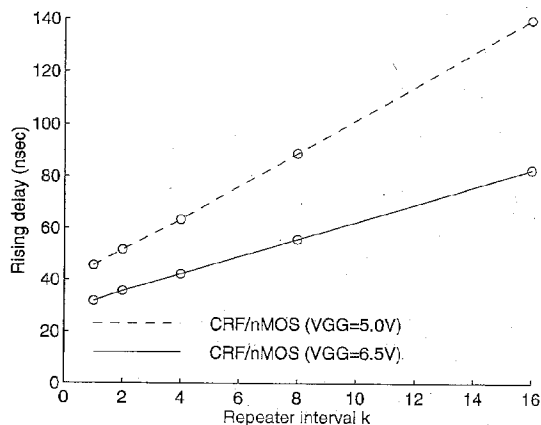
Fig. 23.   Measured delay of CRF repeaters.

for measurement of relative delays and transition times. All reported results are for chain length $n = 64$, with 14 extra switches per node, $V_{DD} = 5$ V, and nMOS switches with $V_{GG} = 6.5$ V or 5 V.

Fig. 21 shows a chip micrograph. The two dynamic chains used the Domino circuit of Fig. 4(b) with 2.5 V threshold or the NORA circuit of Fig. 4(c). The three static chains used a CRF repeater with $V_F = 2.5$ V and $V_R = 1.3$ V. One static chain used nMOS transfer gates, and two static chains used CMOS transfer gates.

VI. MEASUREMENTS

Fig. 22 compares the measured delay for Domino and NORA dynamic regenerative feedback repeaters. The difference in threshold between Domino and NORA results in a delay reduction that relatively increases with the repeater interval k. In Fig. 23, the delay of the rising transition for a static CRF repeater chain with nMOS transistors is compared for nMOS gate voltage V_{GG} at 5 V and at 6.5 V. By using overdriven switches, the rising delay is reduced by about 35%. Since in the test chip the disabled repeaters still contribute capacitance to the chain, the measured delay is linear with

the repeater interval k and does not show a minimum as the curves in Fig. 11.

As expected from the higher RC values, the chains with CMOS switches had longer delays. The rising transition propagated slightly faster than the falling transition in the chain with overdriven nMOS switches in line with the discussion in Section III.

Table I shows the chip characteristics. The area of the repeaters included enable circuitry. For comparison with conventional repeaters, we added the estimated layout area from layouts that did not include enable circuitry. The higher current consumption of the CRF repeater over the dynamic repeater is mainly due to higher capacitive loading.

VII. CONCLUSION

We described the use of regenerative feedback repeaters in programmable interconnections. A static, complementary regenerative feedback (CRF) repeater was proposed, which regenerates the signal locally for a fixed time, after a transition has been detected. Experimental results from a 1.2-μm CMOS implementation show that the loaded delay through 64 switches for static and dynamic repeaters can be reduced by a factor 1.4–2. The main limitation of CRF repeaters, and of all complementary circuits that use regenerative feedback, is the

TABLE I
CHIP CHARACTERISTICS

Process	n-well 1.2μm CMOS	
Size	3.2mm × 3.3mm	
Transistor count	21K	
Cell size (μ^2)	w/enable	[optimized]
Domino repeater	1200	[600]
CRF repeater (k=2)	3750	[2000]
Conventional repeater		[3600]
Current per chain:		
Domino rep. (k=4)	0.9mA @ 10MHz	
CRF rep. (k=4), nMOS sw.	1.5mA @ 10MHz	
I_{DD} per chain	0.1nA	

inability to propagate signals with arbitrary pulse width due to the presence of metastability. As a result, the use of CRF repeaters is limited to FPGA architectures that use self-timing or clocking to eliminate glitches.

ACKNOWLEDGMENT

The authors thank B. Fowler, M. Godfrey, B. Amrutur, and F. Klass for suggestions on design and testing, and D. Wingard for valuable comments.

REFERENCES

[1] J. Rose, A. El Gamal, and A. Sangiovanni-Vincentelli, "Architecture of field-programmable gate arrays," *Proc. IEEE*, vol. 81, no. 7, pp. 1013–1029, July 1993.

[2] J. Jack, D. Noble, and R. Tsien, *Electric Current Flow in Excitable Cells*. London, England: Oxford Univ. Press, 1985, ch. 10.

[3] I. Richer, "The switch-line: A simple lumped transmission line that can support unattenuated propagation," *IEEE Trans. Circuit Theory*, vol. CT-13, no. 4, pp. 388–392, Dec. 1966.

[4] L. Childs and R. Hirose, "An 18 ns 4 K × 4 CMOS SRAM," *IEEE J. Solid-State Circuits*, vol. SC-19, no. 5, pp. 545–551, Oct. 1984.

[5] L. Glasser and D. Dobberpuhl, *The Design and Analysis of VLSI Circuits*. Reading, MA: Addison-Wesley, 1985, p. 420.

[6] J. Schutz, "A CMOS 100 MHz microprocessor," in *ISSCC Dig. Tech. Papers*, Feb. 1991, pp. 90–91.

[7] R. Krambeck, C. Lee, and H.-F. Law, "High-speed compact circuits with CMOS," *IEEE J. Solid-State Circuits*, vol. SC-17, no. 3, pp. 614–619, June 1982.

[8] N. Gonçalves and H. De Man, "NORA: A racefree dynamic CMOS technique for pipelined logic structures," *IEEE J. Solid-State Circuits*, vol. SC-18, no. 3, pp. 261–266, June 1983.

[9] M. Shoji, *CMOS Digital Circuit Technology*. Englewood Cliffs, NJ: Prentice Hall, 1988, p. 168.

[10] R. Guo *et al.*, "A 1024 pin universal interconnect array with routing architecture," in *Proc. CICC*, May 1992, pp. 4.5.1–4.5.4.

[11] I. Dobbelaere, M. Horowitz, and A. El Gamal, "Regenerative feedback repeaters for programmable interconnections," in *ISSCC Dig. Tech. Papers*, Feb. 1995, pp. 116–117.

[12] C. Portmann and T. Meng, "Metastability in CMOS library elements in reduced supply and technology scaled applications," *IEEE J. Solid-State Circuits*, vol. 30, pp. 39–46, Jan. 1995.

Ivo Dobbelaere (S'92–M'95) was born in Ninove, Belgium on June 13, 1966. He received the B.S. degree in electrical engineering from the Katholieke Universiteit Leuven, Belgium in 1989 and the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA in 1990 and 1995, respectively.

His fields of interest are in circuits and architectures for reconfigurable VLSI systems and in high-performance and low-power logic design.

Mark Horowitz (S'77–M'78–SM'95) received the B.S. and M.S. degrees in electrical engineering from the Massachusetts Institute of Technology, Cambridge, MA in 1978 and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA in 1984.

Since September 1984, he has been working in the Computer Systems Laboratory at Stanford where he is currently an Associate Professor of Electrical Engineering. His research area is digital system design. He has lead a number of processor design projects at Stanford including MIPS-X, one of the first processors to include an on-chip instruction cache, and TORCH, a statically-scheduled, superscalar processor. He has also worked in a number of other chip design areas including high-speed memory design, high-bandwidth interfaces, and fast floating point. In 1990, he took a leave from Stanford to help start Rambus, Inc., a company designing high-bandwidth memory interface technology. His current research includes multiprocessor design, low-power circuits, memory design, and processor architecture.

Dr. Horowitz is the recipient of the 1985 Presidential Young Investigator Award, an IBM Faculty Development Award, as well as the 1993 Best Paper Award at the 1994 International Solid-State Circuits Conference.
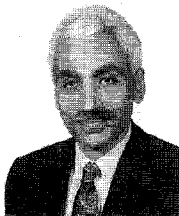
Abbas El Gamal (S'71–M'73–SM'83) received the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, in 1978.

He is currently an Associate Professor of Electrical Engineering at Stanford University. From 1978 to 1980, he was an Assistant Professor of Electrical Engineering at the University of Southern California. From 1981 to 1984, he was an Assistant Professor of Electrical Engineering at Stanford. He was on leave from Stanford from 1984 to 1987—first as Director of LSI Logic Research Lab, then as cofounder and Chief Scientist of Actel Corporation. In 1990, he cofounded SiArc, which is now part of Synopsys Inc. His research interests include VLSI circuits, architectures, and synthesis, FPGA's and mask programmable gate arrays, smart sensors, image compression, error correction, and information theory. He has authored or coauthored more than 60 papers and holds 20 patents in these areas.

Rent's Rule Based Switching Requirements

[Extended Abstract]

André DeHon
Department of Computer Science, 256-80
California Institute of Technology
Pasadena, CA 91125
andre@acm.org

ABSTRACT

A Rent's Rule characterization of recursive bisection captures a measure of the locality in a netlist or graph. From this characterization, we know we can establish a lower bound on layout area implied by wiring. When applying this lower bound to the design of programmable or switched networks, we are ultimately concerned for both the switching requirements and the wiring requirements. Switch requirements are particularly important in conventional VLSI where (a) a switchpoint consumes considerably more area than a wire crossing and (b) switchpoints must reside on the active surface, whereas wiring may take place on any of several wire layers. The most straight-forward, limited-bisection switching networks have switching requirements which grow as $O(N^{2p})$, similar to wiring requirements. In practice, however, this leaves switching dominating wiring. We show that there are limited-bisection networks with $O(N)$ switching growth and highlight some of the tradeoffs between wire utilization and switching, routing complexity, routing guarantees, and switch latency within this design space.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Network Topology*; B.7.1 [**Integrated Circuits**]: Types and Design Styles—*VLSI*

Keywords

Rent's Rule, Switching Requirements, Hierarchical Networks

1. INTRODUCTION AND BACKGROUND

1.1 Conventional, Flat Switching Networks

We have a well developed theory and design space for switching networks which can connect a number of sources (N) to a number of sinks (M). In many common cases, the sources and sinks are the same ($N = M$); we will use that
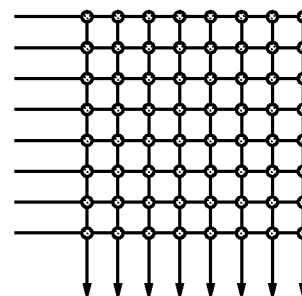
Figure 1: 8×8 Crossbar
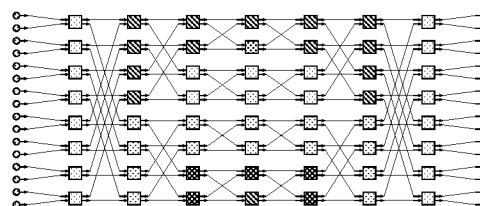
Figure 2: 16×16 Beneš Network

simplification for now, but the results have certainly been generalized for the asymmetric case. This switching theory assumes our only limitations are the number of inputs and outputs, making no assumptions or restrictions about the locality of communication.

From this switching theory, for example, we know:

- A **crossbar** (Figure 1) can connect any N inputs to N outputs with N^2 switches. The path from any input to any output goes through a single series switch, but the lines connecting input and output are each $O(N)$ long and have $2N$ switches connected to them which add some amount of capacitive loading. There is exactly one switch associated with each possible input to output connection, so routing is trivial and guaranteed as long as the input to output mapping is a permutation (or a subset of a permutation).

- a **Beneš** network (Figure 2) can also route any permutation and can do so with only $O(N \log(N))$ switches [1] [2]. It achieves this reduction in switches at the cost of a multistage arrangement. Any path from input to output goes through $2 \log_r(N) - 1$ switches (where r is the radix of each switching stage; each switching stage
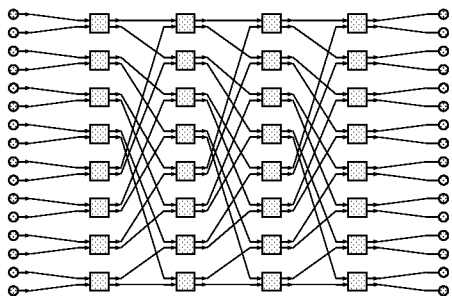
Figure 3: 16×16 Omega Network

may be an $r \times r$ crossbar) on the path from input to output and sees a constant amount of capacitive loading (for constant r) per stage, making total capacitive loading $O(\log(N))$. Total wire length for any route remains $O(N)$. Assuming the permutation is known prior to routing (offline routing), computing the configuration of switches to realize any permutation on a Beneš network is guaranteed and deterministic and can be done in $O(N \log(N))$ time.

- a **banyan** (*e.g.* butterfly, baseline, omega (Figure 3)) network retains the ability to route any single input to output connection, but is not guaranteed to route any permutation. It requires half as many switches as the Beneš network [asymptotically, still $O(N \log(N))$] and half as many switches in any path between input and output ($\log_r(N)$). Total wire length remains $O(N)$ and capacitive loading $O(\log(N))$.

While these networks allow us to reduce switching complexity from N^2 to $O(N \log(N))$, they all still require wiring area that goes as $\Omega(N^2)$ in any two-dimensional layout. This is easy to see using, for example, Thompson's argument about bisection width [12]. The minimum bisection width of any of these networks is $\Omega(N)$, meaning in any layout, there will need to be $\Omega(N)$ wires crossing between the two halves of the layout. When we are limited to 2D-VLSI, this means $\Omega(N)$ wires must cross the 1D-line that bisects the chip. That places a lower bound of $\Omega(N)$ on the width of the chip, if we assume a fixed number of wire layers. Since this property holds recursively, we can make a similar argument for the next cut of the chip and establish that both the width and height of the layout must be $\Omega(N)$, making the entire chip area $\Omega(N^2)$.

This result looks unfortunate, because it says that we can reduce the switches, but, asymptotically, we are stuck with a $\Omega(N^2)$ wiring area. Alternately, it might say that we would need $\Omega(N/\log(N))$ wiring layers to even approach the $O(N \log(N))$ switching area limit.

1.2 Rent's Rule Locality

Fortunately, we have empirical evidence, in the form of Rent's Rule [9], for example, which suggests there is additional structure to typical computations which we can exploit to avoid this limit. That is, the evidence from Rent's Rule suggests that the bisection bandwidth for a design does not need to be $O(N)$, but rather, can be $O(N^p)$ with $p < 1$ for typical designs. This gives us a model for capturing the locality structure of a design—that is, we can look at the bisection growth for a class of designs and characterize

their locality by p; this amounts to modeling non-local IO requirements for a well-clustered sub-region as having geometric growth (*e.g.* [3]).

This restricted bandwidth gives us leverage to design a class of networks with lower wire growth and will allow us to reduce the switching requirements. Using an argument like Thompson's in the non-switched case, we see we now need only $\Omega(N^p)$ wires in the bisection, this leads to a smaller 2D-VLSI area lower bound of $\Omega(N^{2p})$.

1.3 Implication on Switching Networks

Once we make this bisection restriction, we can ask what the switching requirements are for these networks. This paper highlights major points in this network design space; owing to space limitations, we cannot comprehensively cover the design space, but we can show a few major alternatives that demonstrate the key tradeoffs involved. We show that the most straightforward layouts also require $O(N^{2p})$ switches. Since switches are larger than wires in conventional VLSI, we explore network organizations which allow us to reduce the asymptotic switching requirements, sometimes at the cost of increased wiring, increased switch latency, or reduced routing guarantees; these tradeoffs are roughly analogous to the tradeoffs for flat networks reviewed above.

This paper focuses on the case where we are trying to understand how to build a universal network which will allow us to route any design with a characteristic (c,p) [from Rent's Rule IO=cN^p]—or, strictly speaking, any design whose recursive bisection is dominated by a geometric growth wiring schedule given by the Rent Relationship and set of (c,p) parameters. A separate problem arises when we want to map a design whose recursive bisection wire capacities exceed the network's limited bisection capacities. In such a case, the design must be placed sparsely on the physical node sites to meet the limited wiring capacity of the network. In earlier work [5], I show one approach to accommodating such mismatches, and use this to understand what Rent growth parameters should be used in designing a network which may see a variety of design characteristics.

2. SWITCHES

The large area of switches relative to wires is one of the key reasons that we care about the number of switches required by a network. If the wire pitch is 5 to 8 λ, the area of a wire crossing is 25–64λ^2. Contrast this with roughly $1200\lambda^2$ for the static memory cell one would use to configure a switch. Once you add a decent sized pass transistor to make up a passive switch, $2500\lambda^2$ is a more typical switch-point size. This alone makes for a factor of 40 difference in area. This area would show up directly if we were simply building a passive crossbar. In many case, we'll want more than just a pass gate for the switch. We may want to actively rebuffer the switch or even register it. Such switches can easily be 5-10Kλ^2. So, while asymptotic switch counts may match asymptotic wire area, the large constant factor area differences mean that we definitely need to watch the switch count details. (See, for example, Figure 12.)

In standard VLSI, we can use additional metal layers to increase the effective density of wire crossings. For example, with 4 metal layers, and a 7λ wire pitch, we could have two wire crossing in the same 49λ^2 of area. However, switches, in current technology, requires space on the substrate, so
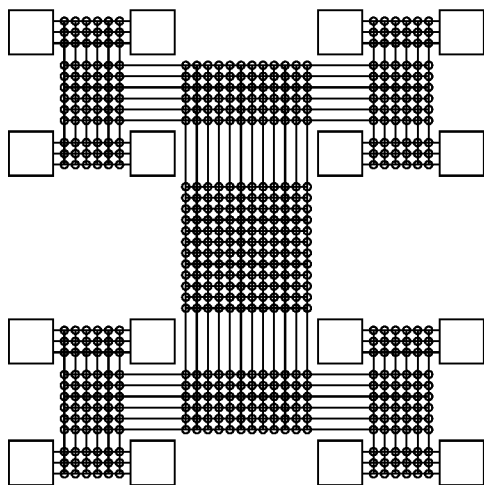
Figure 4: 16 Node Tree of Meshes

Figure 5: Switchbox Topology for Crossbar, N-choose-M, and Beneš-based Networks: this replaces the mesh switchbox used by the Tree of Meshes.

Figure 6: Parent-to-child Connections using 14-choose-10 Depopulated Crossbars: this arrangement allows each child to independently select any subset of 10 signals from the 14 parent signals.

we cannot stack two switches in a $2500\lambda^2$, regardless of the number of metal layers. So, while we may increase the wiring layers to increase the wiring density, we get no such increase in switch density. This further suggests the need to keep switch requirements significantly, and perhaps asymptotically, below wire crossings.

3. LIMITED-BISECTION SWITCHING

3.1 Leighton's Tree of Meshes

One realization of these limited-bisection bandwidth networks is Leighton's Tree of Meshes (Figure 4) [10]. Each level of the tree is connected with an $N \times M$ mesh, where the M and N can be chosen to provide the desired growth rate in the tree. For example, if we are trying to achieve a $p = 0.5$ growth rate and the lower channel has M=10, then the upper channel would have $10 \times 2^{0.5} \approx 10 \times 1.4 \approx 14$ wire channels. By keeping the $1 : 2^{0.5}$ ratio on each mesh, we realize a network whose bandwidth grows geometrically according to Rent's Rule with a $p = 0.5$. The mesh allows us to route connections from children subtrees up the network, and vice versa, and allows connections to cross between a pair of sibling subtrees. We will call the mesh a *switchbox* and will be looking at alternate scheme to realize the switching function throughout this section. We denote the number of wires into the base of the switchbox as W (*e.g.* $W = 10$ in this example).

Strictly speaking, this network may need channel capacities which are a constant factor larger than the design it is trying to route. That is, if we construct the network to a (c',p), where $c' = k \times c$, any design dominated by a (c,p) Rent Relationship will be routable. For Leighton's construction $k = 4$ [3].

For $p > 0.5$, a Tree of Meshes network with N nodes will require $O(N^{2p})$ switches. This is easy to see since the top-level mesh will be $N^p \times \left(\frac{N}{2}\right)^p$. If we recursively sum all the switches in the network, we will see that the total number of switches converges to this asymptote. For $p = 0.5$, the number of switches is $O(N \log(N))$, and the number of switches converges to $O(N)$ for $p < 0.5$. By a similar argument, the number of switches in the worst-case path across the network will be $O(N^p)$ for $p > 0.5$. Since this
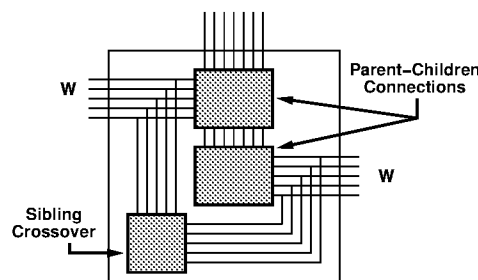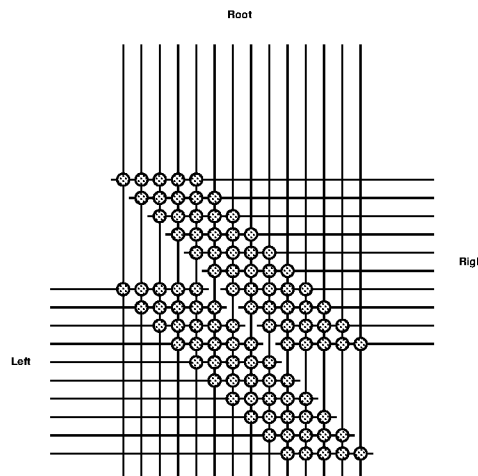
class of networks supports locality, routes only have to go through the root switchbox for the least common ancestor which contains both the source and the sink, making many paths much shorter than the worst case.

3.2 N-choose-M variation

As an alternative to the Tree of Meshes, we can use depopulated, crossbar-like switchboxes for each of the tree stages in place of Leighton's mesh switchbox (Figure 5). If we fully populate the crossbars so that we have a $W \times (2^p) W$ crossbar for each child channel connecting to the parent and a $W \times W$ connection between the two child channels, we get guaranteed routability with no constant overhead—that is, any design whose recursive bisection IO capacities do not dominate the network's channel capacities can be routed. It is also easy to see we have the same asymptotic growth in total switches as the Tree of Meshes. Now, however, we only have $2\log_2(N) - 1$ switches in the worst case path between any source and sink.

A quick look at the $W \times (2^p) W$ crossbar switch for each parent to child connection suggests that we have more switching freedom than we strictly need. In particular, the crossbar allows us to connect any of the W lower channel signals to any of the $(2^p) W$ upper channel signals, whereas we only
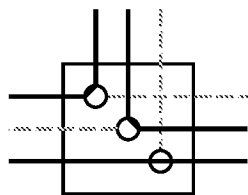
Figure 7: Single Crossover in $p = 0$ **Case: the routed (highlighted) connections show an example where the physical network needs to have 1.5× more wires in each channel than the design in order to be fully routable.**

really need to make sure that each lower channel signal is connected to some signal in the upper channel. That is, we do not really need a full crossbar; we simply need to select a subset $M = W$ of signals from a set $N = (2^p) W$. This can be done with an N-choose-M depopulated crossbar as shown in Figure 6 (for details, see [4]; this result was separately developed by Fujiyoshi et al. [7]). This arrangement needs only $M \times (N - M + 1)$ switches. In our case, that means $W \times ((2^p) W - W + 1)$. When $p > 0$ and W is large, the number of switches is roughly $(2^p - 1) W^2$. This reduces the constant factor required for these switchboxes, which is important in practice, but retains the same asymptotic growth we derived above.

By making a similar observation, the $W \times W$ child crossover can be reduced to $W^2/2$ switches. Here we again note that the exact wire is irrelevant; it is only important that we connect one channel from each child for each crossover connection. Wu and colleagues show that each channel need only fanout to half of the sibling channels in order to guarantee full connections [13]. For large p, if we exploit the crossbar connections between the root and the children when routing siblings (meaning some connections will go through two switches in making a sibling-to-sibling cross connection), we can reduce the number of crossover switches further.

An option to even further reduce crossover switches is to use a single switch for each sibling connection. That is, we number each channel coming in from the two siblings from top to bottom and provide a switch only between the channels numbered in the same manner. This switching scheme, however, can prevent us from being able to use all the wires in the channel. Used in conjunction with the N-choose-M switches above, it means some wires may be used on only one sibling side of the switchbox. Since sibling channels are only connected to a single other channel, it may leave orphaned connections on both sides of the channel. The practical effect is that channels will need to be a constant size larger than the wiring they need to support. This constant is 1.5 for $p = 0$ [14], and decreases with increasing p.

Note that the 3-way switchbox and the optimal 3-way greedy routing switchbox from [13] are degenerate cases of the N-choose-M switching scheme described here for the case where $p = 0$. Figure 7 shows the single crossover 3-way switchbox and demonstrates how this switch reduction may underutilize the physical wires by a constant factor of 1.5.

3.3   Beneš Switching

In the aforementioned networks we pay $O(W^2)$ switches for each of our intermediate switchboxes. We can, however, use our knowledge about tradeoffs in flat networks to reduce the switching cost.

First, we can replace each of the "crossbars" in the switchbox formulation of Figure 5 with Beneš networks. This reduces the total number of switches in the switchboxes to $O(W \log(W))$. Note that the switching networks between child and parent will be asymmetric, but we can easily imagine building a $((2^p) W)$-endpoint Beneš network and pruning the unused nodes on the child side of the network. This has an important effect. The total number of switches now converges to only $O(N)$ as shown in Figure 8. Note, however, that the switchbox area will remain $\Omega(W^2)$ in 2D-VLSI due to wiring since we have previously established that all of the flat networks have $O(N)$ bisection bandwidth. Also note that the $\Omega(W^2)$ switchbox wiring does not increase the total wiring lower bound from the $\Omega(N^{2p})$ bound which is true for all of these limited-bisection networks.

With the Beneš switching, the problem remains fully routable and can be done so deterministically in $O(N)$ time. Note that we simply need to route each of Beneš switches in the network, which we can do in $O(W \log(W))$ time, so the routing time analysis is the same summation as we just performed to count total switches (*i.e.* Figure 8). The major drawback of this scheme is that routes will now have to cross $O(\log^2(N))$ switches in the worst-case path between a source and a sink.

We can, in theory, use superconcentrators to achieve a linear number of switches in the child to parent path [11]. This leaves us with Beneš switches for the crossover, and the superconcentrators still have $O(\log(W))$ delay in the child to parent connection in every switchbox. This variation does not change the asymptotic bounds we have already achieved.

3.4   **Linear Population**

We can further reduce the switching requirement by only connecting each channel wire, from siblings or from parents, to a small, constant number of channels in the sibling or parent channel of the switchbox. That is, rather than providing full concentration between the various switchbox directions, we only connect to a small number of channels and use the fact that we have flexibility over multiple switchboxes to get rich routing.

One example of this is Leiserson's Butterfly Fat-Tree [8] (Figure 9); a similar version is our "linear" population tree (Figure 10). We call this "linear" because the number of switches in each switchbox is linear in the channel width, W. Using these topologies, we can program the growth rate similar to Leighton's Tree of Meshes; however, it is most convenient to grow tree stages in quanta. In the scheme used for HSRA, each stage would either be a 1:1 stage or a 2:1 stage. By choosing which levels do (1:1) or do not (2:1) reduce the total wires out of the top of the switchbox, we can target a particular p value over multiple stages of growth (Figure 11).

The number of switches in any of these linearly populated networks converges to a constant independent of the number of tree levels. For example, if we assume a 4-ary butterfly fat tree with switches with 4 down links and 2 up links, as shown in Figure 9, then the total number of switches is at most $\frac{N}{2}$. To see this, note that each group of 4 leaf nodes needs one

Here we show that the total number of switches in a limited-bisection network using Beneš switchboxes is $O(N)$. There are $cW \log(W)$ switches in each switchbox. Summing over all switchboxes in the tree, we get:

$$N_{switch} = c\left(1 \cdot (N^p)\log(N^p) + 2 \cdot \left(\frac{N}{2}\right)^p \log\left(\left(\frac{N}{2}\right)^p\right) + 4 \cdot \left(\frac{N}{4}\right)^p \log\left(\left(\frac{N}{4}\right)^p\right) + \cdots \right.$$
$$\left. + \frac{N}{2} \cdot \left(\frac{N}{N/2}\right)^p \log\left(\left(\frac{N}{N/2}\right)^p\right) + N \cdot \left(\frac{N}{N}\right)^p \log\left(\left(\frac{N}{N}\right)^p\right)\right)$$

This is better re-expressed using $n = \log(N)$ and extracting the common term p.

$$N_{switch} = cp\left(N\left(\frac{2^p}{2}\right)^n \cdot n + N\left(\frac{2^p}{2}\right)^{n-1} \cdot (n-1) + N\left(\frac{2^p}{2}\right)^{n-2} \cdot (n-2) \cdots \right.$$
$$\left. + N\left(\frac{2^p}{2}\right) \cdot 1 + N \cdot 0\right)$$

Now, if we pull out N and set $r = \frac{2}{2^p}$, we get:

$$N_{switch} = cpN \sum_{n=1}^{\log(N)} \left(\frac{n}{r^n}\right)$$

This is certainly less than the infinite sum where we do not bound n. We further note that the ratio of consecutive terms in this series is:

$$\frac{n+1/r^{n+1}}{n/r^n} = \left(\frac{1}{r}\right)\left(\frac{n+1}{n}\right) = \left(\frac{2^p}{2}\right)\left(\frac{n+1}{n}\right)$$

For $p < 1$ and $n > \frac{2}{2-2^p}$, this ratio is strictly less than one. Hence, this series is bounded by a geometric series and, consequently, the summation is asymptotically bounded by a constant. Since the sum is a constant, we have a total number of switches which is simply a constant times p and N, or $O(N)$ switches. $\square$

Figure 8: Total Switch Accounting when using Beneš Networks in Switchboxes
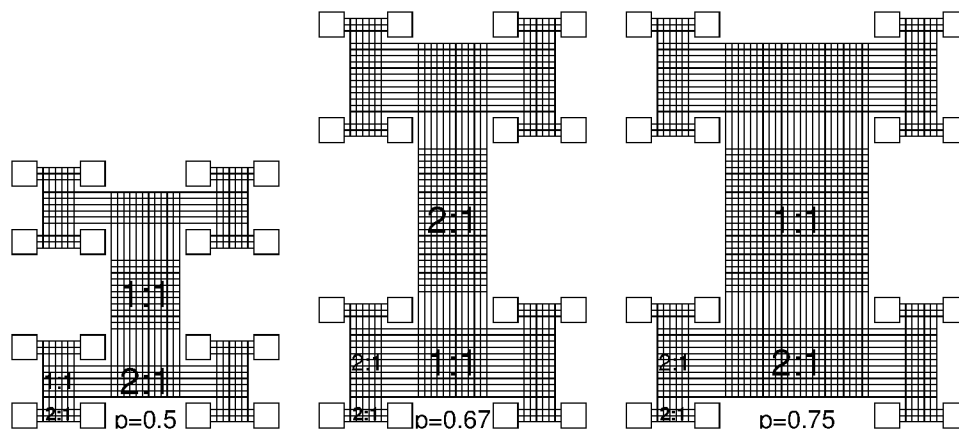
Figure 11: Programming Growth for Tree of Meshes: note that the number of base channels (c) is 3 in all these examples.
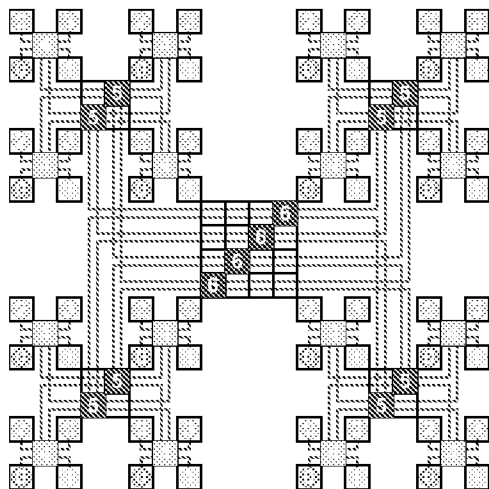
Figure 9: 4-ary Butterfly Fat-Tree
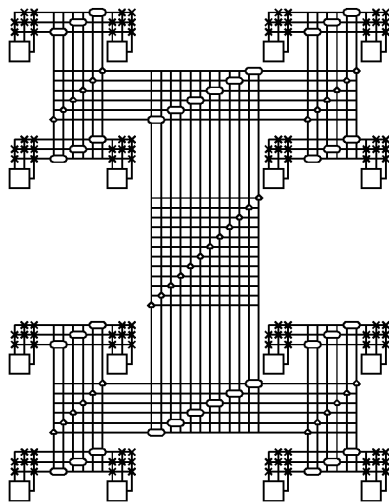
Figure 10: Hierarchical Network with Linear Switch Population

switch at the lowest level (labeled 4 in Figure 9). At the next level, we need half as many switches (every 4 switches on the lower level needs 2 switches at the next level). This relationship continues with each succeeding level requiring half as many switches as the level before. Consequently, the number of switches needed can be calculated as a classical geometric series:

$$N_{switch} = \frac{N}{4} + \frac{1}{2}\left(\frac{N}{4}\right) + \frac{1}{4}\left(\frac{N}{4}\right) + \frac{1}{8}\left(\frac{N}{4}\right) + \cdots \leq \frac{N}{2}$$

These linear organizations also allow us to have only a single switch at each tree level, such that the total switches along the worst-case path is again only $2\log_r(N) - 1$.

This linear population may come at the cost of routability and wire utilization. It is clear from the 3-way single-switch sibling-crossover case (Figure 7) that we will not be able to use every wire in the network perfectly. However, even if we had to pay a small constant factor in wire utilization (*e.g.* we had to guarantee to have 2× as many available wires in each channel as wires in the netlist), this tradeoff between wires and switches would still be a net win, as shown in Figure 12. We have empirical evidence that a small constant overhead is adequate to route typical benchmark circuits. However, we have not yet proved the existence or non-existence of a constant mapping ratio. Our working hypothesis is that there may not be a constant mapping ratio, but that typical circuits do have additional structure which makes this work. These represent important open questions in our understanding of switching requirements for these networks.

These networks with a linear number of switches are particularly attractive when we consider multi-level wiring. Since the number of switches per node converges to a constant independent of N, it may be possible to layout the networks so that they take up a fixed amount of active substrate area and support the super-linear interconnect requirements with additional metal layers. For the linear populated network, like the Butterfly Fat Tree ($p = 0.5$), I have shown [6] that it is possible to layout the network in $\Theta(N)$ active area using $\Theta(\log(N))$ wiring layers.

4. NETWORK ARITY

So far, I have described and shown most of these limited-bisection networks as two-ary trees—that is, each switchbox has only two children. Another topological option is to vary the arity of the switching nodes. In general, increasing the arity will decrease the number of switchboxes, and hence switches, on the path between the worst-case source and sink pair. For example, going from a 2-ary network to a 4-ary network will cut the number of series switches in half for the linear and N-choose-M populated networks. If we are trying to maintain the same wiring guarantees for a given set of (c, p) parameters, the flattened network will have more wires in some intermediate channels than the non-flattened network since bandwidth reduction only occurs at the switchboxes which are now less frequent. Flattening from 2-ary to slightly larger arity may be able to reduce the total number of switches in the network in some cases, but, asymptotically, increased arity will increase switch count. Flattening by any constant amount will only change the constants.

N-choose-M both   Linear
$c = 5$ $p = 0.67$ $c = 10$
(wires shown)   (wires not shown)
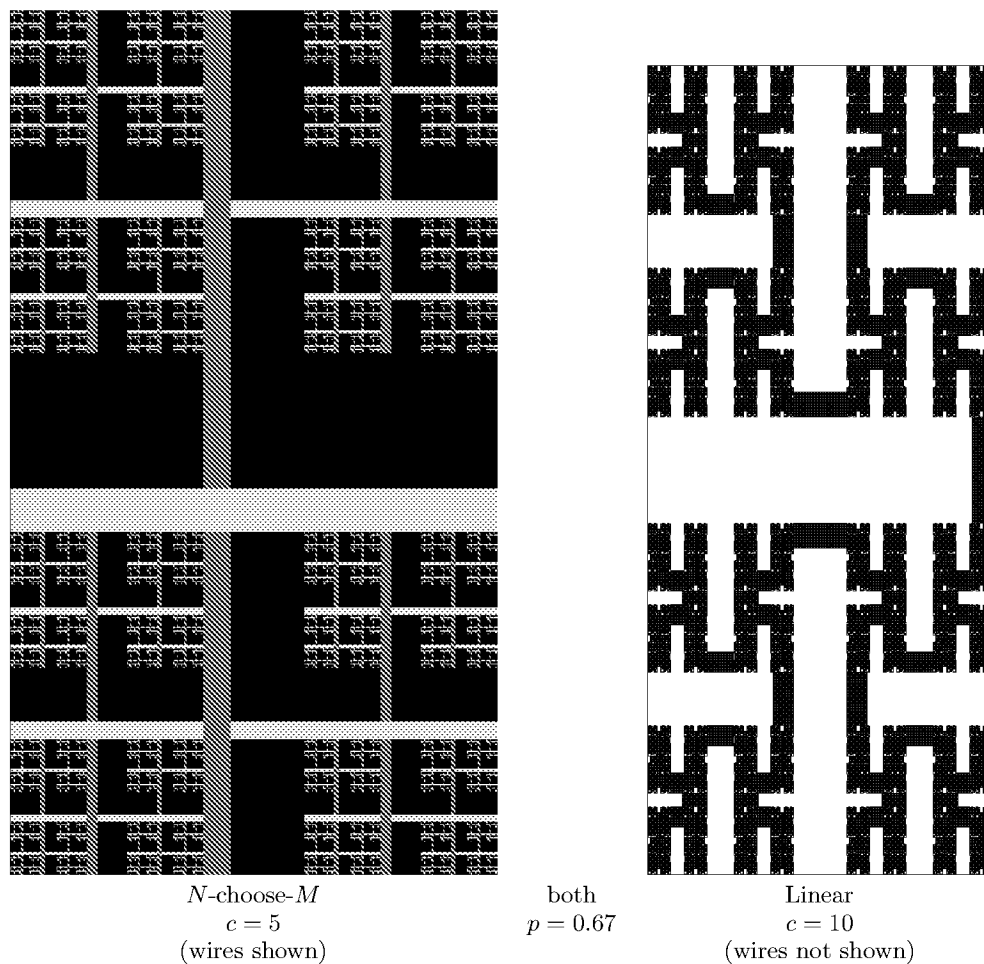
Figure 12: Area Comparison for a 1024 Node Network: this shows that even if we cannot use all the wires in a linearly populated network, typical constants work out that the total area is still less than that of a network which provides switching so that all wires can be used perfectly. This comparison assumes two metal layers dedicated to routing. The non-black area in the N-choose-M case shows the horizontal and vertical wire layers. It should be clear from these diagrams, that additional metal layers would not substantially reduce the area of the N-choose-M layout since it is already switch dominated; the linear layout is clearly wire-channel dominated and could be reduced if additional metal layers were available for routing.

5. SUMMARY

Limited-bisection networks designed with growth according to Rent's Rule, allow us to reduce switch requirements as well as wire requirements. Since switches are large relative to wires and can only exist on the substrate in VLSI, switch area can easily dominate the wiring area in these networks. We showed a number of design points in this space, with switch areas which range from $O(N^{2p})$, matching wire area asymptotes when we have fixed wire layers, down to $O(N)$ switch area, which may permit $O(N)$ layout area with sufficient metalization.

6. REFERENCES

[1] V. Beneš. Permutation groups, complexes, and rearrangeable multistage connecting networks. *Bell System Technical Journal*, 43:1619–1640, July 1964.

[2] V. Beneš. *Mathematical Theory of Connecting Networks and Telephone Traffic*. Academic Press, New York, NY, 1965.

[3] S. Bhatt and F. T. Leighton. A framework for solving vlsi graph layout problems. *Journal of Computer System Sciences*, 28:300–343, 1984.

[4] A. DeHon. Entropy, counting, and programmable interconnect. In *Proceedings of the 1996 International Symposium on Field Programmable Gate Arrays*. ACM/SIGDA, February 1996. See extended version <http://www.cs.caltech.edu/~andre/abstracts/entropy_fpga96.html>.

[5] A. DeHon. Balancing interconnect and computation in a reconfigurable computing array (or, why you don't really want 100% lut utilization). In *Proceedings of the International Symposium on Field Programmable Gate Arrays*, pages 69–78, February 1999.

[6] A. DeHon. Compact, multilayer layout for butterfly fat-tree. In *Proceedings of the Twelfth ACM Symposium on Parallel Algorithms and Architectures (SPAA '2000)*, pages 206–215. ACM, July 2000.

[7] K. Fujiyoshi, Y. Kajitani, and H. Niitsu. Design of minimum and uniform bipartites for optimum connection blocks of fpga. *IEEE Transactions on Computer-Aided Design of Integrated Circuits*, 16(11):1377–1383, November 1997.

[8] R. I. Greenberg and C. E. Leiserson. *Randomness in Computation*, volume 5 of *Advances in Computing Research*, chapter Randomized Routing on Fat-Trees. JAI Press, 1988. Earlier version MIT/LCS/TM-307.

[9] B. S. Landman and R. L. Russo. On pin versus block relationship for partitions of logic circuits. *IEEE Transactions on Computers*, 20:1469–1479, 1971.

[10] F. T. Leighton. New lower bound techniques for vlsi. In *Twenty-Second Annual Symposium on the Foundations of Computer Science*. IEEE, 1981.

[11] N. Pippenger. Superconcentrators. *SIAM Journal of Computing*, 6(2):298–304, 1977.

[12] C. Thompson. Area-time complexity for vlsi. In *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*, pages 81–88, May 1979.

[13] Y.-L. Wu, D. Chang, M. Marek-Sadowska, and S. Tsukiyama. Not necessarily more switches more routability. In *Proceedings of the 1996 Asia Pacific Design Automation Conference*, 1996.

[14] Y.-L. Wu, S. Tsukiyama, and M. Marek-Sadowska. Graph based analysis of 2-d fpga routing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15(1):33–44, January 1996.

Web links for this document: <http://www.cs.caltech.edu/research/ic/abstracts/rentsw_slip01.html>

Substitute for form 1449/PTO **INFORMATION DISCLOSURE STATEMENT BY APPLICANT** *(Use as many sheets as necessary)*	***Complete if Known***	
	Application Number	16/202067
	Filing Date	12-4-2018
	First Named Inventor	Venkat Konda
	Art Unit	
	Examiner Name	

Sheet	1	of	1	Attorney Docket Number	V-0070US

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No.[1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
	1	Guy Lemieux et.al., Generating highlyroutablesparse crossbars for PLDs. In ACMISIGDA Int'l. Symposium on Field Programmable Gate Arrays, pp155-164, Monterey, CA, February 2000	
	2	S. Sivaswamy et. al., "HARP: hard-wired routing pattern FPGAs", FPGA'05, Monterey, California, USA, February 20–22, 2005.	
	3	Yeh, C.-H., E.A. Varvarigos, and B. Parhami, "Efficient VLSI layouts of hypercubic networks," Proc. Symp. Frontiers of Massively Parallel Computation, Feb. 1999	
	4		
	5		
	6		
	7		
	8		
	9		
	10		

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

1 Applicant's unique citation designation number (optional). 2 Applicant is to place a check mark here if English language Translation is attached.
This collection of information is required by 37 CFR 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

Privacy Act Statement

The **Privacy Act of 1974 (P.L. 93-579)** requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (*i.e.*, GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

Efficient VLSI Layouts of Hypercubic Networks

Chi-Hsiang Yeh, Emmanouel A. Varvarigos, and Behrooz Parhami

Department of Electrical and Computer Engineering

University of California, Santa Barbara, CA 93106-9560, USA

Abstract

In this paper, we present efficient VLSI layouts of several hypercubic networks. We show that an N-node hypercube and an N-node cube-connected cycles (CCC) graph can be laid out in $4N^2/9 + o(N^2)$ and $4N^2/(9\log_2^2 N) + o(N^2/\log^2 N)$ areas, respectively, both of which are optimal within a factor of $1.\overline{7} + o(1)$. We introduce the multilayer grid model, and present efficient layouts of hypercubes that use more than 2 layers of wires. We derive efficient layouts for butterfly networks, generalized hypercubes, hierarchical swapped networks, and indirect swapped networks, that are optimal within a factor of $1 + o(1)$. We also present efficient layouts for folded hypercubes, reduced hypercubes, recursive hierarchical swapped networks, and enhanced-cubes, which are the best results reported for these networks thus far.

1. Introduction

The derivation of efficient VLSI layouts for interconnection networks is important, since it improves the cost-performance of the resulting parallel architecture, both by reducing its cost (fewer chips, boards, and assemblies) and by lowering various performance hindrances, such as signal propagation delay, drive power, and fraction of data transfers to off-chip destinations. Efficient layouts for several interconnection networks can be found in [5, 6, 8, 12].

Hypercubes, butterfly networks [13], and cube-connected cycles (CCC) [16] are among the most important interconnection networks. In [6], a collinear layout of an N-node hypercube that requires $N - \log_2 N$ tracks was proposed. In this paper, we show that the collinear layout of a hypercube can be considerably improved to one that uses $\lfloor 2N/3 \rfloor$ tracks, which is within a factor of $1.\overline{3} + o(1)$ from a trivial lower bound. We also show that an N-node hypercube can be laid out in $4N^2/9 + o(N^2)$ area, which is within a factor of $1.\overline{7} + o(1)$ from a trivial lower bound and improves the layout area given in [6] by a factor of $2.25 + o(1)$. We also show that an N-node CCC can be laid out in

$4N^2/(9\log_2^2 N) + o(N^2/\log^2 N)$ area, which is smaller than the layout area given in [7] by a factor of $1.125 + o(1)$, and is within a factor of $1.\overline{7} + o(1)$ from a lower bound. The layouts for the hypercube and CCC given in this paper are the best results reported thus far for these networks.

We introduce the multilayer 2-D grid and multilayer 3-D grid models for VLSI layouts of networks. The motivations for using the multilayer grid model include significant reduction in layout area, volume, and maximum wire length. In particular, we show that an N-node hypercube can be laid out in $\frac{16N^2}{9L^2} + o\left(\frac{N^2}{L^2}\right)$ area, $\frac{16N^2}{9L} + o\left(\frac{N^2}{L}\right)$ volume, and $\frac{2N}{3L} + o\left(\frac{N}{L}\right)$ maximum wire length when we use L layers of wires, L is even, and $L = o(\sqrt{N}/\log N)$.

We derive tight bounds on the VLSI area of generalized hypercubes, hierarchical swapped networks (HSNs) [23, 25], and indirect swapped networks (ISNs) [22], which are optimal within a factor of $1 + o(1)$. Moreover, we present efficient layouts for butterfly networks [13], folded hypercubes [1], reduced hypercubes (RH) [29], recursive hierarchical swapped networks (RHSNs) [23, 25], and enhanced-cubes [21], which are the best results reported for these networks so far in the literature. Our layout method and lower bound techniques can also be extended to a variety of other networks [25, 28].

The organization of the remainder of the paper is the following. In Section 2, we present efficient layouts for hypercubes, folded hypercubes, CCC, and reduced hypercubes. In Section 3, we introduce the multilayer grid model and present multilayer layouts of hypercubes. In Section 4, we present efficient layouts for butterfly networks, generalized hypercubes, HSNs, RHSNs, and ISNs. In Section 5, we show that several of the layouts given in Section 4 have areas that are optimal within a factor of $1 + o(1)$. In Section 6 we present our conclusions.

2. VLSI layouts for hypercubes, CCCs, and related networks

In this section, we present a method for laying out hypercubes, folded hypercubes, CCC, and reduced hypercubes.

We use the extended version [8, 17, 25] of the grid model, also called Thompson's model [18], for the VLSI layout of networks whose node degrees may be larger than 4. In this model, a network is viewed as a graph whose nodes correspond to processing elements and edges correspond to wires. The graph is then embedded in a 2-D grid, where wires have unit width and a node of degree d occupies a square of side d. The wires can run either horizontally or vertically along grid lines.

The area of a layout is defined as the area of the smallest rectangle that contains all the nodes and wires. When there are two layers of wires, it is guaranteed that we can lay out the network within the area. In Section 3, we modify layouts derived in this section to obtain layouts that use more than two layers of wires and have smaller area

2.1. Efficient layouts for hypercubes and several variants

In this subsection, we first derive a collinear layout for the hypercube and then use it to obtain efficient 2-D layouts for hypercubes, folded hypercubes, and their variants.

In a collinear layout all nodes are placed on the same line. A collinear layout that requires $N - \log_2 N$ tracks was presented in [6] for an N-node hypercube. In what follows, we improve on their result by finding a collinear layout that uses only $\lfloor 2N/3 \rfloor$ tracks.

To describe the hypercube layout we use a bottom-up approach, starting with a 2-dimensional hypercube, and inductively moving to hypercubes of higher dimensions. A collinear layout of a 2-dimensional hypercube can be obtained by placing the 4 nodes along a row, connecting node 0 with node 1, and node 1 with node 3, through wires in the first track, and then connecting node 0 with node 2, and node 2 with node 3, through wires in the second track (see Fig. 1a). Clearly, this layout requires 2 tracks.

Assume that we have a collinear layout for an n-cube that requires $f(n)$ tracks, where n is even. To obtain the collinear layout of an $(n + 1)$-cube, we start with the layouts of two n-cubes. By doubling the horizontal space, we can place the ith node of the second layout adjacent (from the right) to the ith node of the first layout. We also double the number of tracks (i.e., vertical space) to accommodate the $2f(n)$ tracks of the two layouts. Moreover, to connect the two n-cubes into an $(n + 1)$-cube, we need an extra track which contains paths connecting adjacent nodes (i.e., the ith nodes of the two layouts). Therefore, the number of tracks required for the collinear layout of the $(n + 1)$-dimensional hypercube is $f(n + 1) = 2f(n) + 1$, assuming that n is even.

To obtain the collinear layout of an $(n + 2)$-cube, we start with the layouts of four n-cubes. By increasing the horizontal space by a factor of 4, we can place the nodes with the same ID of the four layouts adjacent to each other. We also

have to increase the number of tracks by a factor of 4 to accommodate the $4f(n)$ tracks of the four layouts. Finally, to connect the four n-cubes into an $(n + 2)$-cube, we need two extra tracks for laying out the paths that connects each set of 4 nodes of the n-cubes that have the same ID as a 2-cube (see Fig. 1b). Therefore, we have $f(n + 2) = 4f(n) + 2$ when n is even. Since $f(2) = 2$, we obtain the following theorem.

Theorem 2.1 *The number of tracks required for the collinear layout of an N-node hypercube is $\lfloor \frac{2N}{3} \rfloor$.*

Proof: When n is even, we have

$$f(n) = 4f(n - 2) + 2$$

and $f(0) = 0$, where $n = \log_2 N$. Therefore,

$$f(n) = 2^2 f(n - 2) + 2^1 = 2^4 f(n - 4) + 2^3 + 2^1$$

$$= 2^n f(0) + 2^{n-1} + 2^{n-3} + \cdots + 2^1$$

$$= \frac{N}{2} \left(1 + \frac{1}{4} + \frac{1}{16} + \cdots + \frac{4}{N} \right) = \frac{2}{3}(N - 1) = \left\lfloor \frac{2N}{3} \right\rfloor$$

when $n = \log_2 N$ is even. For the case where n is odd, we have

$$f(n) = 2f(n - 1) + 1 = \frac{4}{3} \left(\frac{N}{2} - 1 \right) + 1 = \frac{2}{3}N - \frac{1}{3} = \left\lfloor \frac{2N}{3} \right\rfloor.$$

$\square$

To lay out an n-cube on a 2-D grid, we let $n = n_1 + n_2$, and use 2^{n_1} copies of the collinear layout of an n_2-cube, each placed along a row. We then connect the 2^{n_1} nodes that belong to the same column (i.e., nodes that have the same ID within each of the n_2-cubes) vertically according to the collinear layout of an n_1-cube (see Fig. 1c). Note that when n_2 (and/or n_1) is odd, we can eliminate 2^{n_1} horizontal tracks (and/or 2^{n_1} vertical tracks, respectively) by moving the wires connecting neighboring nodes to horizontal tracks (and/or vertical tracks, respectively) between nodes. When n_2 (and/or n_1) is even, we can also remove 2^{n_1} horizontal tracks (and/or 2^{n_1} vertical tracks, respectively) after some minor modifications at the expense of longer wires. Since in the VLSI model a node of degree $\log_2 N$ requires a square of side $\log_2 N$, we need an extra $O(N\sqrt{N}\log N)$ area to accommodate the nodes. By choosing $n_1 = \Theta(n_2)$, we obtain the following theorem.

Theorem 2.2 *An N-node hypercube can be laid out in $\frac{4}{9}N^2 + o(N^2)$ area.*

The layout area given in Theorem 2.2 for the hypercube improves the corresponding area given in [6] by a factor of $2.25 + o(1)$, and is the best result reported thus far for hypercubes. The area is within a factor of $1.\overline{7} + o(1)$ from a trivial
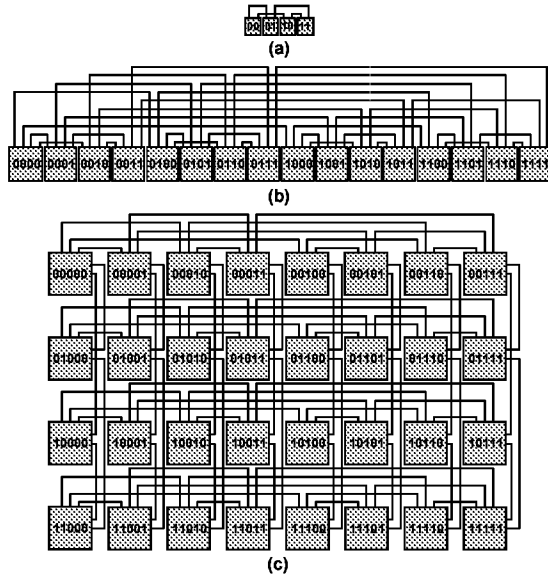
Figure 1. collinear layouts of (a) a 2-cube and (b) a 4-cube. (c) A 2-D layout of a 5-cube.

lower bound $N^2/4$ [which follows from the fact that the area of a graph is at least equal to B^2 [19, 25], where $B = N/2$ is the bisection width of a hypercube, or from Lemma 5.1 of Section 5 since a total exchange task in a hypercube requires $N/2$ steps [20]. The proposed hypercube layout has maximum wire length $N/3 + o(N)$, which is (slightly) shorter than the best previous result [10] for hypercubes of (currently) practical sizes (e.g., $N \leq 2^{14} = 16K$), and has smaller area by a factor of $2.25 + o(1)$ at the same time. Note that we can move the longer wires (and other wires belong to the same tracks) to the first 2^{n_2-1} horizontal tracks (or the first 2^{n_1-1} vertical tracks, respectively) in order to (slightly) reduce the maximum wire length.

An enhanced-cube is a hypercube with one additional outgoing link per node leading to a random node [21]. A folded hypercube [1] is a hypercube with one additional link per node, where each node S has a link connecting it to the node whose label is the bitwise complement of S. By adding additional links to the hypercube layout of Theorem 2.2, we can lay out a folded hypercube in $\frac{49}{36}N^2 + o(N^2)$ area, and an enhanced-cube in $\frac{25}{9}N^2 + o(N^2)$ area. More precisely, we first lay out an N-node hypercube in a square of side $\frac{2}{3}N + o(N)$. To lay out an additional link, we need at most an additional vertical track and an additional horizontal track, in addition to the two ending segments connecting the link to two nodes.

Since there are $N/2$ additional links in a folded hypercube, we need at most $N/2$ extra vertical and horizontal tracks to accommodate all the diameter links. Therefore, the

area for the layout of a folded hypercube is

$$\left(\frac{7}{6}N + o(N)\right) \times \left(\frac{7}{6}N + o(N)\right) = \frac{49}{36}N^2 + o(N^2).$$

Since there are N additional links in an enhanced-cube, we need at most N vertical and horizontal tracks to accommodate all the additional links. Therefore, the area for the layout of an enhanced-cube is $\frac{25}{9}N^2 + o(N^2)$.   □

The preceding layouts for folded hypercubes and enhanced-cubes improve the areas of the corresponding layouts given in [21] by constant factors.

2.2. Efficient layouts for CCC and reduced hypercubes

An n-dimensional cube-connected cycles (CCC) graph is obtained by replacing each node in an n-cube with an n-node cycle [16]. A reduced hypercube, $RH(\log_2 n, \log_2 n)$ [29], can be obtained by replacing each n-node cycle in a CCC with a $\log_2 n$-dimensional hypercube.

Theorem 2.3 *An N-node CCC or $RH(\log_2 n, \log_2 n)$ can be laid out in*

$$\frac{4N^2}{9\log_2^2 N} + o\left(\frac{N^2}{\log^2 N}\right)$$

area.

Proof: We first lay out an n-cube using the 2-D layout introduced in Subsection 2.1, and then lay out the n-node cycles within each of the hypercube nodes. Since the size of an n-dimensional CCC is $N = n2^n$ and its area is dominated by its hypercube links, which requires $2^{n+2}/9 + o(2^n)$ area, an N-node CCC can be laid out in

$$\frac{4N^2}{9\log_2^2 N} + o\left(\frac{N^2}{\log^2 N}\right)$$

area. Using the same layout method, the reduced hypercube can be laid out in asymptotically the same area.   □

In [16], layouts of area $\frac{2N^2}{\log_2^2 N} + o\left(\frac{N^2}{\log^2 N}\right)$ and $\frac{4N^2}{3\log_2^2 N} + o\left(\frac{N^2}{\log^2 N}\right)$ were proposed for the CCC graph. Our layout has area smaller than that of the layouts given in [16] by a factor of at least $3 + o(1)$, and smaller than that of the more recent layouts given in [7] by a factor of $1.125 + o(1)$. The layout area given in Theorem 2.3 is within a factor of $1.\overline{7} + o(1)$ from the lower bound given in [7] and is the best result reported thus far for the CCC network.

3. Layouts under the multilayer grid model

In this section, we introduce the *multilayer grid model* for VLSI layouts of networks. We then derive efficient multilayer layouts for hypercubes.

3.1. The multilayer grid model

In the multilayer grid model, a network is viewed as a graph whose nodes correspond to processing elements and edges correspond to wires. The nodes and edges of the graph are then embedded in a 3-D grid, where edges have unit width, can run along grid lines, but cannot cross each other (i.e., the paths for embedding these edges must be edge- and node-disjoint). The area A of a layout is defined as the area of the smallest rectangle along the x-y directions that contains all the nodes and wires. The volume of a layout is equal to the number L of layers times its area A.

In the multilayer 2-D grid model, the nodes of the graph are embedded in the 2-D grid of the first layer (i.e., $z = 1$), where a node of degree d occupies a square of side d. Note that a network with area A under the extended grid model can be laid out with area no larger than A under the multilayer 2-D grid model with $L = 2$ layers. In the multilayer 3-D grid model, the nodes of the graph are embedded in L_A layers of the 3-D grid, where a node of degree d occupies a $d/h \times d/h \times h$ cuboid and $1 \leq h \leq L_A \leq L$. These L_A layers are called "active layers." The multilayer 2-D grid model is a special case of the multilayer 3-D grid model with $L_A = 1$ active layer. Note that a $d/h \times d/h \times h$ cuboid node requires h active layers for its implementation, while a $d \times d \times 1$ cuboid node requires only 1 active layer. Layouts designed for these models can be easily modified to obtain layouts with different assumptions on the size of nodes. The cost of a layout under the multilayer grid model is a function of $A, L,$ and L_A.

The motivations for using the multilayer grid model are three folds: (1) some technologies can lay out wires using more than 2 layers, leading to significant reduction in layout area; (2) the volume of the layouts of many networks may be reduced by a factor of approximately $L/2$ compared to their layouts under the grid model; and (3) the maximum length of wires in many networks may be reduced by a factor of approximately $L/2$. When we use L layers, the number of tracks in x and y directions may both be reduced by a factor of about $L/2$ in many networks, for a factor of $L^2/4$ reduction in its area compared to the layout under the grid model, while the number of layers is only increased by a factor of $L/2$. This leads to items (1) and (2) and, therefore, the cost of the resultant layout can be significantly reduced. As a comparison, if we fold a layout derived for the grid model in order to use all the available layers, the area can be reduced by a factor of only $L/2$ and the volume cannot be reduced;

if we extended the collinear layout model to its multilayer version, the volume cannot be reduced either since the area can only be reduced by a factor of at most $L/2$ when L layers are used. The maximum wire lengths in many networks are approximately proportional to the number of tracks in x or y direction (or their sum). Therefore, if the numbers of tracks in x and y directions are both reduced by a factor of about $L/2$, the maximum wire length can also be reduced by a factor of approximately $L/2$, leading to significant improvement in performance [item (3)]. As a comparison, the maximum wire length in a collinear layout using L layers or in a layout obtained by folding the layout derived using the grid model remains similar in most cases. These arguments will become clear by looking at the multilayer layouts derived in the following subsections.

We can extend the multilayer grid model to the *multilayer layout model* by allowing nodes and edges to run in other specified directions. Layouts under this model may have smaller area and volume compared with layouts under its multilayer grid model counterpart. Moreover, wires in this model may have different width and cross area, depending on the technology used. For example, wires along the z direction may have larger cross area in PCB. In what follows, we focus on the multilayer 2-D grid model. Layouts under other models will be reported in the near future.

3.2. The layout area and volume of hypercubes under the multilayer grid model

In this subsection we present efficient multilayer layouts for hypercubes.

We first derive hypercube layouts with an even number L of layers. The multilayer 2-D grid layout of a hypercube can be obtained from its 2-D grid layout by partitioning all the horizontal (resp., vertical) tracks above each row (resp., to the right of each column) of nodes into $L/2$ groups, each of which has at most $k_1 = \lceil \frac{2\lfloor 2^{n_2+1}/3 \rfloor}{L} \rceil$ (or $\lceil \frac{2\lfloor 2^{n_2+1}/3 \rfloor - 1}{L} \rceil$ if n_2 is odd) horizontal tracks [resp., $k_2 = \lceil \frac{2\lfloor 2^{n_1+1}/3 \rfloor}{L} \rceil$ (or $\lceil \frac{2\lfloor 2^{n_1+1}/3 \rfloor - 1}{L} \rceil$ if n_1 is odd) vertical tracks] and is wired using two layers. More precisely, the vertical segments connecting the horizontal tracks of groups i (above each row) and the vertical tracks of groups i (to the right of each column) are wired using layer $2i - 1$, and the horizontal tracks of groups i and the horizontal segments connecting the vertical tracks of groups i are wired using layer $2i$, for $i = 1, 2, \ldots, L/2$. When a link makes a turn in the 2-D grid layout, its vertical and horizontal segments, wired in neighboring layers $i - 1$ and i in the multilayer layout, should be connected by a wire (or via) along the z direction.

When $L = o(\sqrt{N}/\log N)$, the area of the resultant L-layer layout can be reduced from $4N^2/9 + o(N^2)$ under the grid

model to

$$\frac{16N^2}{9L^2} + o\left(\frac{N^2}{L^2}\right);$$

the maximum wire length of the L-layer layout can be reduced from $N/3 + o(N)$ to

$$\frac{2N}{3L} + o\left(\frac{N}{L}\right);$$

the total wire length of a routing path is $1.\overline{3}N/L + o(N/L)$; and the volume of the L-layer layout can be reduced from $8N^2/9 + o(N^2)$ (assuming wires cannot cross each other) to

$$\frac{16N^2}{9L} + o\left(\frac{N^2}{L}\right).$$

When L is odd, we simply partition horizontal tracks into $(L+1)/2$ groups, wired on layers $1,3,\ldots,L$, and partition vertical tracks into $(L-1)/2$ groups, wired on layers $2,4,\ldots,L-1$. We can also partition and wire them the other way around. The area of the resultant layout is

$$\frac{16N^2}{9(L^2-1)} + o\left(\frac{N^2}{L^2}\right)$$

when L is odd and $L = o(\sqrt{N}/\log N)$.

These multilayer hypercube layouts are the best results reported in the literature thus far for $L = 2,3,\ldots,$ $o(\sqrt{N}/\log N)$ in terms of area and volume. Since we have obtained area-efficient L-layer layouts for hypercubes, $L = 2,3,\ldots,o(\sqrt{N}/\log N)$, we can optimize the cost for implementation by minimizing the cost function $f(A,L,L_A = 1)$.

If a large number $L = \Omega(\sqrt{N}/\log N)$ of layers are available and more than one active layer is available, we can design hypercube layouts under the multilayer 3-D grid model to further reduce the layout area, maximum wire length, and volume. To obtain multilayer 3-D layouts for an $(n_1 + n_2 + n_3)$-cube, we simply use 2^{n_3} copies of a multilayer 2-D $(n_1 + n_2)$-cube layout, and connect nodes belonging to the same grid point in a way similar to a collinear layout of an n_3-cube. More details will be reported in the near future.

4. VLSI layouts for several networks

In this section, we present efficient layouts for several interconnection networks under the grid model.

4.1. Efficient layouts for generalized hypercubes, HSNs and RHSNs

An l-level hierarchical swapped network, denoted by HSN(l,G) [23, 25], is an l-level network consisting of M level-l clusters, each of which is an HSN($l-1,G$) network,

where M is the number of nodes in the nucleus G. Each of the M^{l-2} nuclei of a level-l cluster has a link connecting it to each of the other $M-1$ level-l clusters. If we view a level-l cluster as a supernode, the HSN(l,G) becomes an M-supernode complete graph with N/M^2 edges between each pair of supernodes.

Theorem 4.1 *An N-node HSN(l,G) can be laid out using $N^2/16 + o(N^2)$ area if*

(a) $l = 2$ *and the nucleus G can be laid out in a square of side $o(M^{\frac{3}{2}})$, or*

(b) $l = 3$ *and the nucleus G can be laid out in a square of side $o(M^2)$, or*

(c) $l \geq 4$,

assuming that M, the size of the nucleus G, is not a constant.

Proof: The inter-cluster links between top-level clusters can be laid out in $N^2/16 + o(N^2)$ area using the layout of an M-node complete graph [25, 27] with multiple edges. When one of the conditions holds, the area for all nuclei does not affect the leading constant of the layout area and the required area is dominated by the top-level inter-cluster links. □

An r-deep recursive hierarchical swapped network (abbreviated RHSN) [23, 25] is obtained by recursively replacing the nucleus of an HSN with an $(r-1)$-deep RHSN. More precisely, RHSN($l_r,l_{r-1},\ldots,l_1,G$) = HSN(l_r, RHSN(l_{r-1}, $l_{r-2},\ldots,l_1,G$)). Therefore, RHSN can be laid out by recursively laying out HSNs.

Theorem 4.2 *An N-node RHSN($l_r,l_{r-1},\ldots,l_1,G$) can be laid out using $N^2/16 + o(N^2)$ area, assuming that the depth r is at least 2 and the number of nodes in an RHSN($l_{r-1},l_{r-2},\ldots,l_1,G$) is not a constant. (In other words, at least one of the parameters r,M, and l_i for any $i \leq r-1$ is not constant, where M is the size of the nucleus G.)*

By shrinking all the nuclei of an HSN into a node, we obtain a radix-M generalized hypercube [4, 11]. This combined with Theorem 4.1 leads to the following theorem for the layout of high-radix hypercubes.

Theorem 4.3 *A radix-M generalized hypercube can be laid out using $M^2N^2/16 + o(M^2N^2)$ area, assuming that M is not a constant.*

The above layout can be easily extended to general mixed-radix generalized hypercubes [4]. As will be shown in Section 5, the proposed layouts for generalized hypercubes and HSNs are optimal within a factor of $1 + o(1)$.

4.2. Optimal layouts for butterfly networks and indirect swapped networks (ISNs)

In this subsection we present efficient layouts for butterfly networks and indirect swapped networks (ISNs) [22]). A butterfly network [13] is obtained by unfolding the structure of a hypercube along routing paths, while an indirect swapped network (ISN) (also called an unfolded swapped network (USN) [22]) is a multistage network obtained by unfolding the structure of a swapped network [23, 25]. We first present optimal layouts for ISNs and then use them to derive optimal layouts for butterfly networks.

Theorem 4.4 *An N-node ISN can be laid out in*

$$\frac{N^2}{4\log_2^2 N} + o\left(\frac{N^2}{\log^2 N}\right)$$

area, assuming that the number M_l of top-level clusters in the corresponding swapped network (which is unfolded to generate the ISN) is not a constant.

Proof: If we place every M_l rows of the ISN into the same top-level block [25, 28], then each pair of the blocks are connected by 2 links. Therefore, we can lay out the inter-cluster links using the layout of an $\left(\frac{N}{M_l\log_2 N} + o\left(\frac{N}{M_l\log N}\right)\right)$-node complete graph with multiple edges, which requires

$$\frac{N^2}{4\log_2^2 N} + o\left(\frac{N^2}{\log^2 N}\right)$$

area [25, 27]. □

The layout area for the ISN improves the corresponding result given in [22] by a factor of $4 + o(1)$.

The following theorem gives a layout for the butterfly network that is optimal within a factor of $1 + o(1)$ from the lower bound given in [2].

Theorem 4.5 *An N-node butterfly network can be laid out in*

$$\frac{N^2}{\log_2^2 N} + o\left(\frac{N^2}{\log^2 N}\right)$$

area.

Proof: If we unfold an HSN(2, $\frac{\log_2 N}{2}$-cube), we obtain a $(\log_2 N + 2)$-stage ISN that uses $\frac{\log_2 N}{2}$-dimensional butterfly networks as the basic modules. If we double up the links connecting the middle two stages of the ISN, remove nodes in the $\left(\frac{\log_2 N}{2} + 2\right)$-th stage, and reconnect each of the replicated links to one of the two links between the $\left(\frac{\log_2 N}{2} + 2\right)$-th and the $\left(\frac{\log_2 N}{2} + 3\right)$-th stage through a removed node, we can obtain an automorphism of a $(\log_2 N)$-dimensional butterfly
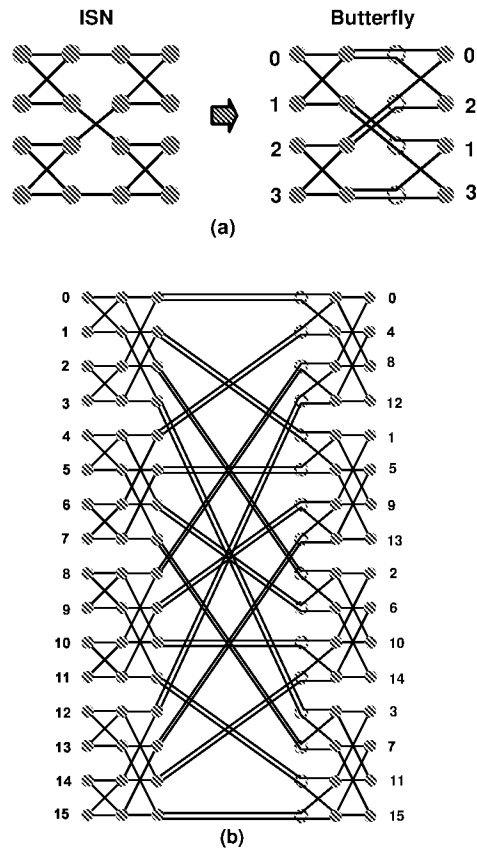
Figure 2. Deriving butterfly networks from indirect swapped networks. (a) Transforming a 4×4 ISN into a 4×4 butterfly network. (b) A resultant 16×16 butterfly network.

(see Fig. 2). Therefore, the area of the butterfly is approximately 4 times that of an ISN; that is

$$\frac{N^2}{\log_2^2 N} + o\left(\frac{N^2}{\log^2 N}\right).$$

□

In [2] the same upper bound for the area of a butterfly network has been presented. The proof given in [2] is, however, considerably more complicated than our construction. It is also interesting that butterfly networks can be laid out based on the layout of a complete graph.

Generalized hypercubes, HSN(l, Q_n) with $l > 2$, and ISNs unfolded from such HSNs can be laid out based on the collinear layouts of complete graphs rather than the 2-D layouts of complete graphs. Similar to the multilayer layouts of hypercubes, the resultant layouts for these networks can be easily partitioned and wired using $L > 2$ layers. The resultant maximum wire lengths in these layouts can be reduced

by a factor of approximately 2 compared to the layouts presented in this paper and the areas are also slightly reduced. Similar to Theorem 4.5 (see Fig. 2), multilayer layouts for butterfly networks can be obtained by modifying the multilayer layouts for ISNs. More details will be reported in the near future.

5. Tight bounds on the VLSI areas of several networks

In this section, we derive tight bounds on the areas of several networks under the grid model.

The total exchange (TE) task [3, 9] (also called all-to-all personalized communication) is a basic communication task that arises often in applications, where each node has to send a different (personalized) packet to every other node of the network. In [25], we have shown the following lemma concerning the relationship between the VLSI area of a network and the throughput for performing TE tasks in it.

Lemma 5.1 *Assume that $f(N)$ total exchange (TE) tasks can be executed in $f(N)T_{TE}$ communication steps in an N-node interconnection network for some integer function $f(N)$, under the all-port communication model. Then the layout area of the network is at least equal to*

$$\frac{\lfloor N/2 \rfloor^2 \times \lceil N/2 \rceil^2}{T_{TE}^2} \approx \frac{N^4}{16 T_{TE}^2}.$$

When performing the TE tasks we assume that links are bidirectional and nodes can have infinitely large routing tables and buffer space and perform infinitely many computation steps if required. (Links are still assumed to have unit width in the layout.)

In what follows we show that several of the layouts presented in Section 4 have areas that are optimal within a factor of $1 + o(1)$.

Theorem 5.2 *The area of the minimal layout of a radix-M generalized hypercube is equal to $M^2N^2/16 \pm o(M^2N^2)$, assuming that M is not a constant, where N is the number of nodes in the network.*

Proof: The upper bound is given in Theorem 4.3. A lower bound on its VLSI area is given by

$$\frac{(M-1)^2 n^2 N^2}{16 n^2} = \frac{M^2 N^2}{16} - o(M^2 N^2)$$

from Lemma 5.1 and the fact that n TE tasks can be performed in nN/M steps in an n-dimensional radix-M hypercube. □

The throughput for performing TE tasks in HSNs is given in the following lemma [25].

Lemma 5.3 *The throughput for executing TE tasks in an N-node $HSN(l, G)$ can be arbitrarily close to $1/N$, provided that the nucleus G can execute l TE tasks in M time steps under the all-port communication model, where M is the number of nodes in G.*

By combining Lemma 5.3 with Theorems 4.1 and 5.1, we can prove that the layout for HSNs is also close to being strictly optimal.

Theorem 5.4 *The area of the minimal layout of an N-node $HSN(l, G)$ is equal to $N^2/16 + o(N^2)$ if the nucleus G can execute l TE tasks in M time steps under the all-port communication model and*

(a) *$l = 2$ and the nucleus G can be laid out in a square of side $o(M^{\frac{3}{2}})$, or*

(b) *$l = 3$ and the nucleus G can be laid out in a square of side $o(M^2)$, or*

(c) *$l \geq 4$,*

assuming that M, the size of a nucleus G, is not a constant.

In Section 4 we derived optimal layouts for butterfly networks based on the layouts of ISNs. In what follows, we show that the lower bound on the VLSI area of a butterfly network given in [2] can be used to derive a lower bound on the area of an ISN.

Theorem 5.5 *The area of the minimal layout of an N-node ISN is equal to $\frac{N^2}{4 \log_2^2 N} \pm o\left(\frac{N^2}{\log^2 N}\right)$, assuming the nucleus of the ISN is a butterfly network.*

Proof: From Theorem 4.5, we can see that if it were possible to lay out an ISN in $\frac{(1-\varepsilon)N^2}{4 \log_2^2 N}$ area, then it would also be possible to lay out a butterfly network in $\frac{(1-\varepsilon)N^2}{\log_2^2 N} + o\left(\frac{N^2}{\log^2 N}\right)$ area, where ε is a positive constant. This contradicts the lower bound given in [2]. Therefore, the area of an ISN is at least $\frac{N^2}{4 \log_2^2 N} - o\left(\frac{N^2}{\log^2 N}\right)$. The upper bound is given in Theorem 4.4. □

Theorem 5.5 can be generalized to ISNs that are based on other nuclei that contain a butterfly network of the same size as a subgraph.

By using the techniques introduced, we can also obtain tight bounds on the bisection widths of the networks investigated in this paper and efficient layouts for many other networks, such as macro-star networks [26] periodically regular chordal rings [14, 15], and cyclic networks [24]. Some examples can be found in [25, 28].

6. Conclusion

We derived layouts for butterfly networks, generalized hypercubes, HSNs, and ISNs that are optimal within a factor of $1 + o(1)$ under the grid model. We presented efficient layouts for hypercubes, CCCs, folded hypercubes, reduced hypercubes, RHSNs, and enhanced-cubes, which are the best results reported thus far under the grid model. In particular, the number of tracks of the collinear layout of hypercubes is optimal within a factor of $1.\bar{3}$, and the areas of proposed 2-D layouts for hypercubes and CCC are optimal within a factor of $1.\bar{7}$ under the grid model. We also derived efficient multilayer layouts for hypercubes, which are the best results reported thus far for the given numbers of layers. The techniques used in this paper can be used to obtain efficient layouts for a wide variety of other interconnection networks [25, 28].

References

[1] Adams, G.B. and H.G. Siegel, "The extra stage cube: a fault-tolerant interconnection network for supersystems," *IEEE Trans. Comput.,* vol. 31, no. 5, May. 1982, pp. 443-454.

[2] Avior, A., T. Calamoneri, S. Even, A. Litman, and A. Rosenberg, "A tight layout of the butterfly network," *Proc. ACM Symp. Parallel Algorithms and Architectures,* Jun. 1996, pp. 170-175.

[3] Bertsekas, D.P. and J. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods,* Athena Scientific, 1997.

[4] Bhuyan L.N. and D.P. Agrawal, "Generalized hypercube and hyperbus structures for a computer network," *IEEE Trans. Comput.,* vol. 33, no. 4, Apr. 1984, pp. 323-333.

[5] Brebner, G., "Relating routing graphs and two-dimensional grids," *VLSI: Algorithms and Architectures,* 1985, pp. 221-231.

[6] Chen, C. and D.P. Agrawal, "dBCube: a new class of hierarchical multiprocessor interconnection networks with area efficient layout," *IEEE Trans. Parallel Distrib. Sys.,* vol. 4, no. 12, Dec. 1993, pp. 1332-1344.

[7] Chen G. and F.C.M. Lau, "A compact layout of cube-connected cycles," *Proc. Int'l Conf. High Performance Computing,* Dec. 1997, 422-427.

[8] Fernández, A. and K. Efe, "Efficient VLSI layouts for homogeneous product networks," *IEEE Trans. Computer,* vol. 46, no. 10, Oct. 1997, pp. 1070-1082.

[9] Johnsson, S.L. and C.-T. Ho, "Optimum broadcasting and personalized communication in hypercubes," *IEEE Trans. Computers,* vol. 38, no. 9, Sep. 1989, pp. 1249-1268.

[10] Lai, T.-H. and A.P. Sprague, "Placement of the processors of a hypercube," *IEEE Trans. Computers,* vol. 40, no. 6, Jun. 1991, pp. 714-722.

[11] Lakshmivarahan, S. and S.K. Dhall, "A new hierarchy of hypercube interconnection schemes for parallel computers," *J. Supercomputing,* vol. 2, 1988, pp. 81-108.

[12] Leighton, F.T., *Complexity Issues in VLSI : Optimal Layouts for the Shuffle-exchange Graph and Other Networks* Cambridge, Mass., MIT Press, 1983.

[13] Leighton, F.T., *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes,* Morgan-Kaufman, San Mateo, CA, 1992.

[14] Parhami B., *Introduction to Parallel Processing: Algorithms and Architectures,* Plenum Publishing Corp., 1999.

[15] Parhami B. and D.-M. Kwai, "Periodically regular chordal rings," *IEEE Trans. Parallel Distrib. Sys.,* to appear.

[16] Preparata, F.P. and J.E. Vuillemin, "The cube-connected cycles: a versatile network for parallel computation," *Commun. ACM,* vol. 24, No. 5, pp. 300-309, May 1981.

[17] Sýkora and I. Vrt'o, "On VLSI layouts of the star graph and related networks," *Integration, VLSI J.* 1994, pp. 83-93.

[18] Thompson, C.D., "Area-time complexity for VLSI," *Proc. ACM Symp. Theory of Computing,* 1979, pp. 81-88.

[19] Thompson, C.D., "A complexity theory for VLSI," Ph.D. dissertation, Dept. of Computer Science, Carnegie-Mellon Univ., Pittsburgh, PA, 1980.

[20] Varvarigos, E.A. and D.P. Bertsekas, "Communication algorithms for isotropic tasks in hypercubes and wraparound meshes," *Parallel Computing,* vol. 18, no. 11, Nov. 1992, pp. 1233-1257.

[21] Varvarigos, E.A., "Static and dynamic communication in parallel computing," Ph.D. dissertation, Dept. Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1992.

[22] Yeh, C.-H. and B. Parhami, "A class of parallel architectures for fast Fourier transform," *Proc. Midwest Symp. Circuits and Systems,* Aug. 1996, pp. 856-859.

[23] Yeh, C.-H. and B. Parhami, "Recursive hierarchical swapped networks: versatile interconnection architectures for highly parallel systems," *Proc. IEEE Symp. Parallel and Distributed Processing,* Oct. 1996, pp. 453-460.

[24] Yeh, C.-H. and B. Parhami, "Cyclic networks – a family of versatile fixed-degree interconnection architectures," *Proc. Int'l Parallel Processing Symp.,* Apr. 1997, pp. 739-743.

[25] Yeh, C.-H., "Efficient low-degree interconnection networks for parallel processing: topologies, algorithms, VLSI layouts, and fault tolerance," Ph.D. dissertation, Dept. Electrical & Computer Engineering, Univ. of California, Santa Barbara, Mar. 1998.

[26] Yeh, C.-H. and E.A. Varvarigos, "Macro-star networks: efficient low-degree alternatives to star graphs," *IEEE Trans. Parallel Distrib. Sys.,* Vol. 9, no. 10, Oct. 1998, pp. 987-1003.

[27] Yeh, C.-H. and B. Parhami, "VLSI layouts of complete graphs and star graphs," *Information Processing Letters,* Vol. 68, Oct. 1998, pp. 39-45.

[28] Yeh, C.-H., B. Parhami, and E.A. Varvarigos, "The recursive grid layout scheme for VLSI layout of hierarchical networks," *Proc. Merged Int'l Parallel Processing Symp. & Symp. Parallel and Distributed Processing,* Apr. 1999, to appear.

[29] Ziavras, S.G., "RH: a versatile family of reduced hypercube interconnection networks," *IEEE Trans. Parallel Distrib. Sys.,* vol. 5, no. 11, Nov. 1994, pp. 1210-1220.

HARP: Hard-wired Routing Pattern FPGAs

Satish Sivaswamy†, Gang Wang‡, Cristinel Ababei†,
Kia Bazargan†, Ryan Kastner‡ and Eli Bozorgzadeh††

†ECE Dept.	‡Dept. of ECE	††Computer Science Dept.
Univ. of Minnesota	Univ. of California, Santa Barbara	Univ. of California, Irvine
Minneapolis, MN 55455	Santa Barbara, CA 93106	Irvine, CA 92697
satish,ababei,kia@ece.umn.edu	wanggang,kastner@ece.ucsb.edu	eli@igor.ics.uci.edu

ABSTRACT

Modern FPGA architectures provide ample routing resources so that designs can be routed successfully. The routing architecture is designed to handle versatile connection configurations. However, providing such great flexibility comes at a high cost in terms of area, delay and power. We propose a new FPGA routing architecture[1] that utilizes a mixture of hardwired and traditional flexible switches. The result is 24% reduction in leakage power consumption, 7% smaller area and 24% shorter delays, which translates to 30% increase in clock frequency. Despite the increase in clock speeds, the overall power consumption is reduced by 8%.

Categories and Subject Descriptors

B.7.1 [**Integrated Circuits**]: Types and Design Style—*Gate arrays*; B.7.2 [**Integrated Circuits**]: Design Aids—*Routing*

General Terms

Design, Performance, Experimentation

1. INTRODUCTION

Prohibitive ASIC mask costs and stringent time-to-market windows have made FPGAs an attractive implementation platforms in recent years. However, circuits implemented on FPGAs are typically slower, occupy more area, and consume more power than ASIC circuits [25]. The FPGA routing architecture is the main culprit in making FPGAs worse than ASIC chips in area, delay and power; a typical FPGA routing architecture uses about 70-90% of the total transistors on the die [6].

A significant body of work from the past two decades focused on switch box design and segmented routing architectures. The basic idea is to use highly flexible switches where horizontal and vertical tracks meet, to facilitate all possible connections between the adjacent tracks. A sketch of the disjoint switch box is shown in Figure 1.

Assuming all tracks have unit length, the disjoint switch box (see Figure 1) can route a large subset of possible routing trees to connect the terminals of a net. As a result, the overall channel width will not be high. However, flexibility in routing

---

[1]This work was supported in part by a grant from NSF under contract CAREER CCF-0347891
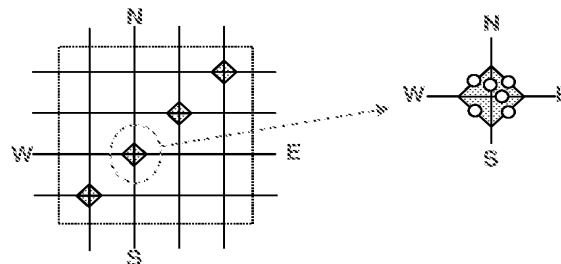
Figure 1: SRAM-based Switch Box.

comes with great performance costs. Building a routing tree from many segments that are connected by switches has the following disadvantages:

- **Circuit Delay:** The delay of a net is mainly dependent on the number of programmable switches in its routing path [14, 4]. Hence, a large number of programmable switches contributes greatly to the overall circuit delay.

- **Area:** By increasing the number of programmable pass transistors (which correspond to the small circles in the switch on the right in Figure 1) inside each switch, we pay an area penalty as each of the programmable pass transistors requires an SRAM cell for programming it and possibly buffers to improve signal slew.

- **Leakage Power:** Leakage power is becoming a major component of the total power consumption [1] and the majority of the leakage power consumption in FPGAs occur in the routing switches [7].

In this work, we extend the idea of eliminating switches to two dimensions; instead of just hardwiring two horizontal or two vertical segments to form longer wires, e.g. segmented routing architectures such as Xilinx Virtex, we form hardwired junctions between horizontal and vertical segments inside switch boxes. These junctions create routing segments in the shape of T's, L's and +'s and their rotated versions. An example of such a switch box is shown in Figure 13. As a result of hardwiring connections, we eliminate some programmable switches, which decreases the delay, area and power dissipation.

However, we must be careful that the reduction in programmable switches does not severely affect the routing flexibility. The distribution of the hard-wired routing patterns (HARPs) are obtained after a careful analysis of the routing profiles of different circuits. Our technique maintains the programmability of FPGAs, while improving their performance metrics. We place and route circuits with these patterns embedded in the switch boxes and report results of area, delay, power and channel width.

The paper is organized as follows. In Section 2, we describe the terminology that we use throughout the paper and the overall flow of our architectural design. In Section 3, we perform an empirical analysis on detailed routings to find the most common routing patterns and their densities. Based on this analysis, we design the architecture of the switch boxes containing the hardwired routing patterns. Details of this step are discussed in Section 3.3. Section 4 explains how hard-wired routing patterns inside the switch boxes are exploited by the router. Experimental results are presented in Section 5. Section 6 gives details on related work. We conclude in Section 7, by outlining our main contribution and discussing future research directions.

2. PRELIMINARIES

2.1 Basic Terminology

The routing of a multi-terminal net is frequently modeled as a *rectilinear Steiner tree (RST)*. A RST has three types of *joint patterns*: L-shape, T-shape, and +-shape. An FPGA routing architecture with uniform unit-length segmentation has a switch at each of the joint patterns. Additionally, there are switches for horizontal (H) and vertical (V) routes that span more than one channel. Modern FPGA devices use multi-length segments (—-shape and |-shape) in order to reduce the number of switches along the horizontal or vertical routes of the nets. This enhances the delay of the routing; however, it reduces the flexibility of the architecture.

We call the switch shown on the right side of Figure 1 a *flexible* switch. A multi-length segmented architecture merges the "W" track and the "E" track to form a longer segment. This is equivalent to removing the pass transistors (and their associated SRAM cells and buffers) that connect the "E" or "W" tracks to other tracks. The result is a hardwired connection between "E" and "W". The area of this new switch is smaller, however, it is less flexible than the original *flexible* switch. If we allow the horizontal track to also connect to the vertical track at this junction (*e.g.*, the way hex lines in Xilinx architectures connect to other segments on the middle point), then two more switches will be used to provide connectivity between wire segments "WE" and "N", and also between "WE" and "S". Obviously, the area and delay of this switch increases, but we gain flexibility.

To the best of our knowledge, no one has extended the idea of hardwiring pass transistors to junctions that are formed between horizontal and vertical tracks. In this work, we study *HARP* (HArd-wired Routing Pattern) architectures that utilize hardwired "switches" at certain junction patterns. The three joint patterns (L, T, +, and H/V) and their various orientations result in eleven possible HARPs: ⊤,⊥, ⊣, ⊢, ⌐, ⌐, ∟, ⌐, |,— and +.

2.2 HARP-based FPGA Routing Architecture Design Flow

Figure 2 shows our design flow to introduce HARPs into *traditional* FPGA routing architectures. First, we place and route a number of circuits on a traditional FPGA architecture. By analyzing the routes of the circuits, we extract the frequency at which different HARP patterns are used in switch boxes. Next, we use the results of the pattern distribution analysis to create a new architecture that has a mixture of flexible and HARP switches. Finally, we place and route designs on the new HARP architecture and compare the results with the traditional architectures.
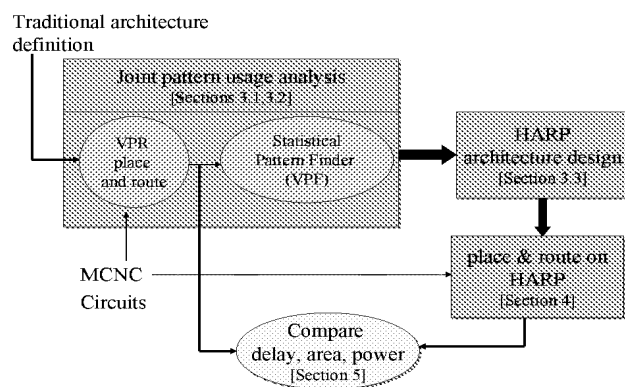
Figure 2: HARP-based Routing Architecture Design Flow.

3. ROUTING PATTERN ANALYSIS

In this section, we discuss the statistical analysis of routing pattern frequencies and correlations between circuits, architectures and these patterns.

3.1 Testing Benchmark and Routing Result Generation

After placing and routing the MCNC benchmark circuits, we observed how often each of the joint patterns can be found in a route of each net. Note that the placement and routing algorithms will affect the frequency of these patterns. We will discuss this issue in Section 7.

Statistical information can guide us on how often we need to insert HARP switches inside switch boxes. To find the pattern frequencies, we routed all the benchmarks on a given traditional segmented FPGA routing architecture applying the VPR FPGA place-and-route tool [3]. For a given routing segmentation, we routed each circuit, detected the patterns in the route files, and applied statistical analysis on the data.

In our studies, we considered two segmentation architectures: unit-length segmentation and multi-length segmentation (similar to Xilinx's Virtex family). We used the VPR router in three different modes: Timing-driven, Routability-driven without bend-cost, and Routability-driven with bend-cost. We experimented our technique on the MCNC benchmark suite [23].

3.2 Analysis of Routing Patterns

3.2.1 VPR Pattern Finder Tool

In order to analyze the behavior of the routing patterns, we have implemented *VPR Pattern Finder*(VPF), a graphic tool for parsing, visualizing and analyzing the VPR routing results [19]. VPF takes a VPR routing result file as input and automatically extracts the routing information, identifies the connection patterns in switch boxes, and in turn generates statistical reports for different patterns.

Based on the underlying FPGA architectures, there are two types of VPR routing file formats: *unit-length* or *multi-length*. In the unit-length architecture, each routing segment has a length of one, while in the latter approach, segments span multiple configurable logical block (CLBs), and are staggered to provide faster and more direct connectivity. A multi-length segment is defined by its starting and ending coordinates. In VPF we provide a uniform model to handle both formats, where the unit-length results are treated in the same way for multi-length segments, except the starting coordinates and the ending coordinates are identical.

(a) switch box indexing   (b) pattern labelling
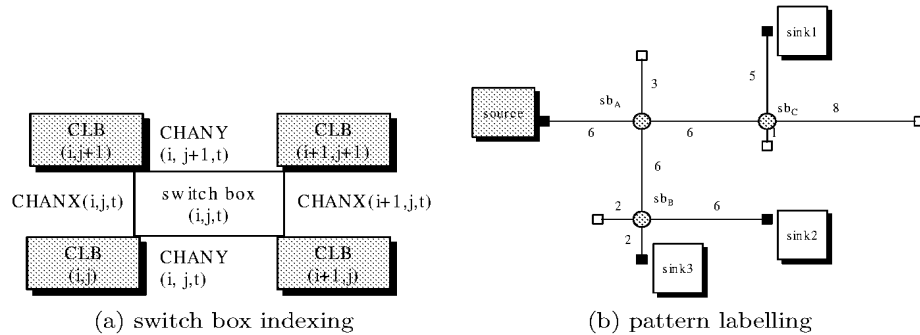
Figure 3: Switch box indexing and pattern labelling in VPF

(a) results for different architectures (both timing driven)

(b) results for different routing considerations on the same multi-length architecture (Virtex-like)
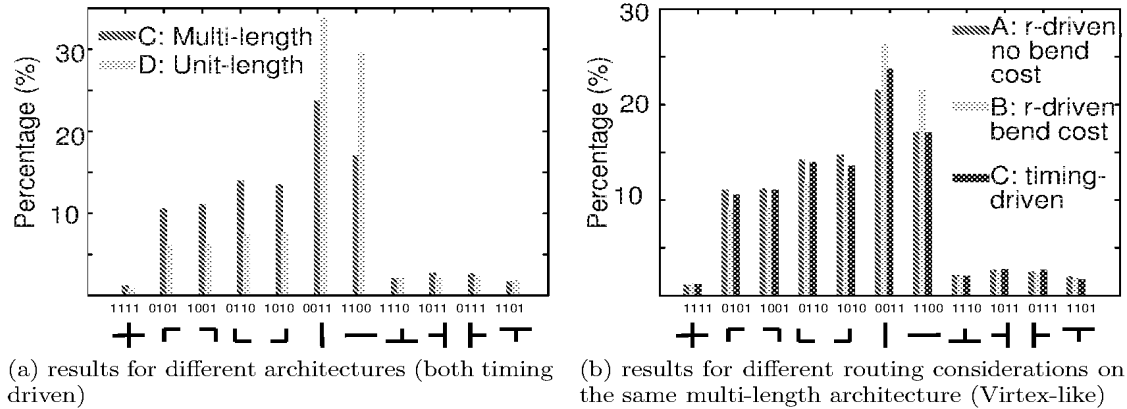
Figure 4: Switch box pattern distributions
(Benchmarks A, B, and C are generated using the same multi-length Virtex style architecture
with different routing considerations, i.e. routability-driven without bend-cost, routability-driven with bend-cost,
and timing-driven; Benchmark D is the timing-driven result using the unit-length architecture)

In our analysis, we focus on the connection patterns of the switch boxes. For a given FPGA layout, a switch box is indexed by a tuple (i, j, t) as shown in Figure 3(a), where i and j indicate the physical location of the switch box and t identifies the track being used. Based on the structure of the switch box, we have 11 possible connection patterns on a switch box. They are ⊤, ⊥, ⊣, ⊢, ⌐, ⌐, ∟, ⌐, |, − and +. We encode these patterns with four binary bits that denote whether the *left, right, top* and *bottom* track associated with the switch box is connected to the junction. As an example, Figure 3(b) shows a sample net, which contains only one source and three sinks. Empty rectangles show ends of the segments that are not connected to this net. The numbers on the lines denote the length of the connections (that are possibly formed by connecting two or more segments). Based on the above discussion, switch box sb_A has pattern ⊤(1101), sb_B has pattern ⊢(0111), and sb_C is of pattern ∟(1010).

It is important to find out how each HARP extends along different directions - the *pattern length*. This information can provide more insight into the routing behavior and offer useful guidance for improving routing quality by hard-wiring these patterns. VPF performs this analysis using the following simple algorithm: (i) For each marked switch box, identify its pattern. Based on the pattern information, try to trace along the valid directions starting from the switch box; (ii) Stop when we meet a switch box that has a pattern other than | or −, or we reach

the source or a sink; (iii) Report this distance as the result of the current direction; (iv) Take the minimum among all directions as the pattern length of the current switch box. Following the same example shown in Figure 3(b) and assuming the numbers by the segments indicate the segment lengths, switch boxes sb_A, sb_B and sb_C have pattern lengths 6, 2 and 5 respectively.

3.2.2 Statistical Results and Analysis

Statistical information about the switch box patterns obtained from VPF provide insight into the behavior of the placement and the routing tools as well as resource demands of the circuits. Figure 4 shows the normalized pattern distributions (in percentage of all the switch boxes) for different benchmarks and segmented architectures. Among them, the unit-length (D) results are generated on an architecture that only supports segment of length one, while those of A, B and C are generated on a Virtex style architecture with multi-length segment routing architecture. A is generated with the routability-driven routing without bend cost, B corresponds to routability-driven with bend cost, and C is generated using the timing-driven routing mode. Placement for all experiments was done using the timing-driven mode.

One interesting observation that can be made from Figure 4(a) is that the multi-length segmented architecture greatly changes the pattern distribution compared to unit length. The combined frequency of vertical and horizontal patterns drops from

(a) Length distribution for pattern | and −

(b) Length distribution for T patterns

(c) Length distribution for L patterns

(d) Length distribution for pattern +
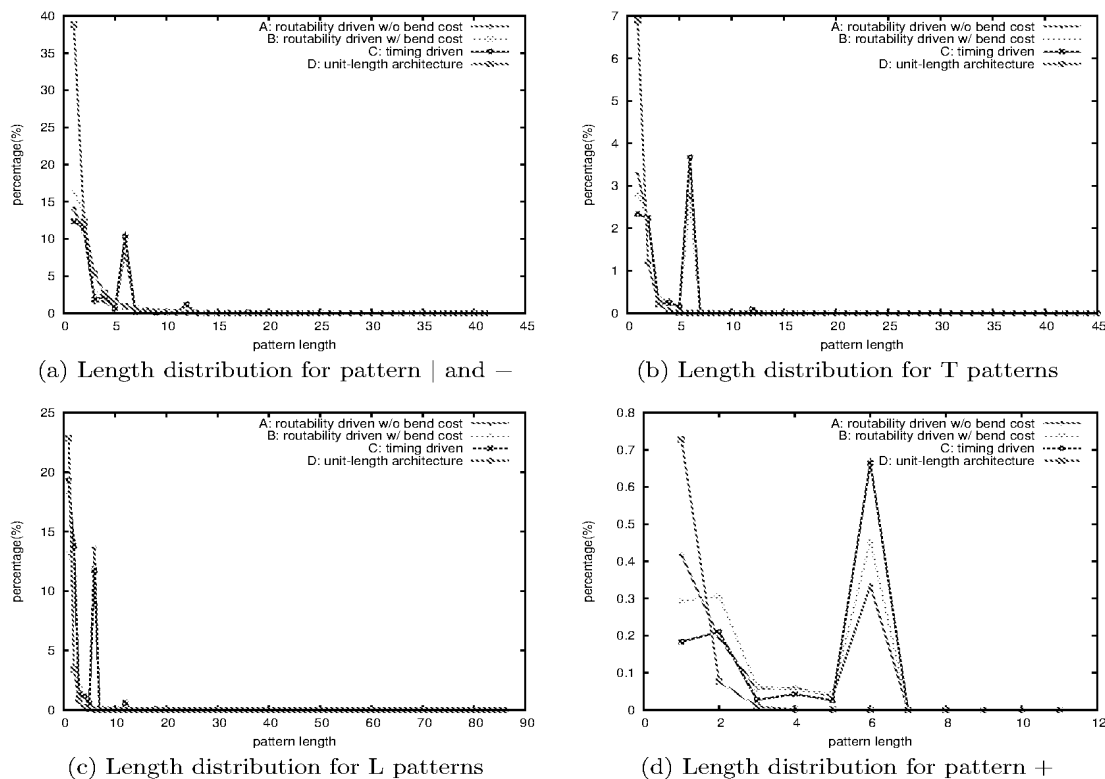
Figure 5: Distribution statistics on switch box pattern lengths
(Benchmarks A, B, and C are generated using the same multi-length Virtex style architecture
with different routing considerations, i.e. routability-driven without bend-cost, routability-driven with bend-cost,
and timing-driven; Benchmark D is the timing-driven result using the unit-length architecture)

63.3% to 41.2%. On the other hand, this drastic change is not seen in Figure 4(b) where the results are obtained on the same Virtex style architecture but with different router settings. In other words, the architecture seems to have a much bigger impact on the switch box pattern distribution than the routing algorithm. Figure 4(a) shows there is little change in the percentage of the T patterns (pattern 1110, 1011, 0111, and 1101) or the + pattern (pattern 1111) when we switch from unit-length to the multi-length segment architecture. On the contrary, there is a significant increase for the L patterns (pattern 0101, 1001, 0110 and 1010). The combined frequency of all the L patterns increases from 27.46% for the unit-length architecture to 41.62% for the multi-length architecture. In other words, for the multi-length architecture, the possibility of having an L patterned switch box is comparable to (if not more than) that of a vertical or horizontal pattern. This is a profound difference when compared with the unit-length results, in which the vertical and horizontal patterns are overwhelmingly dominating the pattern distribution.

Next, we analyze the length of the patterns using the method discussed in Section 3.2.1. Figure 5 illustrates the pattern length distributions for our testing benchmarks. The x-axis in these graphs is the pattern length, while the y-axis is the normalized percentage for switch box patterns with the given length.

We can observe that all these graphs share some common characteristics. First, for the unit-length architecture, the pattern length distribution drops rapidly and monotonically as the

length increases. The results for the multi-length architecture are more sophisticated but still follow a similar trend except for length 6, which shows spikes on all patterns. On all patterns except +, there is a small spike at length 12 too. Such harmonic behavior demonstrated by these spikes is no surprising because in the multi-length architecture, the majority of the segments are of length 6, where switch connections are allowed at both the middle point and the ends of the segments.

We also performed further analysis (results not shown in this paper), focusing on the geometric distribution of different patterns. We observed uniform distribution of all patterns with the exception of one benchmark[2].

3.3   Architecture Design

Architecture design for FPGAs is a complex problem and much work has been done in this area since FPGAs were first proposed. There are many architectural factors (such as switch-block or switch-matrix style, switch-block flexibility F_s, connection-block flexibility F_c, frequency of switch-blocks along routing segments, channel segmentation and staggering, clustering of LUTs in CLBs), which contribute to the quality of the final FPGA platform. Among these factors, switch-block design is of paramount importance. Switch box design has been addressed in previous works.

Previously proposed switch boxes include (see Figure 6.): (i) The Xilinx XC4000-series [21] (also known as *disjoint* or *sub-*

---

[2]For circuit "bigkey", the + patterns were concentrated in the two horizontal channels at the center of the chip.

set). It is area-effective but creates disjoint routing domains. (ii) *Anti-symmetric* switch box [17] is known for good practical routability. (iii) *Universal* switch box [5] can simultaneously route all two-point connections in the switch-block. (iv) *Wilton* switch box [20] eliminates the problem of routing domains and provides greater routing flexibility. This switch-block style was improved later in [10], to become *Imran* switch box as a combination of the disjoint and Wilton styles, which leads to a better trade-off between switch-block area and routability.

All of the previous works assume a switch-block flexibility $F_s = 3$, which means that every track entering and ending at one side of the switch-block will connect to three other tracks at the other three sides of the switch-block. This ensures a good practical compromise between the routing flexibility offered by the switch-block and its area as well as the run-time of the routing algorithms. Based on the analysis presented in
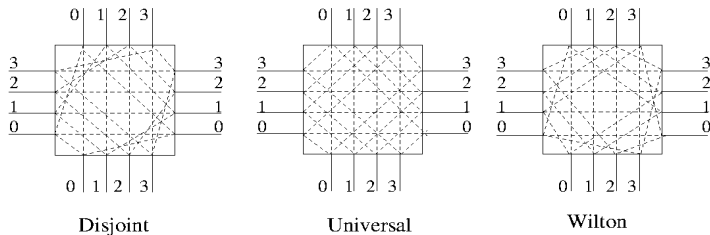
Figure 6: Typical Switch Box Configurations

the previous section, we design a new switch box, which will include hard-wired routing patterns. This means that we remove a certain number of programmable switches (derived from statistical analysis of the routing profiles of various circuits) from the switch boxes and replace them with wires. The composition of these hard-wired patterns are chosen after careful analysis of the routing profiles, hence the effect on the routability of circuits is minimized. We have experimentally verified the minimum effect of HARP switches on the routing of circuits (see Section 5).

Figure 7 shows some of the possible HARPs that can be present inside the switch boxes. The hard-wired patterns are shown using solid lines indicating that they are wires and not programmable switches. The next section describes how the routing tool is made aware of these patterns and how they are exploited to reduce the delay, area and power dissipation.
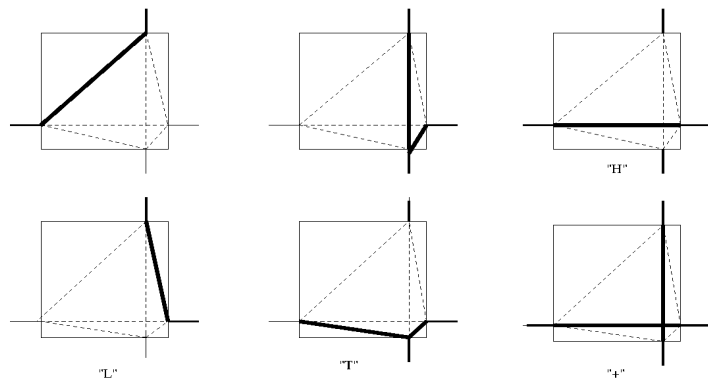
Figure 7: Some possible hard-wired patterns

4. ROUTING WITH HARPS

To harness the advantages of HARP architectures, the placement and routing tools must be adapted to use hard-wired resources for timing critical nets and only use regular switches where hard-wired resources are not available.

In this work, we would like to fully exploit the hard-wired patterns present inside our switch-blocks. This can be done in the detailed routing stage by constructing a routing graph with the hard-wired routing patterns embedded as low cost edges. VPR [2], and the power model from Wilton, *et. al.* [16], which we use for our work employ a routing graph construction approach to perform detailed routing. The routing segments and the logic block input and output pins are represented as vertices in the routing graph with a certain cost associated with them. Edges in the routing graph correspond to the connections between them. Edges maybe bidirectional or unidirectional depending on whether a pass-transistor or a buffered switch is used [3]. A sample routing graph is shown in Figure 8 [3].
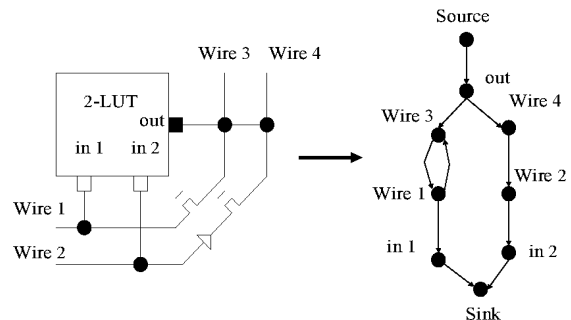
Figure 8: Sample routing graph

The way the routing graph is constructed changes with the presence of hard-wired patterns. These changes occur inside the switch boxes. Figure 9 shows the routing graph for a disjoint switch box (with pass transistor switches) with all tracks terminating at the switch box, and a disjoint switch box with a T-shaped hard wired pattern embedded in it. With the T-shaped pattern in the switch box, the routing graph contains only those edges forming the pattern and all the other edges are removed from the graph.
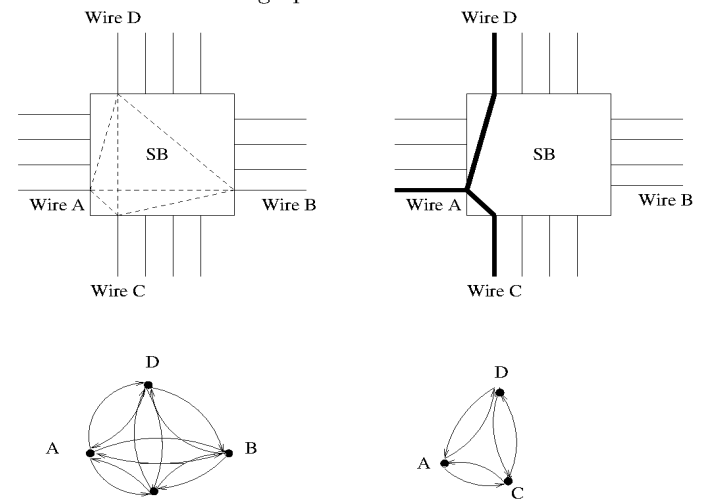
Figure 9: Routing graph with HARPs

Based on the results of the analysis presented in Section 3, we first determine the number of different HARP patterns that need to be inserted inside the switch boxes. Next, the FPGA chip is scanned row-by-row and patterns are inserted based on their desired percentages. When introducing these patterns, we take care not to connect different hard-wired patterns together

to form large trees. The reason for this restriction is illustrated by the example of Figure 10.

When we use two adjacent hard-wired patterns, an L-shaped pattern and a T-shaped pattern to connect terminal A to terminal B in the figure, a dangling segment is formed. This is undesirable as it adds extra capacitance and resistance, which is contrary to the goal of reducing overall power and delay. This problem is overcome by making sure that not many hard-wired patterns are connected back-to-back. In the rest of this section, we present our algorithms assuming that *no* two HARPs are allowed to connect back-to-back, but later on in Section 5, we relax this restriction a little and observe that the *limited* use of merged HARPs will improve the quality of the circuit.
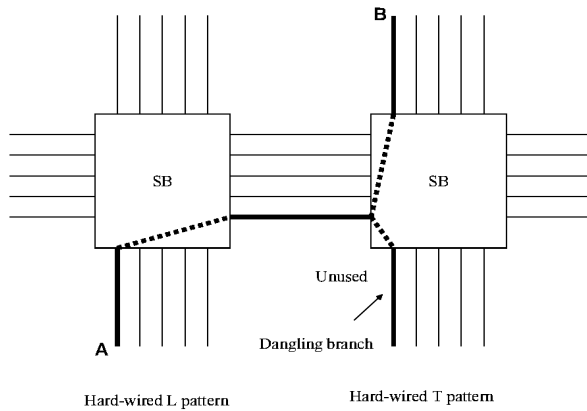
Figure 10: Connecting Hard-wired patterns together

Once we know the number and location of these hard-wired patterns, we change the way VPR constructs the routing graph and include only those edges (corresponding to wire segments) that are actually connected to the pattern[3]. These edges are inserted as low cost edges so that the router will automatically choose these hard-wired patterns when performing detailed routing. The cost is calculated based on the lumped resistance and capacitance of the wire segment (including HARP and regular segments) connected to a switch. The pseudo-code for inserting the hard-wired patterns inside switch boxes is shown in Algorithm 1.

Figure 11 shows how the pattern distribution array used in Algorithm 1 is populated. An array size of 10 is used for illustration purposes. In the figure, V, C, H, T and L denote Vertical, Cross, Horizontal, T and L shaped HARPs (no orientation considered yet). The percentages in the figure are 20%, 10%, 20%, 20% and 30% respectively and the patterns are distributed uniformly in the array (these numbers are taken from the statistical analysis of the routing pattern frequencies).
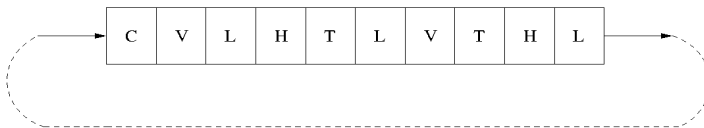
Figure 11: Initialization of Pattern Distribution Array (P)

Note that $\ulcorner$ and $\lrcorner$ can be combined into one switch configuration which makes two disjoint connections (one between right and bottom segments, and the other between top and left segments). The same is true with $\urcorner$ and $\llcorner$. See Figure 13 for

---

[3]For example, the "T" shapes of Figure 13 eliminate one edge from the side that T does not connect to.

examples of L patterns (in SB2, the fourth switch from the bottom is an $\{\ulcorner, \lrcorner\}$ switch). Our architectural generation code is available for download from [8] for non-commercial use.

Algorithm 1 Pseudo-code to Insert HARPs

Input:
Tech mapped netlist .net G(V,E)
Architecture description file

Initialization:
Initialize switch box(SB) matrix of dimensions (n_x, n_y, CW) to *unknown*
/*n_x, n_y denote number of channels in x and y directions and CW denotes channel width*/
Initialize Pattern distribution array(P) of size 100 with symbolic entries denoting various HARPs according to the frequency of distribution.
$Pattern\_Counter \leftarrow 1$

Algorithm:
for $i = 1$ to n_x **do**
   for $j = 1$ to n_y **do**
      for $track = 1$ to CW **do**
         $orientation \leftarrow$ Random $\{0, 90, -90, 180\}$
         $PatternID \leftarrow pattern\_id(P(pattern\_counter\%100),$
                           $orientation)$
         $S \leftarrow$ switches adjacent to switch(i,j,track)
         /*Here we assume a disjoint SB with F_s=3 and segments of lengths 1,2 and 6.*/
         $patternIncompatible \leftarrow false$
         for $\forall s \in S$ **do**
            if s and $PatternID$ make two joined HARPs **then**
               $patternIncompatible \leftarrow true$
            end if
         end for
         if $patternIncompatible == false$ **then**
            $SB(i, j, track) \leftarrow PatternID$
            $pattern\_counter \leftarrow pattern\_counter + 1$
         else
            $SB(i, j, track) \leftarrow regular\_switch$
            /*This denotes a regular switch where all sides of the SB participate in the connections*/
         end if
      end for
   end for
end for

/*To insert edges in the routing graph*/
for all Switch Boxes for i in $1 \ldots n_x$, j in $1 \ldots n_y$ **do**
   for $track = 1$ to CW **do**
      if $SB(i, j, track) > 1$ **then**
         Insert only the edges forming the pattern into the routing graph
         /*For example, if a T pattern is formed, then the north segment should be eliminated.*/
      else
         Insert all possible connections into the routing graph
      end if
   end for
end for

4.1 Estimation of Delay, Area and Power

We use the delay and area models in VPR and the power model developed by [16] to estimate the circuit delay, total area of the chip and the total power dissipation after inserting the hard-wired switches. VPR uses an Elmore delay model to estimate the delay of every net. In this model, pass transistors are represented as resistors and diffusion capacitances to

ground. Pass transistors add parasitic capacitance to the wire irrespective of whether they are on or off leading to a higher delay [3]. In our hard-wired switches, we eliminate the pass transistors and replace the resistance and capacitance values of the pass transistors, used in the delay model, with those of the metal wire (of segment length 1). This is illustrated in Figure 12.
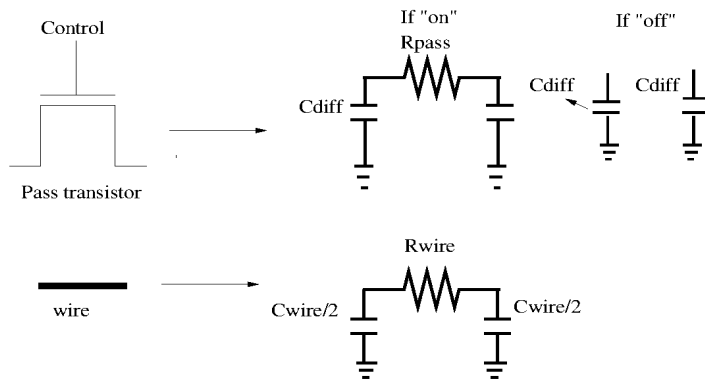

Figure 12: Modeling Pass transistor Switches

To accurately determine the delay of using a hard-wired switch, the capacitance of all the segments forming the pattern are included in the total capacitive load being switched. In addition, when only some of the segments of a hard-wired switch are used to route a signal, the remaining segments are made unavailable to route other nets. This avoids potential resource conflicts that could occur when different nets try to use different parts of the same hard-wired switch. The area model in VPR is based on counting the number of transistors required to implement the FPGA architecture and reports area in terms of the number of minimum width transistor areas required to implement the circuit on the FPGA [3]. For our hard wired switches, we use the same procedure and count the total number of transistors in our implementation. We use the power model developed by [16] to estimate the total power dissipation. Leakage power is estimated by counting the number of unused transistors and SRAM cells and multiplying them with their individual leakage power. Dynamic power is dependent on the charging and discharging capacitance and the clock frequency, which is the critical path delay. The short circuit power is taken as 10% of the dynamic power. The charging and discharging capacitance is obtained from the parasitics used in the delay model of VPR. Since we replaced the pass transistors with the resistance and capacitance values of metal wires for hard-wired switches, the power results reported in our paper accurately reflect the effect of these hard-wired patterns.

5. EXPERIMENTAL RESULTS

We inserted the hard-wired patterns in the switch boxes and used a multi-segment routing architecture with routing-segments of lengths 1, 2, 6, and long lines (similar to the Virtex architecture) for all simulation experiments. HARPs were not inserted on long lines, though. We placed and routed 10 MCNC circuit benchmarks of the VPR package on HARP architectures and report the results of circuit delay, area, leakage power, total power dissipation and channel width.

We updated the delay look up tables used by the placement tool of VPR to reflect delays of HARP connections. However, this had only a marginal impact on the placement quality. The reason is that these delay lookup tables are built assuming no congestion is present, and hence the best routing resources for

delay are always available. This is an optimistic lower bound on the routing delay between two points. In reality, the router will have to use traditional, slower switches for some nets due to congestion. As a result, the delay estimated at placement will not be an accurate representation of the delay at routing, and placement optimizations are not as effective. Consequently, we decided to leave the delay tables untouched from the original implementation in order to account for the fact that the router may not always be able to use HARPs for routing.

Results of the execution of VPR with 50% of all switches replaced with HARPs and that of the traditional "Virtex-like" architecture is presented in Table 1[4]. Columns labeled "Vtx" show the results of the traditional "Virtex-like" routing architecture. The last two rows of the table show average values for Vtx and HARP, and the ratio between HARP average and Vtx average.

We observe that the insertion of hard-wired routing patterns has a profound impact on delay and leakage power dissipation, reducing delay by about 21.7% and leakage power by 19.7% on average. Insertion of hard-wired routing patterns as low cost edges in the routing graph encourages the routing tool to use them whenever possible. This leads to a considerable speed up of the circuit. Also, the elimination of the program bits results in fewer SRAM cells and a lower leakage power dissipation. We find that the total area of the circuit decreases by about 5.3% on average. However, the average channel width increases by around 15.4%. This is expected, since, the introduction of hard-wired routing patterns reduces the flexibility of the routing architecture causing the router to use more tracks to route certain connections. However, the overall routing area of the circuit decreases because the reduction in individual switch area dominates the increase in number of switches caused by increased channel width.

Total power dissipation reduces on average by about 6.23%. In spite of the 20% reduction in the leakage power dissipation, the total power reduces by only around 6% on average. This is explained by taking into account the dynamic power dissipation. Dynamic power dissipation is dependent on the switching rate of the circuit. With hard-wired patterns in the switch boxes, the critical path delays of the circuits are reduced considerably resulting in a higher clock rate. This causes increased dynamic power dissipation. A fair comparison metric between the Virtex-like routing architecture and HARPs would probably be the power-delay product. We can explore different configurations by considering the energy dissipation or the power-delay product. Power-delay of HARP is 30% better than Vtx. Depending on whether optimizing for speed or power is more critical, we can clock the circuits at a higher clock frequency to get a faster circuit or we can clock the circuit at a lower speed (e.g., the clock speed that a traditional Virtex routing architecture can achieve) to achieve more savings in power dissipation.

Another point to be observed is that the performance of the hard-wired patterns is considerably better for larger circuits. From Table 1, we observe that for the 3 biggest circuits, spla, pdc and ex1010, introducing hard-wired patterns cause an improvement of 26% in the circuit delay and 34% in the total energy consumption. Leakage power is becoming increasingly more important compared to other sources of power consumption according to ITRS [9]. As a result, our proposed HARP architecture will become more effective as technology advances.

We also explored the potential benefits of increasing the per-

---

[4]Results for only 10 of the benchmarks are reported for want of space. The others yield similar results

Circuit	Delay (x10^{-8})		Area (x10^6)		Channel Width		Leakage power		Total power		Energy (x10^{-8})	
	Vtx	HARP	Vtx	HARP	Vtx	HARP	Vtx	HARP	Vtx	HARP	Vtx	HARP
misex3	6.31	5.33	2.88	2.71	20	23	0.119	0.094	0.223	0.222	1.407	1.183
alu4	7.12	5.97	3.11	2.74	19	21	0.129	0.096	0.235	0.225	1.673	1.343
apex4	7.17	5.98	2.83	2.58	22	24	0.117	0.092	0.183	0.166	1.312	0.993
ex5p	6.40	5.50	2.36	2.23	22	25	0.098	0.080	0.173	0.164	1.107	0.902
des	7.89	6.00	6.02	5.75	16	18	0.251	0.209	0.415	0.408	3.274	2.448
seq	6.39	5.31	3.58	3.46	20	24	0.149	0.121	0.271	0.270	1.732	1.433
apex2	7.45	5.92	4.01	3.80	21	24	0.168	0.134	0.281	0.281	2.093	1.663
spla	12.40	8.93	9.90	9.70	28	33	0.428	0.342	0.525	0.479	6.51	4.278
pdc	14.20	10.70	14.80	13.70	33	39	0.637	0.505	0.750	0.652	10.65	6.976
ex1010	15.50	11.50	8.95	8.66	19	23	0.378	0.314	0.477	0.448	7.393	5.152
Avg	9.083	7.114	5.844	5.533	22	25.4	0.2474	0.1987	0.3533	0.3315	3.715	2.637
Ratio		78.32%		94.68%		115.45%		80.32%		93.83%	70.98%	

Table 1: Comparison of 50% HARPs with no HARPs

centage of HARPs inside the switch boxes by allowing a small percentage (10%) of HARPs to connect to each other. This is illustrated in Figure 13, which shows HARP switches (HARP SW) and regular switches (FlexSW) lumped to form distinct regions inside the switch boxes. This representation is just for illustration purposes. In reality, HARPs are distributed throughout the switch boxes, and not just at lower tracks.
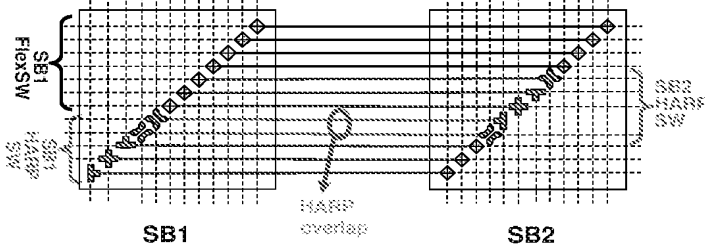
Figure 13: Overlaps in HARPs

Allowing a small percentage of HARPs to connect directly could create more complex routing patterns to be formed by combining hardwired patterns that we have used. However, doing so could also have the undesired affect of creating dangling wire segments (as illustrated in Figure 10) which could have an adverse effect on delay and power. Part of our future work is to find the best percentage of HARPs that could connect directly. In terms of implementation, to increase the percentage of HARPS beyond 50%, we relax the constraint present in algorithm 1 of not allowing different HARPs to connect together and allow HARPs to connect together sometimes (*e.g.*, to allow 60% of the switches to be HARP, we should allow HARPs to connect 10% of the time). As before, we take care not to form large trees of HARPs. This is done by making sure that there is at least one regular switch after every K switches, K being a constant (in our experiments, we used $K = 3$). The results of increasing the percentage of HARPs is presented in Table 2.

We observe that increasing the percentage of HARPs(60%) inside switch boxes increases the potential savings in circuit delay, energy and area to about 24%, 34% and 7% respectively.

6. RELATED WORK

There has been a lot of work on programmable architectures to improve the performance of FPGAs. Modern FPGAs utilize multi-length horizontal and vertical segments. Recently, there has been a flurry of research in structured ASIC solutions [24], which aim to provide a middle ground between ASICs and FPGA. Tong *et. al.* [18], Jayakumar and Khatri [11], and

Yan and Marek-Sadowska [22] proposed via-configurable gate array implementation platforms, in which connections are programmed by the presence or absence of vias. This results in improvements in power-delay performance but the flexibility and cost savings are limited because of its mask programmability.

There have been several CAD techniques aimed at reducing the number of "bends" in the routing architecture. This paper was inspired by our initial work on pattern routing [12, 13]. This work focused on the concept of using prespecified patterns to route a net. By doing so, we allow a more accurate prediction mechanism for metrics such as congestion and wirelength earlier in the design flow. The work focused on an ASIC design flow, rather than the FPGA flow found in this paper.

Maidee *et. al.* [15] proposed a "terminal alignment" heuristic, which reduces the number of bends on nets, and hence eliminates switches that need to connect horizontal tracks to vertical ones. As a result, the number of switches used in routing of critical nets decreases. They achieved 5% delay improvement over VPR.

7. DISCUSSION AND CONCLUSION

We propose a technique to reduce circuit delay, area and power dissipation by introducing hard-wired patterns inside switch boxes. The population of the HARPs is guided by statistical analysis of routing trees that are generated on a traditional architecture by the VPR tool. We analyzed the routing profiles of various circuit benchmarks and came up with a statistical measure of the routing patterns present inside the switch boxes. The routing graph construction of VPR was modified to include these patterns. Simulation results after detailed routing showed a potential improvement of 24% in circuit delay, 7% in the circuit area, 24% in the leakage power dissipation and about 8% in the total power dissipation. We observed that by introducing hardwired patterns, we can considerably speed up the circuit and at the same time achieve reasonable savings in circuit area and power dissipation.

In Section 3.1 we mentioned that the placement and routing algorithm and the architecture will affect the outcome of the statistical analysis. In Section 3.2.2 we showed that the architecture has a bigger role compared to the routing algorithm. But nevertheless, both the physical design algorithms and the architecture will skew the pattern frequency analysis. Apart from the fact that there is a certain degree of inevitability in this influence, we argue that such effect could be considered useful. We would like to generate the architecture with an eye on the CAD algorithms. If we create an architecture that con-

Circuit	Delay x(10^{-8})	Area x(10^6)	Channel Width	Leakage power	Total power	Energy x(10^{-8})
misex3	4.98	2.68	24	0.093	0.226	1.125
alu4	5.67	2.73	22	0.0101	0.228	1.292
apex4	5.74	2.59	26	0.093	0.168	0.964
ex5p	5.6	2.19	26	0.062	0.157	0.879
des	6.36	5.66	18	0.188	0.388	2.467
seq	5.30	3.45	26	0.121	0.268	1.420
apex2	6.10	3.74	25	0.114	0.270	1.647
spla	8.31	9.38	33	0.314	0.465	3.864
pdc	9.42	13.5	40	0.490	0.645	6.075
ex1010	11.4	8.45	24	0.299	0.429	4.891
Average	6.88	5.437	26	0.188	0.324	2.463
Ratio	0.758	0.930	1.20	0.760	0.917	0.663

Table 2: Comparison of 60% HARPs with no HARPs

forms to the behavior of the placement and routing algorithms, the potential benefits will be greater. Further work is needed in making the placer aware of the changes in the routing architecture. We also need to look at the possibility of modifying Steiner tree routing algorithms to make full use of the hard-wired patterns and to achieve better correlation between the placement tool, routing tool and the routing architecture.

8. REFERENCES

[1] J. H. Anderson, F. Najm, and T. Tuan. "Active Leakage Power Optimization for FPGAs". In Proc. of ACM/SIGDA International Symposium on Field programmable gate arrays, 2004.

[2] V. Betz and J. Rose. "VPR: A New Packing, Placement and Routing Tool for FPGA Research". In *International Workshop on Field-programmable Logic and Applications*, 1997.

[3] V. Betz, J. Rose, A. Marquardt. Architecture and CAD for Deep-Submicron FPGAs. Kluwer Academic Publishers, 1999.

[4] Y.-W. Chang, Y.-T. Chang, "An Architecture-Driven Metric for Simultaneous Placement and Global Routing for FPGAs," In Proc. of ACM/IEEE Design Automation Conference, 2000, pp. 567-572.

[5] Y.-W. Chang, D. F. Wong, "Universal Switch Modules for FPGA Design," ACM Trans. Design Automation of Electronic Systems, Vol. 1, No. 1, Jan. 1996, pp. 80-101.

[6] A. DeHon, "Reconfigurable architectures for general-purpose computing", Ph.D. dissertation, Massachusetts Institute of Technology, 1996.

[7] A. Gayasen, K. Lee, N. Vijaykrishnan, M. Kandemir, M.J. Irwin, and T. Tuan "A Dual-V_{DD} Low Power FPGA Architecture". In *Proceedings of International Conference on Field Programmable Logic and Applications*, 2004.

[8] HARP Architecture Generator and P&R Tool URL, http://www.ece.umn.edu/users/kia/ (click on the Download link)

[9] The International Techonlogy Road Map for Semiconductors, 2003 Edition.

[10] M. Imran Masud. FPGA Routing Structures: A Novel Switch Block and Depopulated interconnect Matrix Architecture. M.A.Sc. Thesis, University of British-Columbia, 1999.

[11] N. Jayakumar and S. P. Khatri, "A METAL and VIA Maskset Programmable VLSI Design Methodology using PLAs", In *Proceedings of International Conference on Computer Aided Design*, 2004.

[12] R. Kastner, E. Bozorgzadeh,and M. Sarrafzadeh, "Predictable Routing", In *Proceedings of International Conference on Computer Aided Design*, 2000.

[13] R. Kastner, E. Bozorgzadeh,and M. Sarrafzadeh, "Pattern routing: use and theory for increasing predictability and avoiding coupling", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 7, No. 7, Pages 777-790, 2002.

[14] M. Khellah, S. Brown, Z. Vranesic, "Minimizing Interconnection Delays in Array-based FPGAs," In Proc. of IEEE Custom Integrated Circuits Conference, 1994, pp. 181-184.

[15] P. Maidee, C. Ababei, and K. Bazargan, "Fast Timing-driven Partitioning-based Placement for Island Style FPGAs", In *Proc. ACM/IEEE Design Automation Conference (DAC)*, 2003, pp. 598-603

[16] K. Poon, A. Yan, and S. Wilton, "A flexible Power Model for FPGAs". In *Proceedings of International Conference on Field Programmable Logic and Applications*, 2002.

[17] J. Rose, S. Brown, "Flexibility of interconnection structures for field-programmable gate arrays," IEEE J. Solid-State Circuits, 26, 3, 1991, pp. 277-282.

[18] K.Y. Tong, V. Kheterpal, V. Rovner, L. Pileggi, H. Schmit "Regular Logic Fabrics for a Via Patterned Gate Array (VPGA)". In *Proceedings of IEEE Custom Integrated Circuits Conference*, 2003.

[19] VPR Pattern Finder URL, http://www.ece.ucsb.edu/~express/software.html, 2004

[20] S. Wilton. Architecture and Algorithms for Field Programmable Gate Arrays with Embedded Memory. Ph.D. Thesis, University of Toronto, 1997.

[21] XC4000 FPGA Family Data Sheet. Xilinx, Inc.

[22] Y. Ran, and M. Marek-Sadowska, Designing a Via-Configurable Regular Fabric. In Proceedings of Custom Integrated Circuits Conference (2004).

[23] S. Yang, "Logic Synthesis and Optimization Benchmarks, Version 3.0", *Tech. Report, Microelectronics Centre of North carolina*, 1991.

[24] B. Zahiri, Structured ASICs: Opportunities and challenges. In *Proceedings of International Conference on Computer Design (2003), pp. 404409.*

[25] P. S. Zuchowski, C. B. Reynolds, R. J. Grupp, S. G. Davis, B. Cremen and B. Troxel, "Hybrid ASIC and FPGA Architecture", *International Conference on Computer-Aided Design (ICCAD)*, pp. 187 – 194, 2002.

Generating Highly-Routable Sparse Crossbars for PLDs

Guy Lemieux
Dept. of Elec. & Comp. Eng.
University of Toronto
Toronto, Canada
lemieux@eecg.toronto.edu

Paul Leventis*
Right Track CAD Corp.
313-720 Spadina Ave.
Toronto, Canada
paul@rtrack.com

David Lewis
Dept. of Elec. & Comp. Eng.
University of Toronto
Toronto, Canada
lewis@eecg.toronto.edu

ABSTRACT

A method for evaluating and constructing sparse crossbars which are both area efficient and highly routable is presented. The evaluation method uses a network flow algorithm to accurately compute the percentage of random test vectors that can be routed. The construction method attempts to maximize the spread of the switch locations, such that any given subset of input wires can connect to as many output wires as possible. Based on Hall's Theorem, we argue that this increases the likelihood of routing.

The hardest test vectors to route are those which attempt to use all of the crossbar outputs. Results in this paper show that area-efficient sparse crossbars can be constructed by providing more outputs than required and a sufficient number of switches. In a few specific case studies, it is shown that sparse crossbars with about 90% fewer switches than a full crossbar can be constructed, and these crossbars are capable of routing over 95% of randomly chosen routing vectors. In one case, a new switch matrix which can replace the one in the Altera FLEX8000 family is shown. This new switch matrix uses approximately 14% more transistors, yet can increase the routability of the most difficult test vectors from 1% to over 96%.

1. INTRODUCTION

Programmable logic devices commonly use full crossbars and sparse crossbars as building blocks in routing networks. Typically, a full crossbar is chosen when a highly-routable crossbar is desired, and a sparse crossbar containing significantly fewer crosspoints is selected when area use is most important. This naturally brings up the question, "Is it possible to get the best of both worlds?"

There are many instances where a highly routable crossbar would be preferred, but the area cost of a full crossbar is prohibitive. For example, the Plasma FPGA [5] in the Hewlett-Packard Teramac [4] reconfigurable logic system would have used full crossbars to guarantee routability. However, to save area, it was necessary to use only 1/4 of the the switches.

In Teramac and large-scale logic emulation systems, such as those by Quickturn [18], circuits are partitioned across a large number of

---

*This work was performed at the University of Toronto.

FPGAs. Each of the generated subcircuits must successfully place-and-route in an FPGA, otherwise time-consuming re-partitioning and re-routing is required. To make routable subcircuits, one can intentionally underutilise the LUTs in the FPGA [9], or use an FPGA that is designed to be highly routable.

Highly routable components in a single FPGA can also benefit users by reducing compute time and memory use. The latest FPGAs by Altera and Xilinx have a large number of LUTs and wiring resources. To route these FPGAs, CAD tools usually store the following details in memory: a representation of the circuit, its mapping to FPGA resources, and a model of the entire FPGA. This leads to considerable memory use. For example, Altera recommends using 1GB of RAM to route designs for the APEX 20K1000E device [2]. It may be possible to make the CAD tools more efficient if they follow the Teramac and logic emulation system model: partition a circuit into smaller subcircuits, then place and route each piece independently. To do this effectively without rip-up and re-partitioning, there must be confidence that each subcircuit is very likely to route.

As another example, CPLDs are required to be highly routable because they are often close to 100% utilised. Full crossbars are not normally used in the global interconnect of CPLDs due to the area overhead involved, so an area-saving sparse pattern is required.

The above scenarios indicate that highly routable, sparsely populated crossbars would be useful, yet there is little published work in this area. In this paper, this issue is addressed by describing conditions for routability (Hall's Theorem), a method for evaluating routability without resorting to place-and-route experiments, and a construction algorithm that achieves good performance. Results for a few design cases are shown to exemplify the area requirements and routability obtainable from sparse crossbars.

2. CROSSBAR TYPES AND PROPERTIES

An $n \times m$ crossbar connects n different input wires to m output wires, typically with $n \geq m$. An example of a few crossbars are shown in Figure 1. At the locations where an input crosses an output wire, a programmable switch, or *crosspoint*, may be present. We use the term *capacity* of a crossbar to mean the number of signals being routed through it. The term *population* refers to the number of switches in the crossbar, p.

2.1 Full Crossbars

A *fully-populated crossbar* or *full crossbar* contains switches at every intersection point of the input and output wires, using a total of $p = n \cdot m$ switches. The term *crossbar* usually refers to a full crossbar. An example of a full crossbar is shown on the left in Figure 1. Full crossbars are extremely flexible because they can connect *any* wire on the input side to connect to *any* wire on the output side, *i.e.* they support any permutation of the outputs. Additionally, full
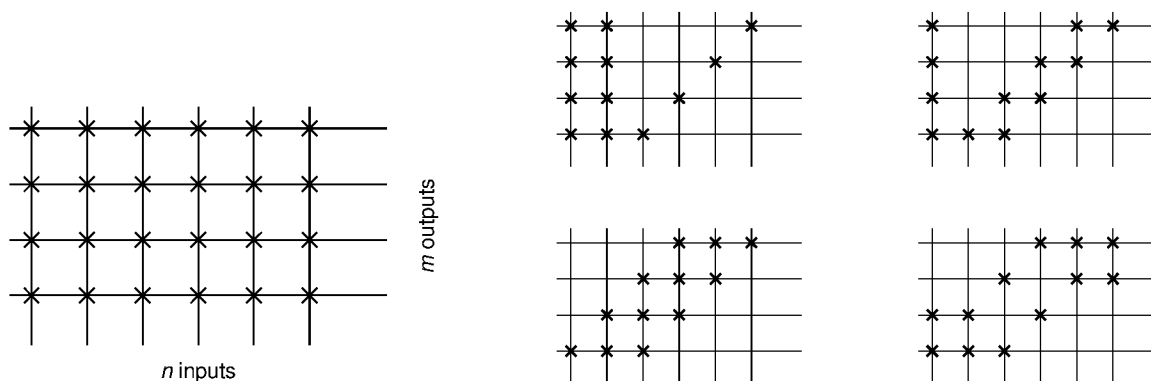
Figure 1: Examples of 6×4 crossbars: a full crossbar on the left, full-capacity minimal crossbars on the right.

crossbars can be used at *full capacity*: they can connect as many signals as the number of outputs in the crossbar.

2.2   Full-Capacity Minimal Crossbars

Full-capacity minimal crossbars are well-known constructions that use fewer switches than a full crossbar. They are slightly less flexible than full crossbars, but they retain the full-capacity property. For convenience, we shall refer to them simply as *minimal crossbars.*

Minimal crossbars are less flexible than full crossbars because they remove the freedom to assign a specific input wire to a specific output wire. Thus, any m input wires can be connected to all m output wires, but the ordering of the signals on the output wires may not be freely chosen.

A minimal crossbar always uses $p = (n - m + 1) \cdot m$ switches. Nakamura [16] has shown that no switches can be removed from a minimal crossbar without also removing the full-capacity property. Minimal crossbars do not save many switches when $n \gg m$, but the number of switches is reduced from a quadratic expression to a roughly linear one when $n \simeq m$.

There are many different topologies for minimal crossbars, a few of which are shown in Figure 1. The simplest topology, called a *fat-and-slim crossbar*, uses a full crossbar between the first $n - m$ input wires and all m output wires. Each of the remaining m input wires have only one switch and are connected to a different output wire. This results in balanced *fan-in* for the output wires, but largely unbalanced *fan-out* on the inputs.

Some minimal crossbar topologies simultaneously balance the number of switches on the input and output wires. Fujiyoshi [11] defines a class of minimal crossbars called *bi-scattered* which have naturally balanced fan-in. They also provide a switch placement algorithm to generate bi-scattered crossbars with balanced fan-out. Guo [12] suggests a transformation that redistributes switches of a fat-and-slim crossbar, yet preserves full-capacity and the already-balanced fan-in arrangement. They prove this transformation can be used to obtain balanced or nearly-balanced (within ± 1) fan-outs.

2.3   Perfect Crossbars

Since both full and minimal crossbars support full-capacity, it is convenient to refer to them as *perfect* crossbars. Perfect crossbars are one way to implement an *(n, m)-concentrator*, a type of graph that can disjointly route any m-sized subset of the n inputs to m outputs.

2.4   Sparse Crossbars

A *sparse crossbar* refers to a crossbar which has few switches, *i.e.* is sparsely populated. The demarcation point of when a crossbar becomes "sparse" is debatable: for example, nearly square crossbars can be sparsely populated yet support full capacity. This paper assumes that a crossbar is sparse if it contains fewer than $p < (n - m + 1) \cdot m$ switches. Hence, no matter how well it is designed, a sparse crossbar can never be made perfect.

Oruç [17] has proven that a sparse crossbar of guaranteed capacity c, where $c \leq m$, must contain $p \geq \lceil m \cdot (n - m + 1)/(m - c + 1) \rceil$ switches. This lower bound is not necessarily tight, but when $c = m$, the number of switches in a minimal crossbar is obtained. Sparse crossbars with guaranteed capacity c are also referred to as *(n, m, c)-concentrators.*

The transformation suggested by Guo [12] can also be applied to sparse crossbars, provided that there exists some input which covers all of the outputs reachable by another input. The transformation states that switches can be moved from the one input to the other, causing the guaranteed routing capacity of the crossbar, c to increase or stay the same — but it will never decrease. This is a beneficial transformation, but it is not often applicable in sparse crossbars because it is uncommon to have one input completely cover another.

2.5   Graph Representation

Crossbars are easily modeled as a graph when wires are represented by nodes and switches are represented by edges. A crossbar forms a bipartite graph G composed of two sets of nodes and a set of edges. The node sets are a set of input wires I and a set of output wires O. There are no edges within each set, but an edge can exist between any node in set I and any node in set O.

3.   EVALUATING ROUTABILITY

The traditional approach to evaluate the routability of an FPGA, and hence evaluate the sparse crossbars contained therein, is to run place and route experiments with a suite of benchmark circuits. This is an effective method to design an FPGA and its CAD tools in concert, but it can be a lengthy process. As well, the routing performance of the crossbars in the FPGA relies upon the effectiveness of the CAD tools and the benchmarks to exercise the architecture.

Our goal was to find a quicker way to test the routability of sparse crossbars independently of the CAD tool or benchmark circuits used. As well, this new method should provide a more sensitive, yet still practical, measurement of routability. This approach also helps avoid the problem of "training" an FPGA architecture or CAD tool to a particular benchmark suite.

One routability metric considered was the maximum guaranteed capacity of a crossbar, c. With this metric, we wish to find the largest value c such that any subset $I' \subseteq I$ of size $|I'| \leq c$ is guaranteed to be routable. The main problem with this metric is that it is very difficult to compute: the algorithm has inherently exponential complexity because it must examine all subsets of I with cardinality c or smaller. We implemented a branch-and-bound algorithm to search for this

value, but it is impractical for large crossbars. A greedy heuristic search was also implemented, but the results were not robust.

Instead, the routability of a crossbar is measured using a Monte Carlo test. For this test, a number of random test vectors are generated, and each is routed on the crossbar using a network flow algorithm. The routability of the crossbar is estimated as the percentage of test vectors which can be successfully routed.

A test vector of size k is the number of signals to be routed through the crossbar. More specifically, it is a subset of the input wires, $I' \subset I$, where $|I'| = k$. In terms of real FPGA routing, this test vector represents the case where logic signals have already been assigned to specific wires due to previous routing restrictions.

A highly-routable crossbar must be able to route many of these pre-constrained vectors. We evaluate routability as a function of the number of signals in a test vector. This distinguishes the easily routed vectors, i.e., when k is small, from the difficult ones.

In this paper, we arbitrarily define the *highly-routable* point as being able to route at least 95% of the hardest test vectors, i.e., those containing the maximum number of signals intended to be carried by the crossbar.

A network flow algorithm [7] is used to route the test vectors because it is *guaranteed* to find a routing solution if one exists. When routing a test vector of size k, switches are assigned unit capacity, so a flow of size k must be found to produce a solution. If a lower flow value is found, it indicates the largest number of wires that were actually routable. This guarantee of finding a solution is important in that it represents an ideal routing tool, hence it isolates the effectiveness of the CAD tools from the performance of the crossbar.

4. ROUTABLE SWITCH PATTERNS

This section looks at the following basic problem: given p switches, how should they be placed in a sparse crossbar to make it as routable as possible. The foundations for the switch-placement algorithm presented in the next section are based on the following theorem and observations.

4.1   Hall's Theorem

Hall's Theorem [13] is a result that can be applied to bipartite graphs defining whether a *maximum matching* can be found. A matching is a subset of the edges in the graph such that no two edges share a node. Hence, every pair of edges in a matching involve 4 distinct nodes. Hall's Theorem gives the precise conditions under which a matching can exist.

Hall's Theorem. Given bipartite graph G composed of a set of edges, E, and two independent sets of nodes, X and Y, then G has a matching of X into Y if and only if

$$\forall S \subseteq X, |S| \leq |N(S)|$$

where $|S|$ denotes the cardinality of subset S, and $N(S)$ is the set of neighbours of S in Y.

4.2   Application of Hall's Theorem

In terms of sparse crossbars, a matching actually forms a routing solution of a sparse crossbar. The Y set represents the output wire set O, and X is a specific test vector of the input wires $X = I' \subseteq I$. A test vector is routable if and only if Hall's condition is satisfied, and the matching gives the solution. The edges in the matching are the switches which must be turned on to form the connections.

To design a routable sparse crossbar, switches should be placed so that Hall's condition is satisfied for as many test vectors as possible. For test vectors of size k, it is a necessary condition that at least k distinct output wires are reachable by switches.

The switch placement algorithm described in the next section assumes that the switches placed on any specific subset of input wires

should be spread out to as many output wires as possible. This is equivalent to making the neighbour set $N(S)$ as large as possible so that Hall's condition is satisfied.

Switch placement is not trivial because the switch pattern chosen for one subset of input wires may consequently make the pattern for some other subset too close. We argue that this also implies that each input wire, having equal likelihood of being a part of any particular subset, should have an equal number of switches. If one input has fewer switches, it would not be able to "spread out" to as many different neighbours. As a result, subsets which included this input may be less routable. To get around this, it should be given more switches so the fan-outs of the input wires are roughly equal. A similar argument implies that the fan-ins of the output wires should also be balanced. For this reason, the switch matrices constructed in this paper all have balanced fan-in and fan-out.

4.3   Hamming Distance and Coding Theory

The switch placement problem requires that subsets of the input wires span as many output wires as possible. Doing this for every possible subset of input wires is a difficult task, so we chose to approximate this by spreading out the switches for every pair of input wires. In this form, the switch placement problem becomes identical to the problem designing communication codes so that code-design techniques such as those from [14] can be used.

The location where switches are placed on an input wire can be represented by a bitvector of length m, where a 1 in the bitvector indicates that a switch is present. There are n such bitvectors, one for each input, forming the codewords of a binary code.

The number of neighbours of an input wire subset is the number of ones in the bitwise-OR of their bitvectors. Given two bitvectors, bv_1 and bv_2, the increase in the number of neighbours (output wires) reached by the combination of the two is related to the Hamming distance[1] between them, $d(bv_1, bv_2)$. Spreading out the switches between a pair of input wires i and j is the same as maximizing $d(bv_i, bv_j)$. Code design techniques attempt to maximize the minimum d between all of the codewords.

5. SWITCH PLACEMENT ALGORITHM

In our construction algorithm, the switch pattern is determined in two stages: first an initial switch pattern is chosen, then that pattern is iteratively optimized. The minimum inputs required are the matrix size, $n \times m$, and the number of switches p.

5.1   Initial Switch Pattern

The goal of the initial switch pattern is to place switches so they will obey given fan-in and fan-out specifications. These specifications form a limit on the number of switches that will be placed on each wire. The user may provide any valid fan-in and/or fan-out distribution, or, if no specification is provided, a balanced one is automatically generated based on p, n and m.

A switch pattern which obeys the fan-in/out specifications is generated in one of two ways: either randomly, or by network flows. The random method generates random locations in the crossbar and places a switch there if it won't violate the fan-in/out specifications. If, after a certain number of tries, it cannot find a valid location for the next switch, it erases all switches and starts over. Usually an initial pattern is found the first time, unless there are a large number of switches to place. If it still fails after restarting a number times, the tool falls back to the network flow method.

The network flow method temporarily places a switch at every location in the crossbar, and assigns each a unit capacity. The maximum flow from the input to output wires is found, using the fan-out and fan-in specifications as flow capacities for the wires. If an initial

---

[1] The Hamming distance is the number of bit positions that differ between the two bitvectors.

Figure 2: The switch matrix on the left has identical Hamming costs before and after the swap indicated. After the swap, it cannot route any subsets which include wires $\{1,5,6\}$. Hence, the cost function is not always effective at distinguishing good switch swaps. The switch matrix on the right has lower Hamming cost after the swap indicated, and routes all subsets of size 3. Before the swap it could not route subset $\{1,2,3\}$. In this case, the cost function can identify a good swap.

switch pattern can be generated to obey the given constraints, it will be found as solution with a total flow of p. The switches which the flow solver used are kept, and the other switches are discarded. The network flow method is used as a backup method because solving the flow network is usually slower than the random method.

5.2   Switch Placement Optimizer

The routability of an initial switch pattern can be improved by moving a number of switches to produce a more "spread out" pattern using a number of "switch swaps". A simulated-annealing approach was initially used [15], following the approach used in [10]. Since the overall goal was similar, we chose to minimize the same cost function:

$$\sum_{\forall i,j} \frac{1}{d(bv_i, bv_j)^2}.$$

In the process of designing many switch matrices, it was noticed that any hill-climbing moves which raised the cost function were nearly always found again and undone. Instead, an algorithm that follows the simple approach of accepting any swap that lowers the cost function proved to be just as effective and considerably faster. An algorithm that systematically evaluated all possible swap candidates was also tried, but that algorithm ran considerably slower.

The resulting improvement algorithm works in a greedy fashion: it generates random swap candidates, but it only accepts the swaps if the routability improves. The algorithm stops when it is unable to find any improvement in cost after checking a large set of swap candidates (about 10 000, for example).

5.3   Cost Function Pitfalls

Other cost functions have been considered. As noted in [10], the alternative of maximizing the minimum Hamming distance of the code is difficult because not all switch swaps would lead to an observable change in the cost function.

Another cost function would be to maximize the total Hamming distance between all pairs, *i.e.*,

$$\sum_{\forall i,j} d(bv_i, bv_j).$$

Unfortunately, this does not sufficiently penalize close bitvectors. For example, consider the 3 bitvectors 111000, 011100, 000111 with Hamming distances of 2, 4, and 6. The alternative switch topology 111000, 001110, 010011 gives distances of 4, 4, and 4 and has better routability. However, no difference between these bitvector sets is found if only the total Hamming distance is examined.

It would also be possible to run a Monte Carlo simulation and accept a swap only if routability improved. However, this leads to two problems. First, a Monte Carlo simulation would be much slower to compute. Second, the results of a single swap may not be readily discernible by the simulation.

In comparison to the above alternatives, the Hamming distance cost used is relatively quick to compute and it can distinguish most (but not all) changes to the switch pattern.

5.4   Generating Swap Candidates

Swap candidates are determined by the four intersection points of two input wires and two output wires. To preserve the fan-in/out distribution profiles, a swap operation must consist of two switches and two empty locations positioned diagonally on the intersection points.

To generate a swap candidate, two input wires are chosen at random. Given the placement of switches on these two wires, two output wires are randomly selected chosen to form a swap candidate. If no valid candidate exists, a new pair of input wires is chosen.

The fan-in/out distribution profiles can also be preserved while moving a single switch, provided the following conditions are met. A switch can be moved to another output wire (along the same input wire) if the original output wire fan-in is one greater than the new output wire fan-in before the move. Similarly, a switch can be moved along an output wire provided the fan-outs of the old and new input wire locations differ by one. Improvements arising from these single swaps are done exhaustively after the greedy algorithm gives up on swapping switch pairs.

5.5   Limitations of the Algorithm

When $p \leq 2n$, the switch matrix is very sparse and we have examples of suboptimal performance by our algorithm. Under these conditions, small disconnected components may be present in the bipartite graph, yet they are indistinguishable by the cost function. For example, consider the leftmost matrix in Figure 2. Performing the switch swap indicated produces a matrix with identical cost, yet the new matrix is not as routable. The original matrix routes any test vector of size 3, but the new matrix cannot route the input subset $\{1,5,6\}$.

In contrast, consider the switch matrix on the right of Figure 2 which cannot route the subset $\{1,2,3\}$. Here, the algorithm will find the single switch move indicated to lower the cost, and the resulting switch pattern routes *all* groups of 3.

6.   RESULTS

We have developed a tool in C++ to construct and test sparse crossbars using the switch placement algorithm and evaluation method described above. A number of routing experiments have been run on sparse crossbars with 168 inputs and 24 outputs. This default size was chosen because it is small enough to run experiments quickly, and it is the same size as the one used in Altera's FLEX8000 family [1]. Altera has confirmed that the FLEX8000 sparse crossbar contains 2 switches for every crossbar input [3], however we are not privy to the location of the switches.

6.1   Adding Extra Switches

The first set of experiments investigate how sensitive the routability of a sparse crossbar is to the addition of switches. It is uncer-
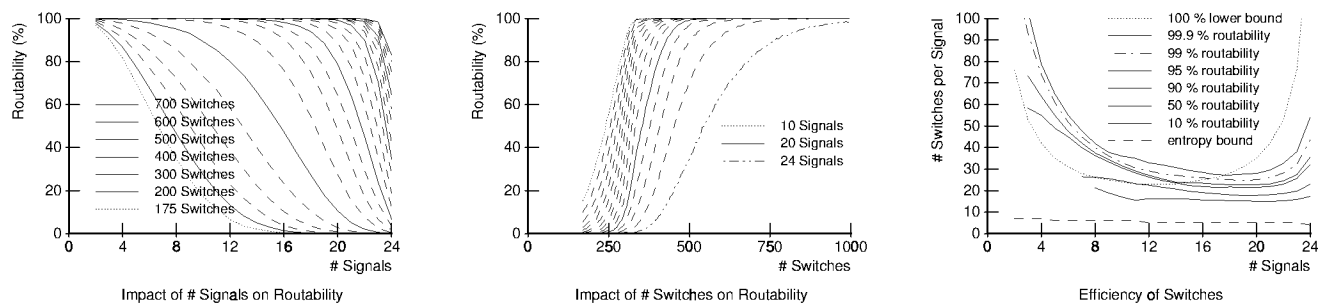
Figure 3: The effect of adding extra switches on routability of a 168 × 24 crossbar. The number of output wires is fixed at 24 in each graph. Three different ways of looking at the same routability data are presented. Monte Carlo simulations were done with 10,000 test vectors for each signal level.

tain if there is an obvious breakpoint in routability improvement, if routability increases in a smooth or discrete fashion, or how many switches are required to maintain a desired level. In Figure 3, a number of graphs and curves of the same data are shown to illustrate the effect of adding switches on routability.

The leftmost graph in Figure 3 shows a routability curve for each fixed switch count. For a given number of signals (*i.e.*, test vector size), 10,000 random vectors are routed. One curve shows the percentage of vectors routed as the test vector size increases. Clearly, large test vectors are more difficult to route, and sometimes the dropoff is very rapid. Each curve represents holding the number of switches constant: from 175 to 700 switches, in steps of 25. As switches are added, the entire routability curve shifts upward. Typically, the amount of the shift decreases as more switches are added, implying less utility is gained from each additional switch.

The middle graph in Figure 3 shows a similar routability curve for each test vector size, but the number of switches varies along the *x*-axis. For a given test vector size of 20 signals, for example, the greatest improvement in routability occurs as the number of switches is increased from 300 to 500. The largest test vector size of 24 signals shows the slowest improvement rate, and requires a large number of switches to become highly routable.

The rightmost graph in Figure 3 measures the utilisation of switches when routing the test vector signals. Each curve represents a fixed routability level, say 90%. The *x*-axis is the number of signals to be routed, and the *y*-axis is the smallest number of switches per signal required to achieve 90% routability. The curves show that most, but not all, of the crossbar outputs should be used to make efficient use of the switches. If nearly all of the outputs are used, *i.e.*, more than 20, significantly more switches per signal are needed to sustain the desired level of routability. Hence, the value obtained by adding each additional each switch in this region is small; many switches are needed to make a significant contribution to routability.

Two additional curves of interest are shown in this third graph: the entropy curve and the lower bound curve. The 100% lower bound curve, obtained from the formula $\lceil (n-k+1)m/(m-k+1) \rceil$, shows a lower bound for the minimum number of switches required to reach perfect routability. A large number of additional switches per signal is needed to go from 99.9% to 100% routability for large test vector sizes. This lower bound may also indicate inefficiency in our switch placements when the number of signals is small (≤ 16) — except the lower bound is not guaranteed to be tight.

The entropy curve shows the absolute minimum number of SRAM bits that would be needed to program the switch matrix. As shown by DeHon [8], the number of bits required is $\lceil \log_2 \binom{n}{k} \rceil$.

6.2   Adding Extra Output Wires

In the previous subsection, the switch matrix was designed with exactly 24 outputs to match the size of the Altera crossbar. Next, the number of crossbar output wires were gradually increased from 24 to 48, but the crossbar is used for only up to 24 signals. The results are shown in Figure 4 for a number of different switch counts. When the number of switches is low, the routability increase from having more output wires is not significant. However, once 340 switches are reached, dramatic improvements of up to 100% can be seen when additional output wires are used. Hence, a certain minimum number of switches must be present to take advantage of the extra output wires.

In the Altera FLEX8000 architecture, there is a cost associated with having more output wires. Each additional output must be considered as an additional input to the local interconnect in the Altera LABs[2]. If the local interconnect is to remain fully connected, additional switches must be placed inside the LAB. The total number of switches (sparse crossbar + local cluster) must be considered, and is shown in Figure 5. From this graph, it can be seen that the minimum number of switches at 99.95% routability is obtained with 30 output wires and approximately 1470 switches (510 switches in the sparse crossbar and 960 inside the LAB). This is significantly more than Altera's 1104 switches, but the level of routability is also much higher.

6.3   Adding Both Switches and Wires

To examine the combined effect of adding switches and widening the output stage of the sparse matrix, see Figure 6. Three key curves are shown: the baseline architecture, which is similar to the FLEX8000 with 336 switches and 24 outputs (dotted curve), the improvement from increasing to 30 output wires, and the improvement from increasing to 30 output wires and 510 switches (solid curves). In comparison, the 24-output crossbar is shown to be less routable with the same 510 switches (lower dashed curve).

However, adding output wires forced more switches to be added to the local interconnect. To make a fair comparison, these same switches should also be added to the 24-output crossbar, for a total of 702 switches (upper dashed curve). The result is still not as effective as the crossbar with more outputs.

6.4   Summary

The results from this section indicate that, to be area-efficient, a sparse crossbar should not be used at maximum utilisation. Rather, the number of outputs should be more than the number of signals

---

[2]A FLEX8000 LAB is a completely-connected cluster of eight 4-LUTs sharing 24 inputs.
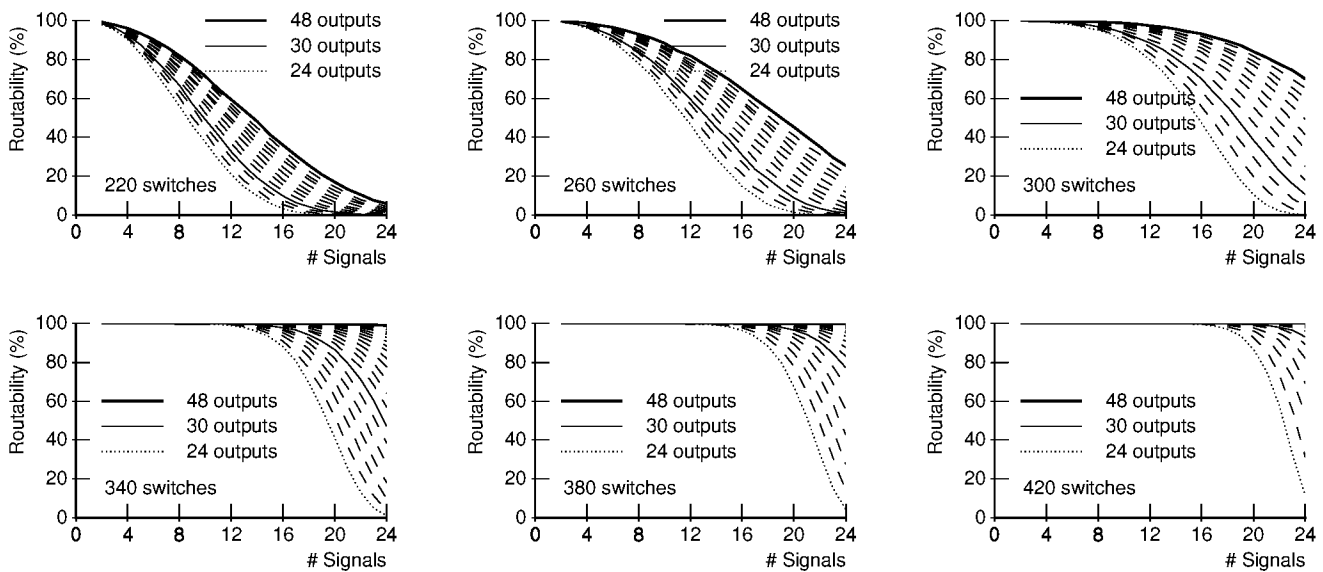
Figure 4: The effect of adding extra output wires on routability of a 168×24 crossbar. The number of switches is fixed in each graph. The curves in each graph show how routability improves as the number of output wires is increased from 24 to 48. There were 10,000 test vectors used for each signal level.
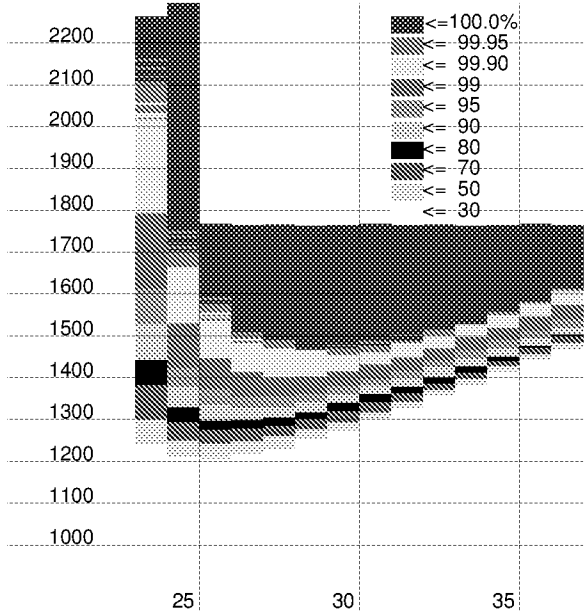
Figure 5: Effect of adding output wires on total switch counts. The number of outputs is varied along the x-axis, from 24 to 37. The total number of switches (including those in the sparse crossbar plus those in the full crossbar inside the LAB) varies along the y-axis. The shading of the graph represents the level of routability obtained for 20,000 test vectors, each containing 24 signals.
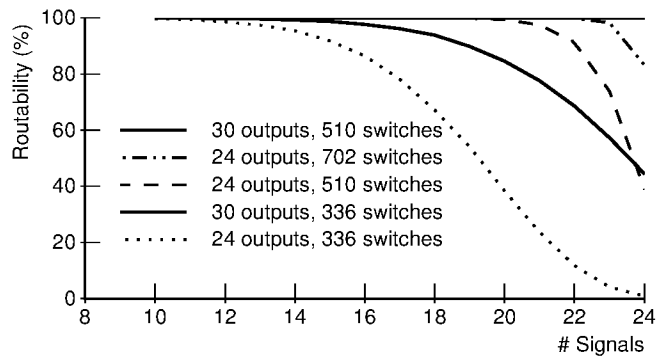
Figure 6: Combined effect of adding output wires and switches to achieve nearly 100% routability. A total of 20,000 test vectors were used for each signal level.

that are to be routed through it. As well, it is important to choose the number of switches and wires together, since a minimum number of switches are needed to benefit from the extra output wires.

7. DESIGN EXAMPLES

In the following design examples, based on architecture models in Figure 7, we searched for a number of sparse crossbar configurations which could achieve 95% or better routability and had the lowest area cost in terms of total transistors per LUT input. In counting transistors, we counted both the sparse crossbar switches and the switches of a fully-connected lower interconnect level.

Rather than use one SRAM cell and pass transistor per switch, we assumed a crossbar output is implemented using a single n-input multiplexer and encoded SRAM bits. We also assumed that each SRAM cell uses 6 transistors, and the n-input multiplexer uses a tree
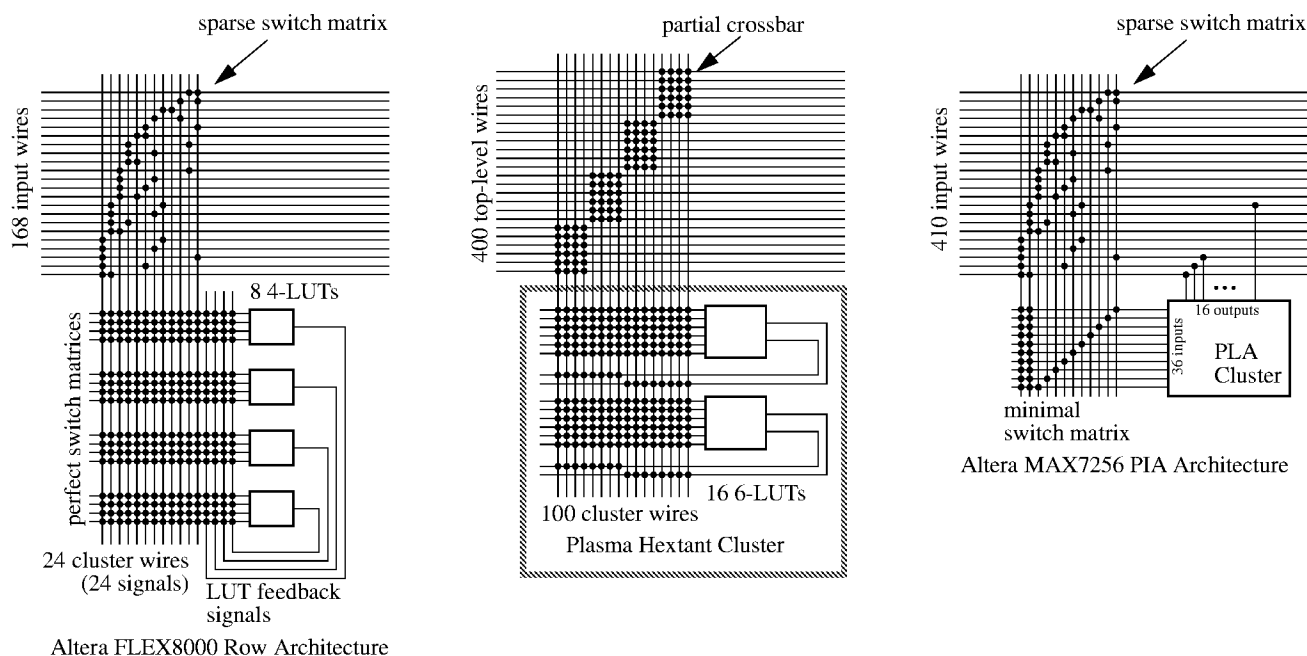
Figure 7: Models used for the Altera FLEX8000 FPGA, HP Plasma FPGA, and Altera MAX7256 CPLD architectures. The exact switch patterns shown were invented for illustration only.

of $2:1$ muxes requiring $2n - 2$ pass transistors. To keep things simple, we did not account for any additional buffering of signals or wider transistor sizes that would accompany a real design.

To find the lowest-area configuration, we explored a variety of switch densities and wider-than-required output stages. Designs with 20% or more transistors than the baseline architecture were immediately rejected. The baseline architecture chosen assumed two switches per input wire and a fully connected local interconnect (similar to Altera FLEX8000). We performed a quick routability simulation for the remaining designs using only a few (1000) test vectors, and rejected those with less than 95% routability at the largest test vector size. We retested the some of the remaining designs with a larger number of test vectors to ensure their measured performance.

7.1   Altera FLEX8000

The Altera FLEX8000 device uses a 168×24 sparse crossbar to connect the FastTrack row wires into the LAB clusters. The sparse crossbar is 1/12 populated, such that each row wire has two "opportunities" to connect into a cluster. Within the cluster, the eight 4-LUT inputs select from 24 sparse crossbar outputs and 8 LUT feedback signals using a full crossbar. This design uses approximately 129 transistors per LUT input, including the cluster interconnect. With our switch placement, the routability is excellent when there are fewer than 10 signals entering a cluster. However, if 15 or more signals enter a cluster, the routability drops below 90%. At full capacity, the routability of 24 signals drops below 1%.

Our construction techniques and search found a 168×29 sparse crossbar containing 464 switching points, or 2.7 connections per input wire. This design uses approximately 157 transistors per LUT input, including the LUT feedback connections, representing an increase of 22%. Assuming the cluster interconnect used a minimal crossbar instead of a full crossbar, our search found a 168×26 crossbar with 546 switching points, using 147 transistors per LUT input, an increase of only 14%. For a modest increase in transistor count, the improvement in routability shown in Figure 8 is dramatic.

Some other organizations found are listed in Table 1.

7.2   HP Teramac Plasma

Teramac[4], from HP Labs, is a large reconfigurable system made up of custom-designed Plasma[5] FPGAs. A full Teramac system is designed to have the capacity of about one million gates distributed over 1728 FPGA chips. An important goal in the Plasma design was to design a highly routable FPGA: to limit compile times to about an hour, placing and routing each FPGA must be done quickly (within 3 seconds). This approach meant each FPGA should be nearly 100% routable so that almost no time would be spent in ripup or repartitioning the mapped circuit.

The Plasma 2-level hierarchy comprises sixteen clusters, called *hextants*, of sixteen LUTs each. The six LUT inputs in each cluster are fully connected to 100 cluster-level wires, and the two LUT outputs are 1/2 populated. At the top level there are 400 signal wires, which must connect to the 100 cluster wires. This 400×100 partial crossbar is 1/4 populated using $10,000$ crosspoints, implying it is composed of four diagonally placed 100×25 full crossbars. Conservative measurements of the die photograph in [5] indicates that the partial crossbar switches alone consume 23% of total chip area, or 32% of the core area (excluding the I/O pads).

The Plasma chip is easy to route because the partial crossbars make it predictable to route: as long as fewer than 25 signals enter each full crossbar, it can be routed. The router need only consider which crossbar it routes to, and not the precise detailed route. Hence, there would be no need for ripup. Despite this advantage, there are few signal assignments that can satisfy the partial crossbar when more than 75 input signals are required, as shown in Figure 8.

Our sparse crossbar search found a 400×104 sparse crossbar with $1,456$ switching points, or 3.6 switches per top-level wire, for a switch density of roughly 1/28. This design uses approximately 292 transistors per LUT input, including the cluster-level interconnect (but not the LUT output switches).[3] Even though this sparse cross-

---

[3] Plasma used one 5T SRAM cell and one pass transistor per switch-
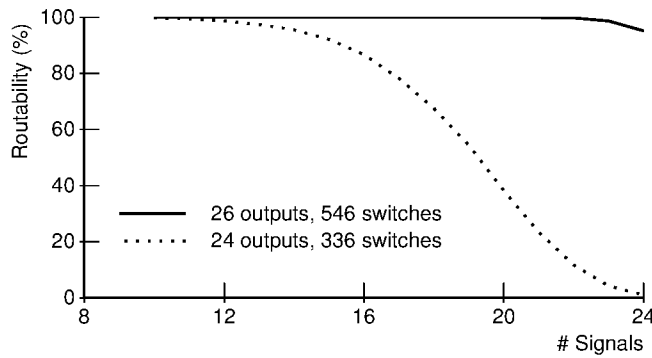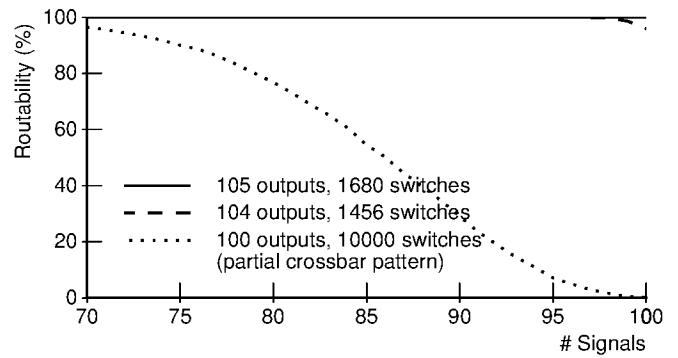
Figure 8: Routability Improvements made to the FLEX8000 and HP Plasma architectures.

Table 1: Highly routable, area-efficient sparse crossbars suitable for use in the Altera FLEX8000 family. The first group shows the number of switches required to obtain high routability with exactly 24 outputs. The second group adds a minimal crossbar between the sparse crossbar and the LAB full crossbar to reduce the sparse crossbar outputs to 24. The third group widens the full crossbar local interconnect inside the LAB to match the number of sparse crossbar outputs. All transistor counts include the local interconnect. Transistor counts in parentheses indicate that further reduction is possible if minimal crossbars are used within the LAB instead.

Crossbar Size	Switching Points	Transistors	Routability
Provide exactly 24 outputs from the sparse crossbar.			
168 × 24	336	4144	1.0%
168 × 24	1008	5776	98.6%
Reduce to exactly 24 outputs using an additional minimal crossbar.			
168 × 26 × 24	546 + 72	5468	96.1%
168 × 25 × 24	700 + 48	5492	98.2%
168 × 27 × 24	567 + 96	5650	99.2%
168 × 29 × 24	464 + 144	5694	98.6%
168 × 30 × 24	450 + 168	5800	98.2%
Provide more than 24 outputs, increase cluster interconnect.			
168 × 29	464	5022 (4830)	98.6%
168 × 30	450	5080 (4888)	98.2%
168 × 26	546	5084 (4700)	96.1%
168 × 31	434	5134 (4942)	98.6%
168 × 27	567	5218 (4834)	99.2%
168 × 25	700	5300 (4916)	98.2%

Table 2: Highly routable, area-efficient sparse crossbars suitable for use in HP Plasma FPGAs. Total transistor counts include the local cluster interconnect to choose LUT inputs, but not the LUT outputs.

Crossbar Size	Switching Points	Transistors	Routability
Partial crossbar switch pattern used by HP.			
400 × 100	10000	47040	0.4%
Sparse crossbar patterns found.			
400 × 104	1456	28048	95.9%
400 × 105	1365	28080	95.1%
400 × 103	1648	28218	98.8%
400 × 106	1378	28320	98.6%
400 × 107	1284	28346	96.9%
400 × 108	1296	28584	99.3%
400 × 109	1199	28604	97.1%
400 × 105	1680	28710	100.0%
400 × 102	1734	28788	96.6%
400 × 108	1404	28800	99.9%

bar contains nearly 1/7 the number of switching points of Plasma, it has significantly improved routability. It can route over 95% of vectors containing 100 input signals, whereas Plasma can route less than 1%. Given this new switch pattern, a router would have even higher assurances it could route each Plasma chip independently.

Alternatively, a sparse crossbar of size 400 × 105 with 1,680 switching points, or 299 transistors per LUT input, can be constructed which routes over 99.9% of the test vectors. A few other organizations found are listed in Table 2.

7.3   Altera MAX7000

The Altera MAX7256 CPLD has 2-levels of hierarchy, where the top level contains multiple sparse $n \times 36$ crossbars. The value of n used in the MAX7256 device is not known, but it is probably not more than 410 (one wire for each macrocell output and I/O pin). Here, we shall assume $n = 410$.

We have noted before that providing more than 36 crossbar outputs is necessary to keep area low while obtaining high routability. Rather than increase the number of inputs to the product-term AND planes by this amount, we have chosen to connect the crossbar outputs to a minimal crossbar, which can perfectly select any 36 of these signals for the AND plane. In this case, this minimal crossbar is area-efficient because it is close to being square, so it requires

ing point, but we are assuming a different implementation here.

Table 3: Highly routable, area-efficient sparse crossbars suitable for use in the Altera MAX7256 CPLD. Total transistor counts include the minimal crossbar selector, if appropriate, but not the product term array.

Crossbar Size	Switching Points	Transistors	Routability
1-level, provide exactly 36 outputs from the sparse crossbar.			
410×36	$2448 + 0$	6336	96.40%
410×36	$2952 + 0$	7344	99.70%
2-levels, reduce to exactly 36 outputs using a minimal crossbar.			
$410 \times 43 \times 36$	$1161 + 288$	4678	97.2%
$410 \times 42 \times 36$	$1218 + 252$	4692	97.6%
$410 \times 41 \times 36$	$1271 + 216$	4698	97.8%
$410 \times 39 \times 36$	$1443 + 144$	4860	96.7%
$410 \times 45 \times 36$	$1080 + 360$	4932	96.2%
$410 \times 38 \times 36$	$1558 + 108$	4984	96.2%
$410 \times 40 \times 36$	$1360 + 180$	5016	97.2%
$410 \times 43 \times 36$	$1333 + 288$	5022	100.0%

few switching points.

Assuming an SRAM and mux-based crossbar implementation, the best organization found used a 410×43 sparse crossbar containing 1161 switching points, or about 2.8 switches per input. This crossbar is over 97% routable when 36 signals are required. The minimal crossbar requires an additional 288 switching points. A total of 4678 transistors would be required to construct the two switching stages, or 129 transistors per output. Alternatively, a 410×43 sparse crossbar containing 1333 switching points was found to achieve over 99.9% routability. This system used 5022 transistors, or 139 transistors per output.

Both of these 2-level organizations use significantly fewer transistors (and switching points) than an architecture with only one sparse crossbar containing exactly 36 outputs. The best 2-level organization contains 26% fewer transistors and 40% fewer switching points than the best 1-level. A few other organizations found are shown in Table 3.

7.4   Varying FLEX8000 Cluster Size

In this section, we present the results of generating highly routable sparse crossbars for variations of the Altera FLEX8000 architecture shown in Figure 7. The goal is to understand the impact of cluster size and crossbar input size on the area efficiency of the sparse crossbar. To do this, we normalize the area based on transistors per LUT input. This accounts for the increased logic capacity of larger clusters, and allows us to directly compare the results.

We varied the cluster size, N, from between 2 and 12 LUTs, the number of top-level wires, n, were varied from 168 to 995. The maximum number of output signals required by the crossbar was set to be $3N$, which gives 24 inputs for the FLEX8000 case of $N = 8$. We also repeated these experiments with $2N + 2$ output signals, as recommended by Betz [6].

In general, it was usually possible to find multiple sparse crossbars which are both area-efficient and fit within the desired routability constraints. When multiple designs matching the criterion were found, the one with the lowest transistors per LUT input was selected. Sometimes a design could not be found, so the data point was left out of the results.

First, we examine the impact of cluster size on area as shown in the top graphs of Figure 9. For a sparse crossbar with only 168 inputs the effect of cluster size is not significant on area, but it can be seen that a cluster size between 4 and 7 gives the best efficiency. In contrast, small cluster sizes become very inefficient when the number

of crossbar inputs is increased. Selecting a cluster size of at least 8 is necessary in these cases. Letting the aspect ratio of the sparse crossbar get too large hinders the efficiency.

Next, we examine the impact of increasing the number of crossbar inputs for specific cluster sizes, as shown in the lower graphs of Figure 9. This is an orthogonal view of the same data, except some cluster sizes have been left out for clarity. From this data, we can see that the area cost for a given cluster size is a roughly linear function of the size of the crossbar. The slope of the larger cluster sizes is smaller, making them more area-efficient at larger crossbar sizes.

8.   CONCLUSIONS AND FUTURE WORK

We have shown a method for evaluating and constructing sparse crossbars. The construction technique is based on an understanding of Hall's Theorem to generate highly routable crossbars.

Routability of sparse crossbars can be improved by adding additional switches and by widening the output stage of the crossbar. The latter method was the most effective once there were enough switches to be used: approximately two per input in the case of a 168×24 crossbar. Careful evaluation using both methods is necessary to obtain optimum routability at minimum area.

We have demonstrated with a few design examples that it is beneficial to plan to underutilise the output stage of a sparse crossbar and design using the correct number of switches. In the Plasma example, only 4 additional output wires and a switch density of 1/28 was required to remain highly routable; this organization uses 75% fewer switches than Plasma, and obtains superior results. In the Altera FLEX8000 example, 5 additional output wires and a slight increase in switch density (from 1/12 to about 1/10) was needed to obtain over 95% routability. It was also found that cluster sizes between 4 and 7 give the most area-efficient interconnect in the FLEX8000.

Planning on high routability does not require an exorbitant amount of switching resources: in the examples given, the most dense switch pattern used was less than 1/10 populated.

For future work, we have tried to design and simulate cascaded sparse crossbars that depend on one another. To date, we have not been successful in generating consistently improved results with cascaded crossbars — independently optimized crossbars produce more consistent results. Also there is difficulty modeling congestion with a network flow solver, so plans are underway to integrate the construction algorithms into an actual router.

9.   REFERENCES

[1] Altera, San Jose, CA. *1996 Data Book*, 1996.

[2] Altera, San Jose, CA. *News & Views Newsletter*, August 1999.

[3] Altera. *Private communication.*, 1999.

[4] R. Amerson, R. Carter, W. Culbertson, P. Kuekes, and G. Snider. Teramac – configurable custom computing. In *IEEE Symposium on FPGA's for Custom Computing*, 1995.

[5] R. Amerson, R. Carter, W. Culbertson, P. Kuekes, and G. Snider. Plasma: An FPGA for million gate systems. In *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pages 10–16, 1996.

[6] V. Betz, J. Rose, and A. Marquardt. *Architecture and CAD for Deep-Submicron FPGAs*. Kluwer Academic Publishers, Boston, 1999.

[7] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1993.

[8] A. DeHon. Entropy, counting, and programmable interconnect. In *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pages 73–79, 1996.

[9] A. DeHon. Balancing interconnect and computation in a reconfigurable computing array. In *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pages 69–78, 1999.
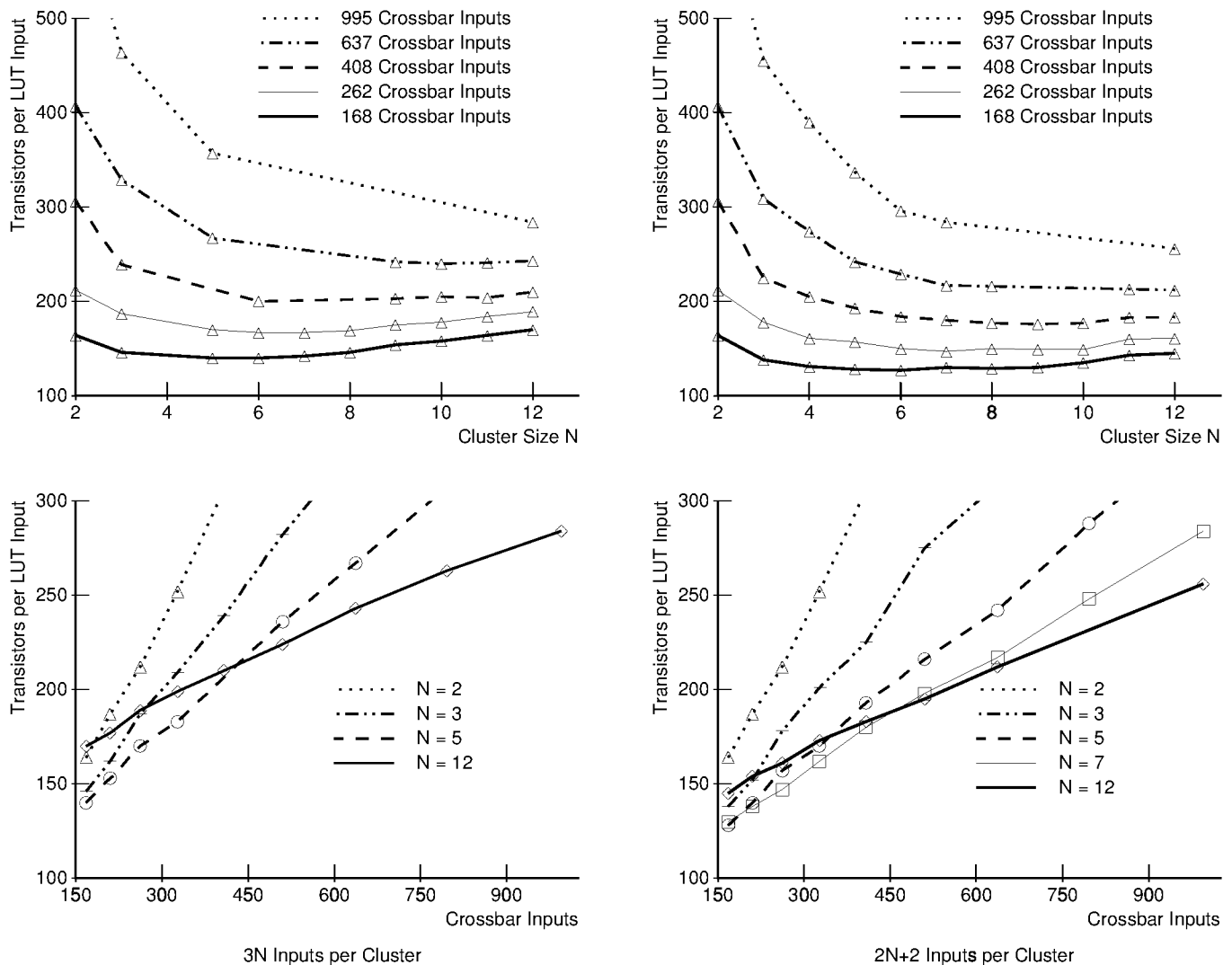
Figure 9: Effect of varying the cluster size N (upper graphs) and number of top-level inputs (lower graphs). There are $3N$ inputs per cluster on the left, $2N+2$ inputs on the right.

[10] A. El Gamal, L. A. Heinachandra, I. Shperling, and V. K. Wei. Using simulated annealing to design good codes. *IEEE Transactions on Information Theory*, 33(1):116–123, January 1987.

[11] K. Fujiyoshi, Y. Kajitani, and H. Niitsu. Design of optimum totally-perfect connection-blocks of FPGA. In *IEEE International Symposium on Circuits and Systems*, pages 221–224, May 1994.

[12] W. Guo and A. Y. Oruç. Regular sparse crossbar concentrators. *IEEE Transactions on Computers*, 47(3):363–368, March 1998.

[13] P. Hall. On representatives of subsets. *Journal of the London Mathematical Society*, 10:26–30, 1935.

[14] I. S. Honkala and P. R. J. Östergård. Applications in code design. In E. Aarts and J. Lenstra, editors, *Local Search in Combinatorial Optimization*, chapter 12. Wiley, 1997.

[15] P. Leventis. Placement algorithms and routing architecture for long-line based FPGAs. Bachelor thesis, University of Toronto, 1999.

[16] S. Nakamura and G. M. Masson. Lower bounds on crosspoints in concentrators. *IEEE Transactions on Computers*, C-31(12):1173–1179, December 1982.

[17] A. Y. Oruç and H. M. Huang. Crosspoint complexity of sparse crossbar concentrators. *IEEE Transactions on Information Theory*, 42(5):1466–1179, September 1996.

[18] J. Vargese, M. Butts, and J. Batcheller. An efficient logic emulation system. *IEEE Transactions on VLSI*, 1(2):171–174, June 1993.

Electronic Acknowledgement Receipt

EFS ID:	34486076
Application Number:	16202067
International Application Number:	
Confirmation Number:	8134
Title of Invention:	VLSI Layouts of Fully Connected Generalized Networks
First Named Inventor/Applicant Name:	Venkat Konda
Customer Number:	38139
Filer:	Venkat Konda
Filer Authorized By:	
Attorney Docket Number:	V-0070US
Receipt Date:	04-DEC-2018
Filing Date:	27-NOV-2018
Time Stamp:	21:18:37
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Information Disclosure Statement (IDS) Form (SB08)	16-202067-sb0008b.pdf	529624 ebed329005435122018c7f188950989c603a9cb7	no	2

Warnings:

Information:					
This is not an USPTO supplied IDS fillable form					
2	Non Patent Literature	Efficient-layout-of-hyperCntwks.pdf	235617 / cea3f29ab13dbe4f26e5abd98ab609adf980 6b8c	no	8

Warnings:	
Information:	

3	Non Patent Literature	fpga05-harp.pdf	261760 / e9a60a1a1e2d7b2ab393590e3109eac3ec7 8b0df	no	9

Warnings:	
Information:	

4	Non Patent Literature	Generating-2000.pdf	191395 / 72b0f7ce5f8fc1bdb81a7b829f802f2118c4c 209	no	10

Warnings:	
Information:	

Total Files Size (in bytes):	1218396

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.
National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.
New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	16/202067
Filing Date	12-4-2018
First Named Inventor	Venkat Konda
Art Unit	
Examiner Name	

Sheet	1	of	1	Attorney Docket Number	V-0070US

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Document Number — Number-Kind Code[2] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
	1	US- 8270400-b2	09-18-2012	Venkat Konda	all FIGs
	2	US- 8170040-b2	05-01-2012	Venkat Konda	all FIGs
	3	US- 8363649-b2	01-29-2013	Venkat Konda	all FIGs
	4	US- 6185220-b1	02-06-2001	Muthukrishnan et. al.	layout FIGs
	5	US- 6940308-b2	09-06-2005	Wong	layout FIGs
	6	US- 5451936	09-19-1995	Yang et. al.	layout FIGs
	7	US- 5153843	10-061992	Kenneth E. Batcher	layout FIGs
	8	US- 6018523	01-25-2000	Shimon Even	layout FIGs
	9	US-			
	10	US-			
	11	US-			
	12	US-			
	13	US-			
	14	US-			
		US-			
		US-			
		US-			
		US-			
		US-			

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document Country Code[3]-Number[4]-Kind Code[5] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages Or Relevant Figures Appear	T[6]

Examiner Signature		Date Considered	

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1] Applicant's unique citation designation number (optional). [2] See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. [3] Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). [4] For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [5] Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. [6] Applicant is to place a check mark here if English language Translation is attached.

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 (1-800-786-9199) and select option 2.

Privacy Act Statement

The Privacy Act of 1974 (P.L. 93-579) requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (i.e., GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

Electronic Acknowledgement Receipt

EFS ID:	34486221
Application Number:	16202067
International Application Number:	
Confirmation Number:	8134
Title of Invention:	VLSI Layouts of Fully Connected Generalized Networks
First Named Inventor/Applicant Name:	Venkat Konda
Customer Number:	38139
Filer:	Venkat Konda
Filer Authorized By:	
Attorney Docket Number:	V-0070US
Receipt Date:	04-DEC-2018
Filing Date:	27-NOV-2018
Time Stamp:	22:05:06
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Information Disclosure Statement (IDS) Form (SB08)	16-202067-sb0008a.pdf	252007 547e388cbd7d51710fea118814b6e22b693 f3ace	no	2

Warnings:

Information:

This is not an USPTO supplied IDS fillable form

Total Files Size (in bytes):	252007

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.
National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.
New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

PATENT COOPERATION TREATY

From the
INTERNATIONAL SEARCHING AUTHORITY

To:

VENKAT KONDA
6278, GRAND OAK WAY
SAN JOSE, CA 95135

PCT

WRITTEN OPINION OF THE
INTERNATIONAL SEARCHING AUTHORITY

(PCT Rule 43*bis*.1)

Date of mailing *(day/month/year)*	**03 SEP 2008**

Applicant's or agent's file reference	FOR FURTHER ACTION
S-0045 PCT	See paragraph 2 below

International application No.	International filing date *(day/month/year)*	Priority date *(day/month/year)*
PCT/US2008/064605	22 May 2008	25 May 2007

International Patent Classification (IPC) or both national classification and IPC
IPC(8) - H01L 25/00 (2008.04)
USPC - 326/41

Applicant
KONDA, VENKAT

---

1. This opinion contains indications relating to the following items:

   ☒ Box No. I   Basis of the opinion

   ☐ Box No. II   Priority

   ☐ Box No. III   Non-establishment of opinion with regard to novelty, inventive step and industrial applicability

   ☐ Box No. IV   Lack of unity of invention

   ☒ Box No. V   Reasoned statement under Rule 43*bis*.1(a)(i) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement

   ☐ Box No. VI   Certain documents cited

   ☐ Box No. VII   Certain defects in the international application

   ☐ Box No. VIII Certain observations on the international application

2. **FURTHER ACTION**

   If a demand for international preliminary examination is made, this opinion will be considered to be a written opinion of the International Preliminary Examining Authority ("IPEA") except that this does not apply where the applicant chooses an Authority other than this one to be the IPEA and the chosen IPEA has notified the International Bureau under Rule 66.1*bis*(b) that written opinions of this International Searching Authority will not be so considered.

   If this opinion is, as provided above, considered to be a written opinion of the IPEA, the applicant is invited to submit to the IPEA a written reply together, where appropriate, with amendments, before the expiration of 3 months from the date of mailing of Form PCT/ISA/220 or before the expiration of 22 months from the priority date, whichever expires later.

   For further options, see Form PCT/ISA/220.

3. For further details, see notes to Form PCT/ISA/220.

Name and mailing address of the ISA/US	Date of completion of this opinion	Authorized officer:
Mail Stop PCT, Attn: ISA/US Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-3201	25 August 2008	Blaine Copenheaver PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774

Form PCT/ISA/237 (cover sheet) (April 2007)

PCT/US2008/064605 03.09.2008

<table>
<tr><td rowspan="2" align="center">WRITTEN OPINION OF THE
INTERNATIONAL SEARCHING AUTHORITY</td><td>International application No.
PCT/US2008/064605</td></tr>
</table>

Box No. I   Basis of this opinion

1. With regard to the language, this opinion has been established on the basis of:

   [X] the international application in the language in which it was filed.

   [ ] a translation of the international application into _____ which is the language of a translation furnished for the purposes of international search (Rules 12.3(a) and 23.1(b)).

2. [ ] This opinion has been established taking into account the **rectification of an obvious mistake** authorized by or notified to this Authority under Rule 91 (Rule 43*bis*.1(a))

3. With regard to any **nucleotide and/or amino acid sequence** disclosed in the international application, this opinion has been established on the basis of:

   a. type of material

   [ ] _ a sequence listing

   [ ] table(s) related to the sequence listing

   b. format of material

   [ ] on paper

   [ ] in electronic form

   c. time of filing/furnishing

   [ ] contained in the international application as filed

   [ ] filed together with the international application in electronic form

   [ ] furnished subsequently to this Authority for the purposes of search

4. [ ] In addition, in the case that more than one version or copy of a sequence listing and/or table(s) relating thereto has been filed or furnished, the required statements that the information in the subsequent or additional copies is identical to that in the application as filed or does not go beyond the application as filed, as appropriate, were furnished.

5. Additional comments:

Form PCT/ISA/237 (Box No. I) (April 2007)

<div align="center">

WRITTEN OPINION OF THE
INTERNATIONAL SEARCHING AUTHORITY

</div>

International application No.
PCT/US2008/064605

Box No. V   Reasoned statement under Rule 43bis.1(a)(i) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement

1.   Statement

Novelty (N)	Claims	9-11, 13-15, 20, 26-28 30-32, 34-39, 41-43, 45-47	YES
	Claims	1-8, 12, 16-19, 21-25, 29, 33, 40, 44, 48-49	NO
Inventive step (IS)	Claims	None	YES
	Claims	1-49	NO
Industrial applicability (IA)	Claims	1-49	YES
	Claims	None	NO

2.   Citations and explanations:

Claims 1-8, 12, 16-19, 21-25, 29, 33, 40, 44, and 48-49 lack novelty under PCT Article 33(2) as being anticipated by Wong.

Regarding claim 1, Wong discloses an integrated circuit device comprising a plurality of sub-integrated circuit blocks and a routing network ('ASIC', Col. 13, Lines 4-5), and each plurality of sub-integrated circuit blocks comprising a plurality of inlet links and a plurality of outlet links ('input-output pin-pair', Col. 13, Line 25); and Said routing network interconnects anyone of said outlet link of one of said subintegrated circuit block to one or more said inlet links of one or more of said subintegrated circuit blocks ('peripheral blocks of an integrated circuit', Col. 14, Line 48); and Said routing network comprising of a plurality of stages y, starting from the lowest stage to the highest stage ('hierarchical levels', Col. 2, Lines 7-12); and Said routing network comprising a plurality of switches of size d X d , where d ≥ 2, in each said stage and each said switch of size d X d having d inlet links and d outlet links ('2x2 switch', Col. 2, Line 27); and Said each sub-integrated circuit block comprising a plurality of said switches corresponding to each said stage ('switches at first rank of hierarchy', Col. 2, Lines 34-35; Fig. 3C); and Said each sub-integrated circuit block comprising a plurality of forward connecting links connecting from switches in lower stage to switches in the immediate succeeding higher stage (Col. 2, Lines 16-22; see 'input' and 'output' links in Fig. 2B; Fig. 3A), and also comprising a plurality of backward connecting links connecting from switches in higher stage to switches in the immediate preceding lower stage (see 'upper network' and 'lower network' in Fig .3B; see 'upper' and 'lower' in Fig .3C; 'inputs' and 'outputs', Fig. 4B); and Said each sub-integrated circuit block comprising a plurality straight links in said forward connecting links from switches in lower stage to switches in the immediate succeeding higher stage and a plurality cross links in said forward connecting links from switches in lower stage to switches in the immediate succeeding higher stage ('pass (straight) mode or cross mode', Col. 5, Lines 4-13; 'cross', Fig. 2B; see 'upper network' and 'lower network' in Fig .3B; see 'upper' and 'lower' in Fig .3C; 'inputs' and 'outputs', Fig. 4B). and further comprising a plurality of straight links in said backward connecting links from switches in higher stage to switches in the immediate preceding lower stage and a plurality of cross links in said backward connecting links from switches in higher stage to switches in the immediate preceding lower stage ('pass (straight) mode or cross mode', Col. 5, Lines 4-13; 'cross', Fig. 2B; see 'upper network' and 'lower network' in Fig .3B; see 'upper' and 'lower' in Fig .3C; 'inputs' and 'outputs', Fig. 4B).

Regarding claim 2, Wong discloses said all straight links are connecting from switches in each said sub-integrated circuit block are connecting to switches in the same said sub-integrated circuit block (Fig. 13A); and said all cross links are connecting as either vertical or horizontal links between switches in two different said sub-integrated circuit blocks (Fig. 13A).

Regarding claim 3, Wong discloses said plurality of sub- integrated circuit blocks arranged in a two-dimensional grid ('width and height dimensions', Col. 12, Lines 48-51).

Regarding claim 4, Wong discloses said cross links in succeeding stages are connecting as alternative vertical and horizontal links between switches in said sub-integrated circuit blocks (Col. 2, Line 7-12; Fig. 10, Fig. 14B).

Regarding claim 5, Wong discloses said cross links from switches in a stage in one of said sub-integrated circuit blocks are connecting to switches in the succeeding stage in another of said sub-integrated circuit blocks so that said cross links are either vertical links or horizontal and vice versa, and hereinafter such cross links are "shuffle exchange links" ('rearrangeable interconnection network', Col. 1, Line 61 – Col. 2, Line 6; 'rearrangeable', Col. 4, Lines 12-17).

Regarding claim 6, Wong discloses all horizontal shuffle exchange links between switches in any two corresponding said succeeding stages are substantially of equal length and said vertical shuffle exchange links between switches in any two corresponding said succeeding stages are substantially of equal length in the entire said integrated circuit device ('original Benes network' and 'every route must travel through all levels', Col. 7, Lines 26-27; Col. 11, Lines 23-25).

Regarding claim 7, Wong discloses the shortest horizontal shuffle exchange links are connecting at the lowest stage and between switches in two nearest neighboring said sub-integrated circuit blocks, and length of the horizontal shuffle exchange links is doubled in each succeeding stage ('shorter routes', Col. 7, Lines 23-25); and the shortest vertical shuffle exchange links are connecting at the lowest stage and between switches in two nearest neighboring said sub-integrated circuit blocks, and length of the vertical shuffle exchange links is doubled in each succeeding stage ("length = 2*(log2 N)", Col. 11, Lines 23-25).

See supplemental page.

Form PCT/ISA/237 (Box No. V) (April 2007)

<table>
<tr><td rowspan="2" style="text-align:center">WRITTEN OPINION OF THE
INTERNATIONAL SEARCHING AUTHORITY</td><td>International application No.</td></tr>
<tr><td>PCT/US2008/064605</td></tr>
</table>

Supplemental Box

In case the space in any of the preceding boxes is not sufficient.
Continuation of:
Box V
2. Citations and explanations

Regarding claim 8, Wong discloses wherein y ≥ (log2 N) so that the
length of the horizontal shuffle exchange links in the highest stage is equal to half the size of the horizontal size of said two dimensional grid of sub-integrated circuit blocks and the length of the vertical shuffle exchange links in the highest stage is equal to half the size of the vertical size of said two dimensional grid of sub-integrated circuit blocks ("length = 2*(log2 N)", Col. 11, Lines 23-25).

Regarding claim 12, Wong discloses wherein y ≥ (log2, N) so that the length of the horizontal shuffle exchange links in the highest stage is equal to half the size of the horizontal size of said two dimensional grid of sub-integrated circuit blocks and the length of the vertical shuffle exchange links in the highest stage is equal to half the size of the vertical size of said two dimensional grid of sub-integrated circuit blocks ("length = 2*(log2 N)", Col. 11, Lines 23-25), and said each sub-integrated circuit block further comprising a plurality of U-turn links within switches in each of said stages in each of said sub-integrated circuit blocks (see u-turn links, Figs. 5 and 6B).

Regarding claim 16, Wong discloses said horizontal and vertical links are implemented on two or more metal layers ('CMOS integrated circuit', Col. 5, Lines 22-25).

Regarding claim 17, Wong discloses wherein said switches comprising active and reprogrammable cross points and said each cross point is programmable by an SRAM cell or a Flash Cell ('SRAM-based FPGA', Col. 2, Lines 25-26, 'FPGA (Field Programmable Gate Array)', Col. 1, Lines 14-17).

Regarding claim 18, Wong discloses said sub-integrated circuit blocks are of equal die size ('sub-networks of equal size', Col. 6, Lines 10-11).

Regarding claim 19, Wong discloses said sub-integrated circuit blocks are Lookup Tables (hereinafter "LUTs") and said integrated circuit device is a field programmable gate array (FPGA) device or field programmable gate array (FPGA) block embedded in another integrated circuit device ('look-up tables', Col. 8, Line 51; 'FPGA (Field Programmable Gate Array)', Col. 1, Lines 14-17).

Regarding claim 21, Wong discloses said sub-integrated circuit blocks comprising any arbitrary hardware logic or memory circuits (Col. 14, Lines 44-51).

Regarding claim 22, Wong discloses said switches comprising active one-time programmable cross points and said integrated circuit device is a mask programmable gate array (MPGA) device or a structured ASIC device ('MPGA', Col. 14, Lines 29-31; 'ASIC', Col. 13, Lines 4-5).

Regarding claim 23, Wong discloses said switches comprising passive cross points or just connection of two links or not and said integrated circuit device is a Application Specific Integrated Circuit (ASIC) device ('logic cells', Col. 1, Lines 23-25; 'ASIC', Col. 13, Lines 4-5).

Regarding claim 24, Wong discloses said sub-integrated circuit blocks further recursively comprise one or more super-sub-integrated circuit blocks and a sub-routing network ('one or more sub-networks', Col. 6, Lines 7-9).

Regarding claim 25, Wong discloses said all horizontal shuffle exchange links between switches in any two corresponding said succeeding stages are of different length and said vertical shuffle exchange links between switches in any two corresponding said succeeding stages are of different length and y ≥ (log 2 N) ("length = 2*(log2 N)", Col. 11, Lines 23-25).

Regarding claim 29, Wong discloses said all horizontal shuffle exchange links between switches in any two corresponding said succeeding stages are of different length and said vertical shuffle exchange links between switches in any two corresponding said succeeding stages are of different length and y ≥ (log2, N) (length = 2*(log2 N)", Col. 11, Lines 23-25) and said each sub-integrated circuit block further comprising a plurality of U-turn links within switches in each of said stages in each of said sub-integrated circuit blocks (see u-turn links, Figs. 5 and 6B).

Regarding claim 33, Wong discloses said straight links connecting from switches in each said sub-integrated circuit block are connecting to switches in the same said sub-integrated circuit block; and said cross links are connecting as vertical or horizontal or diagonal links between two different said sub-integrated circuit blocks (see link arrangement in Fig. 3B).

Regarding claim 40, Wong discloses said all horizontal shuffle exchange links between switches in any two corresponding said succeeding stages are of different length and said vertical shuffle exchange links between switches in any two corresponding said succeeding stages are of different length and y ≥ (log2, N) (length = 2*(log2 N)", Col. 11, Lines 23-25).

Regarding claim 44, Wong discloses said all horizontal shuffle exchange links between switches in any two corresponding said succeeding stages are of different length and said vertical shuffle exchange links between switches in any two corresponding said succeeding stages are of different length and y ≥ (log 2 N), and said each sub-integrated circuit block further comprising a plurality of U-turn links within switches in each of said stages in each of said sub-integrated circuit blocks (length = 2*(log2 N)", Col. 11, Lines 23-25; see u-turn links, Figs. 5 and 6B).

Regarding claim 48, Wong discloses said plurality of forward connecting links use a plurality of buffers to amplify signals driven through them and said plurality of backward connecting links use a plurality of buffers to amplify signals driven through them; and said buffers can be inverting or non-inverting buffers ('buffers', Col. 10, Lines 59-64).

See supplemental page.

Form PCT/ISA/237 (Supplemental Box) (April 2007)

PCT/US2008/064605 03.09.2008

WRITTEN OPINION OF THE INTERNATIONAL SEARCHING AUTHORITY	International application No. PCT/US2008/064605

Supplemental Box

In case the space in any of the preceding boxes is not sufficient.
Continuation of:
Box V
2. Citations and explanations

Regarding claim 49, Wong discloses said wherein said all switches of size d X d are either fully populated or partially populated ('populated and depopulated cells', Col. 3, Lines 34-41).

Claims 9-11, 13-15, 20, 26-28, 30-32, 34-39, 41-43, and 45-47 lack an inventive step under PCT Article 33(3) as being obvious over Wong in view of Wu et al., hereinafter Wu.

Regarding claim 9, Wong discloses a Benes network wherein d =2 and there is only one switch in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there is only one switch in each said stage in each said subintegrated circuit block connecting said backward connecting links (Col. 2, Lines 27-30; '2x2 Benes network', Col. 5, Lines 26-31), but does not disclose said routing network is rearrangeably nonblocking for unicast Benes network with full bandwidth.
Wu discloses a rearrangeable non-blocking switch (Abstract) and a routing network being rearrangeably nonblocking for unicast Benes network with full bandwidth ('rearrangeably nonblocking for unicast traffic', Col. 2, Lines 52-54).
-Therefore it would have been obvious to one ordinarily skilled in the art at the time of the invention to supplement the teachings of Wong and have routing network being rearrangeably nonblocking for unicast Benes network with full bandwidth, in order to satisfy specific routing requirements.

Regarding claim 10, Wong discloses d = 2 and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links ('8x8 Benes network' (each switch has two logic cells that are 2x2, hence 8x8 is 4 switches), Col. 2, Lines 31-33), but does not disclose said routing network is strictly nonblocking for unicast Benes network and rearrangeably nonblocking for arbitrary fan-out multicast Benes network with full bandwidth.
Wu discloses a rearrangeable non-blocking switch (Abstract) and a routing network being strictly nonblocking for unicast Benes network (Col. 2, Lines 9-11, see Fig. 9) and rearrangeably nonblocking for arbitrary fan-out multicast Benes network with full bandwidth (Col. 4, Lines 6-7; see 'fanouts', Fig. 7B).
Therefore it would have been obvious to one ordinarily skilled in the art at the time of the invention to supplement the teachings of Wong and have a routing network be strictly nonblocking for unicast Benes and rearrangeably nonblocking for arbitrary fan-out multicast Benes network with full bandwidth, as disclosed by Wu, in order to satisfy specific routing requirements.

Regarding claim 11, Wong discloses d =2 and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links ('8x8 Benes network', Col. 2, Lines 31-33), but does not disclose said routing network is strictly nonblocking for arbitrary fan-out multicast Benes network with full bandwidth.
Wu discloses a rearrangeable non-blocking switch (Abstract) and a routing network strictly nonblocking for arbitrary fan-out multicast Benes network with full bandwidth (Col. 2, Lines 8-11; see 'fanouts', Fig. 7B).
Therefore it would have been obvious to one ordinarily skilled in the art at the time of the invention to supplement the teachings of Wong and have a routing network be strictly nonblocking for arbitrary fan-out multicast Benes network with full bandwidth, as disclosed by Wu, in order to satisfy specific routing requirements.

Regarding claim 13, Wong discloses d = 2 and there is only one switch in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there is only one switch in each said stage in each said subintegrated circuit block connecting said backward connecting links ('8x8 Benes network', Col. 2, Lines 31-33) and a routing network being a butterfly fat tree network 7, but does not disclose said routing network is rearrangeably nonblocking for unicast traffic.
Wu discloses a rearrangeable non-blocking switch (Abstract) and a routing network being rearrangeably nonblocking for unicast traffic ('rearrangeably nonblocking for unicast traffic', Col. 2, Lines 52-54).
Therefore it would have been obvious to one ordinarily skilled in the art at the time of the invention to supplement the teachings of Wong and have a routing network be rearrangeably nonblocking for unicast traffic, as disclosed by Wu, in order to satisfy specific routing requirements.

Regarding claim 14, Wong discloses wherein d =2 and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links ('8x8 Benes network', Col. 2, Lines 31-33) and a routing network being a butterfly fat tree network (Col. 13, Lines 23-24), but does not disclose said routing network is strictly nonblocking for unicast traffic and rearrangeably nonblocking for arbitrary fan-out multicast traffic.
Wu discloses a rearrangeable non-blocking switch (Abstract) and a routing network being strictly nonblocking for unicast traffic (Col. 2, Lines 9-11, see Fig. 9) and rearrangeably nonblocking for arbitrary fan-out multicast Benes network with full bandwidth (Col. 4, Lines 6-7; see 'fanouts', Fig. 7B).
Therefore it would have been obvious to one ordinarily skilled in the art at the time of the invention to supplement the teachings of Wong and have a routing network be strictly nonblocking for unicast Benes and rearrangeably nonblocking for arbitrary fan-out multicast Benes network with full bandwidth, as disclosed by Wu, in order to satisfy specific routing requirements.

See supplemental page.

Form PCT/ISA/237 (Supplemental Box) (April 2007)

PCT/US2008/064605 03.09.2008

WRITTEN OPINION OF THE **INTERNATIONAL SEARCHING AUTHORITY**	International application No. PCT/US2008/064605

Supplemental Box

In case the space in any of the preceding boxes is not sufficient.
Continuation of:
Box 5
2. Citations and explanations

Regarding claim 15, Wong discloses wherein d =2 and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links ('8x8 Benes network', Col. 2, Lines 31-33) and a routing network being a butterfly fat tree network with full-bandwidth (Col. 13, Lines 23-24), but does not disclose said routing network is strictly nonblocking for arbitrary fan-out multicast traffic.
Wu discloses a rearrangeable non-blocking switch (Abstract) wherein a routing network is strictly nonblocking for arbitrary fan-out (Col. 2, Lines 8-11; see 'fanouts', Fig. 7B).
Therefore it would have been obvious to one ordinarily skilled in the art at the time of the invention to supplement the teachings of Wong and have a routing network be strictly nonblocking for arbitrary fan-out multicast Benes network with full bandwidth, as disclosed by Wu, in order to satisfy specific routing requirements.

Regarding claim 20, Wong discloses said sub-integrated circuit blocks are AND or OR gates and said integrated circuit device is a programmable logic device (PLD) ('logic gates', Col. 7, Line 36; 'programmable gate arrays', Col. 14, Lines 29-31).

Regarding claim 26, Wong discloses d =2 and there is only one switch in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there is only one switch in each said stage in each said subintegrated circuit block connecting said backward connecting links and said routing network is generalized with full bandwidth (Col. 2, Lines 27-30; '2x2 Benes network', Col. 5, Lines 26-31; 'networks can be generalized', Col. 6, Lines 7-9), but does not disclose said routing network is rearrangeably nonblocking for unicast traffic.
Wu discloses a rearrangeable non-blocking switch (Abstract) and a routing network being rearrangeably nonblocking for unicast traffic ('rearrangeably nonblocking for unicast traffic', Col. 2, Lines 52-54).
Therefore it would have been obvious to one ordinarily skilled in the art at the time of the invention to supplement the teachings of Wong and have a routing network be rearrangeably nonblocking for unicast traffic, as disclosed by Wu, in order to satisfy specific routing requirements.

Regarding clam 27, Wong discloses d =2 and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links ('8x8 Benes network' (each switch has two logic cells that are 2x2, hence 8x8 is 4 switches), Col. 2, Lines 31-33), and said routing network being a generalized network ('networks can be generalized', Col. 6, Lines 7-9), but does not disclose said routing network is strictly nonblocking for unicast generalized multi-stage network and rearrangeably nonblocking for arbitrary fan-out multicast generalized multi-stage network with full bandwidth.
Wu discloses a rearrangeable non-blocking switch (Abstract) and a routing network being rearrangeably nonblocking for arbitrary fan-out multicast multi-stage network with full bandwidth (Col. 4, Lines 6-7; see 'fanouts', Fig. 7B).
Therefore it would have been obvious to one ordinarily skilled in the art at the time of the invention to supplement the teachings of Wong and have a routing network be rearrangeably nonblocking for arbitrary fan-out multicast generalized multi-stage network with full bandwidth, as disclosed by Wu, in order to satisfy specific routing requirements.

Regarding claim 28, Wong discloses d = 2 and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is a generalized routing network ('8x8 Benes network' (each switch has two logic cells that are 2x2, hence 8x8 is 4 switches), Col. 2, Lines 31-33; 'networks can be generalized', Col. 6, Lines 7-9), but does not disclose said routing network is strictly nonblocking for arbitrary fan-out multicast multistage.
Wu discloses a rearrangeable non-blocking switch (Abstract) and a routing network strictly nonblocking for arbitrary fan-out multicast multistage (Col. 2, Lines 8-11; see 'fanouts', Fig. 7B).
Therefore it would have been obvious to one ordinarily skilled in the art at the time of the invention to supplement the teachings of Wong and have a routing network be strictly nonblocking for arbitrary fan-out multicast multistage network with full bandwidth, as disclosed by Wu, in order to satisfy specific routing requirements.

Regarding claim 30, Wong discloses d = 2 and there is only one switch in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there is only one switch in each said stage in each said subintegrated circuit block connecting said backward connecting links and said routing network is a generalized butterfly fat tree network with full bandwidth (Col. 2, Lines 27-30; '2x2 Benes network', Col. 5, Lines 26-31; 'networks can be generalized', Col. 6, Lines 7-9; 'butterfly pattern', Col. 13, Lines 23-24), but does not disclose said routing network is rearrangeably nonblocking for unicast traffic.
Wu discloses a rearrangeable non-blocking switch (Abstract) and a routing network being rearrangeably nonblocking for unicast traffic ('rearrangeably nonblocking for unicast traffic', Col. 2, Lines 52-54).
Therefore it would have been obvious to one ordinarily skilled in the art at the time of the invention to supplement the teachings of Wong and have a routing network be rearrangeably nonblocking for unicast traffic, as disclosed by Wu, in order to satisfy specific routing requirements.

See supplemental page.

Form PCT/ISA/237 (Supplemental Box) (April 2007)

VENKAT KONDA EXHIBIT 2005
PCT/US2008/064605 03.09.2008

WRITTEN OPINION OF THE INTERNATIONAL SEARCHING AUTHORITY	International application No. PCT/US2008/064605

Supplemental Box

In case the space in any of the preceding boxes is not sufficient.
Continuation of:
Box V
2. Citations and explanations

Regarding claim 31, Wong discloses d = 2 and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network being a generalized butterfly fat tree network with full-bandwidth ('8x8 Benes network' (each switch has two logic cells that are 2x2, hence 8x8 is 4 switches), Col. 2, Lines 31-33; 'networks can be generalized', Col. 6, Lines 7-9; 'butterfly pattern', Col. 13, Lines 23-24), but does not disclose said routing network is strictly nonblocking for unicast traffic and rearrangeably nonblocking for arbitrary fan-out multicast traffic.
Wu discloses a rearrangeable non-blocking switch (Abstract) and a routing network being strictly nonblocking for unicast traffic (Col. 2, Lines 9-11, see Fig. 9) and rearrangeably nonblocking for arbitrary fan-out multicast traffic (Col. 4, Lines 6-7; see 'fanouts', Fig. 7B).
Therefore it would have been obvious to one ordinarily skilled in the art at the time of the invention to supplement the teachings of Wong and have a routing network be strictly nonblocking for unicast traffic and rearrangeably nonblocking for arbitrary fan-out multicast traffic, as disclosed by Wu, in order to satisfy specific routing requirements.

Regarding claim 32, Wong discloses wherein d = 2 and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network generalized butterfly fat tree network with full bandwidth ('8x8 Benes network' (each switch has two logic cells that are 2x2, hence 8x8 is 4 switches), Col. 2, Lines 31-33; 'networks can be generalized', Col. 6, Lines 7-9), but does not disclose said routing network is strictly nonblocking for arbitrary fan-out multicast traffic.
Wu discloses a rearrangeable non-blocking switch (Abstract) and a routing network strictly nonblocking for arbitrary fan-out multicast traffic (Col. 2, Lines 8-11; see 'fanouts', Fig. 7B).
Therefore it would have been obvious to one ordinarily skilled in the art at the time of the invention to supplement the teachings of Wong and have a routing network be strictly nonblocking for arbitrary fan-out multicast traffic, as disclosed by Wu, in order to satisfy specific routing requirements.

Regarding claim 34, Wong discloses d = 4 ('plurality of input terminals and a number of output terminals', claim 1) and there is only one switch in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there is only one switch in each said stage in each said subintegrated circuit block connecting said backward connecting links and said routing network multi-link Benes network with full bandwidth (Col. 2, Lines 27-30; '2x2 Benes network', Col. 5, Lines 26-31), but does not disclose said routing network is rearrangeably nonblocking for unicast traffic.
Wu discloses a rearrangeable non-blocking switch (Abstract) and a routing network being rearrangeably nonblocking for unicast traffic ('rearrangeably nonblocking for unicast traffic', Col. 2, Lines 52-54).
Therefore it would have been obvious to one ordinarily skilled in the art at the time of the invention to supplement the teachings of Wong and have a routing network be rearrangeably nonblocking for unicast traffic, as disclosed by Wu, in order to satisfy specific routing requirements.

Regarding claim 35, Wong discloses d =4 ('plurality of input terminals and a number of output terminals', claim 1) and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is a multi-link Benes network ('8x8 Benes network' (each switch has two logic cells that are 2x2, hence 8x8 is 4 switches), Col. 2, Lines 31-33; Col. 6, Lines 7-9), but does not disclose said routing network is strictly nonblocking for unicast traffic and rearrangeably nonblocking for arbitrary fan-out multicast traffic.
Wu discloses a rearrangeable non-blocking switch (Abstract) and a routing network being strictly nonblocking for unicast traffic (Col. 2, Lines 9-11, see Fig. 9) and rearrangeably nonblocking for arbitrary fan-out multicast traffic (Col. 4, Lines 6-7; see 'fanouts', Fig. 7B).
Therefore it would have been obvious to one ordinarily skilled in the art at the time of the invention to supplement the teachings of Wong and have a routing network be strictly nonblocking for unicast traffic and rearrangeably nonblocking for arbitrary fan-out multicast traffic, as disclosed by Wu, in order to satisfy specific routing requirements.

Regarding claim 36, Wong discloses d =4 ('plurality of input terminals and a number of output terminals', claim 1) and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network multi-link Benes network with full bandwidth ('8x8 Benes network' (each switch has two logic cells that are 2x2, hence 8x8 is 4 switches), Col. 2, Lines 31-33; Col. 6, Lines 7-9), but does not disclose said routing network is strictly nonblocking for arbitrary fan-out multicast traffic.
Wu discloses a rearrangeable non-blocking switch (Abstract) and a routing network being rearrangeably nonblocking for arbitrary fan-out multicast traffic (Col. 4, Lines 6-7; see 'fanouts', Fig. 7B).
Therefore it would have been obvious to one ordinarily skilled in the art at the time of the invention to supplement the teachings of Wong and have a routing network be rearrangeably nonblocking for arbitrary fan-out multicast traffic, as disclosed by Wu, in order to satisfy specific routing requirements.

See supplemental page.

Form PCT/ISA/237 (Supplemental Box) (April 2007)

PCT/US2008/064605 03.09.2008

<table>
<tr><td>WRITTEN OPINION OF THE
INTERNATIONAL SEARCHING AUTHORITY</td><td>International application No.
PCT/US2008/064605</td></tr>
</table>

Supplemental Box

In case the space in any of the preceding boxes is not sufficient.
Continuation of:
Box V
2. Citations and explanations

Regarding claim 37, Wong discloses d = 4 ('plurality of input terminals and a number of output terminals', claim 1) and there is only one switch in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there is only one switch in each said stage in each said subintegrated circuit block connecting said backward connecting links and said routing network is a butterfly fat tree network with full bandwidth (Col. 2, Lines 27-30; Col. 2, Lines 31-33; 'butterfly pattern', Col. 13, Lines 23-24), but does not disclose said routing network is rearrangeably nonblocking for unicast traffic.
Wu discloses a rearrangeable non-blocking switch (Abstract) and a routing network being rearrangeably nonblocking for unicast traffic ('rearrangeably nonblocking for unicast traffic', Col. 2, Lines 52-54).
Therefore it would have been obvious to one ordinarily skilled in the art at the time of the invention to supplement the teachings of Wong and have a routing network be rearrangeably nonblocking for unicast traffic, as disclosed by Wu, in order to satisfy specific routing requirements.

Regarding claim 38, Wong discloses wherein d = 4 ('plurality of input terminals and a number of output terminals', claim 1) and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links ('8x8 Benes network' (each switch has two logic cells that are 2x2, hence 8x8 is 4 switches), Col. 2, Lines 31-33; Col. 6, Lines 7-9), and said routing network is a multi-link butterfly fat tree network (Col. 2, Lines 27-30; Col. 2, Lines 31-33; 'butterfly pattern', Col. 13, Lines 23-24), but does not disclose said routing network is strictly nonblocking for unicast and rearrangeably nonblocking for arbitrary fan-out multicast traffic.
Wu discloses a rearrangeable non-blocking switch (Abstract) and a routing network being strictly nonblocking for unicast traffic (Col. 2, Lines 9-11, see Fig. 9) and rearrangeably nonblocking for arbitrary fan-out multicast traffic (Col. 4, Lines 6-7; see 'fanouts', Fig. 7B).
Therefore it would have been obvious to one ordinarily skilled in the art at the time of the invention to supplement the teachings of Wong and have a routing network be strictly nonblocking for unicast traffic and rearrangeably nonblocking for arbitrary fan-out multicast traffic, as disclosed by Wu, in order to satisfy specific routing requirements.

Regarding claim 39, Wong discloses d = 4 ('plurality of input terminals and a number of output terminals', claim 1) and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is a multi-link butterfly fat tree network with full bandwidth ('8x8 Benes network' (each switch has two logic cells that are 2x2, hence 8x8 is 4 switches; 'butterfly pattern', Col. 13, Lines 23-24), Col. 2, Lines 31-33; Col. 6, Lines 7-9), but does not discloses said routing network is strictly nonblocking for arbitrary fan-out multicast.
Wu discloses a rearrangeable non-blocking switch (Abstract) and a routing network strictly nonblocking for arbitrary fan-out multicast traffic (Col. 2, Lines 8-11; see 'fanouts', Fig. 7B).
Therefore it would have been obvious to one ordinarily skilled in the art at the time of the invention to supplement the teachings of Wong and have a routing network be strictly nonblocking for arbitrary fan-out multicast traffic, as disclosed by Wu, in order to satisfy specific routing requirements.

Regarding claim 41, Wong discloses d = 4 ('plurality of input terminals and a number of output terminals', claim 1) and there is only one switch in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there is only one switch in each said stage in each said subintegrated circuit block connecting said backward connecting links (Col. 2, Lines 27-30; Col. 2, Lines 31-33) and said routing network is a generalized multi-link multi-stage network with full bandwidth ('networks can be generalized', Col. 6, Lines 7-9), but does not disclose said routing network is rearrangeably nonblocking for unicast traffic.
Wu discloses a rearrangeable non-blocking switch (Abstract) and a routing network being rearrangeably nonblocking for unicast traffic ('rearrangeably nonblocking for unicast traffic', Col. 2, Lines 52-54).
Therefore it would have been obvious to one ordinarily skilled in the art at the time of the invention to supplement the teachings of Wong and have a routing network be rearrangeably nonblocking for unicast traffic, as disclosed by Wu, in order to satisfy specific routing requirements.

See supplemental page.

Form PCT/ISA/237 (Supplemental Box) (April 2007)

PCT/US2008/064605 03.09.2008

WRITTEN OPINION OF THE **INTERNATIONAL SEARCHING AUTHORITY**	International application No. PCT/US2008/064605

Supplemental Box

In case the space in any of the preceding boxes is not sufficient.
Continuation of:

Box V
2. Citations and explanations

Regarding claim 42, Wong discloses d =4 ('plurality of input terminals and a number of output terminals', claim 1) and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links ('8x8 Benes network' (each switch has two logic cells that are 2x2, hence 8x8 is 4 switches), Col. 2, Lines 31-33; Col. 6, Lines 7-9) and said routing network is a generalized multi-link multi-stage network ('networks can be generalized', Col. 6, Lines 7-9), but does not disclose said routing network is strictly nonblocking for unicast and rearrangeably nonblocking for arbitrary fan-out multicast traffic.
Wu discloses a rearrangeable non-blocking switch (Abstract) and a routing network being strictly nonblocking for unicast traffic (Col. 2, Lines 9-11, see Fig. 9) and rearrangeably nonblocking for arbitrary fan-out multicast traffic (Col. 4, Lines 6-7; see 'fanouts', Fig. 7B).
Therefore it would have been obvious to one ordinarily skilled in the art at the time of the invention to supplement the teachings of Wong and have a routing network be strictly nonblocking for unicast traffic and rearrangeably nonblocking for arbitrary fan-out multicast traffic, as disclosed by Wu, in order to satisfy specific routing requirements.

Regarding claim 43, Wong discloses d = 4 ('plurality of input terminals and a number of output terminals', claim 1) and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is a generalized multi-link multistage network with full bandwidth ('8x8 Benes network' (each switch has two logic cells that are 2x2, hence 8x8 is 4 switches), Col. 2, Lines 31-33; Col. 6, Lines 7-9; 'networks can be generalized', Col. 6, Lines 7-9), but does not disclose said routing network is strictly nonblocking for arbitrary fan-out multicast traffic.
Wu discloses a rearrangeable non-blocking switch (Abstract) and a routing network strictly nonblocking for arbitrary fan-out multicast traffic (Col. 2, Lines 8-11; see 'fanouts', Fig. 7B).
Therefore it would have been obvious to one ordinarily skilled in the art at the time of the invention to supplement the teachings of Wong and have a routing network be strictly nonblocking for arbitrary fan-out multicast traffic, as disclosed by Wu, in order to satisfy specific routing requirements.

Regarding claim 45, Wong discloses d = 4 ('plurality of input terminals and a number of output terminals', claim 1) and there is only one switch in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there is only one switch in each said stage in each said subintegrated circuit block connecting said backward connecting links and said routing network is a generalized multi-link butterfly fat tree network with full bandwidth. (Col. 2, Lines 27-30; Col. 2, Lines 31-33; 'networks can be generalized', Col. 6, Lines 7-9; 'butterfly pattern', Col. 13, Lines 23-24), but does not disclose said routing network is rearrangeably nonblocking for unicast traffic.
Wu discloses a rearrangeable non-blocking switch (Abstract) and a routing network being rearrangeably nonblocking for unicast traffic ('rearrangeably nonblocking for unicast traffic', Col. 2, Lines 52-54).
Therefore it would have been obvious to one ordinarily skilled in the art at the time of the invention to supplement the teachings of Wong and have a routing network be rearrangeably nonblocking for unicast traffic, as disclosed by Wu, in order to satisfy specific routing requirements.

Regarding claim 46, Wong discloses d = 4 ('plurality of input terminals and a number of output terminals', claim 1) and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network generalized multi-link butterfly fat tree network with full bandwidth ('8x8 Benes network' (each switch has two logic cells that are 2x2, hence 8x8 is 4 switches), Col. 2, Lines 31-33; Col. 6, Lines 7-9; networks can be generalized', Col. 6, Lines 7-9; 'butterfly pattern', Col. 13, Lines 23-24), but does not disclose said routing network is strictly nonblocking for unicast traffic and rearrangeably nonblocking for arbitrary fan-out multicast traffic.
Wu discloses a rearrangeable non-blocking switch (Abstract) and a routing network being strictly nonblocking for unicast traffic (Col. 2, Lines 9-11, see Fig. 9) and rearrangeably nonblocking for arbitrary fan-out multicast traffic (Col. 4, Lines 6-7; see 'fanouts', Fig. 7B).
Therefore it would have been obvious to one ordinarily skilled in the art at the time of the invention to supplement the teachings of Wong and have a routing network be strictly nonblocking for unicast traffic and rearrangeably nonblocking for arbitrary fan-out multicast traffic, as disclosed by Wu, in order to satisfy specific routing requirements.

Regarding claim 47, Wong discloses d = 4 ('plurality of input terminals and a number of output terminals', claim 1) and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is a generalized multilink butterfly fat tree network with full bandwidth. ('8x8 Benes network' (each switch has two logic cells that are 2x2, hence 8x8 is 4 switches), Col. 2, Lines 31-33; Col. 6, Lines 7-9; networks can be generalized', Col. 6, Lines 7-9; 'butterfly pattern', Col. 13, Lines 23-24), but does not disclose said routing network is strictly nonblocking for arbitrary fan-out multicast traffic.
Wu discloses a rearrangeable non-blocking switch (Abstract) and a routing network strictly nonblocking for arbitrary fan-out multicast traffic (Col. 2, Lines 8-11; see 'fanouts', Fig. 7B).
Therefore it would have been obvious to one ordinarily skilled in the art at the time of the invention to supplement the teachings of Wong and have a routing network be strictly nonblocking for arbitrary fan-out multicast traffic, as disclosed by Wu, in order to satisfy specific routing requirements.

Claims 1-49 meet the criteria set out in PCT Article 33(4), and thus have industrial applicability because the subject matter claimed can be made or used in industry.

Form PCT/ISA/237 (Supplemental Box) (April 2007)

INTERNATIONAL SEARCH REPORT	International application No.
	PCT/US2008/064605

A.   CLASSIFICATION OF SUBJECT MATTER
IPC(8) - H01L 25/00 (2008.04)
USPC - 326/41
According to International Patent Classification (IPC) or to both national classification and IPC

B.   FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC(8) - H01L 25/00 (2008.04)
USPC - 326/41

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

MicroPatent

C.   DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 6,940,308 B2 (WONG) 06 September 2005 (06.09.2005) entire document	1-8, 12, 16-19, 21-25, 29, 33, 40, 44, 48, 49
Y		9-11, 13-15, 20, 26-28, 30-32, 34-39, 41-43, 45-47
Y	US 7,154,887 B2 (WU et al) 26 December 2006 (26.12.2006) entire document	9-11, 13-15, 20, 26-28, 30-32, 34-39, 41-43, 45-47

☐   Further documents are listed in the continuation of Box C.   ☐

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance		
"E"	earlier application or patent but published on or after the international filing date	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed	"&"	document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
25 August 2008	03 SEP 2008

Name and mailing address of the ISA/US	Authorized officer:
Mail Stop PCT, Attn: ISA/US, Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-3201	Blaine R. Copenheaver PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774

Form PCT/ISA/210 (second sheet) (April 2005)

Electronic Acknowledgement Receipt

EFS ID:	34422330
Application Number:	16202067
International Application Number:	
Confirmation Number:	8134
Title of Invention:	VLSI Layouts of Fully Connected Generalized Networks
First Named Inventor/Applicant Name:	Venkat Konda
Customer Number:	38139
Filer:	Venkat Konda
Filer Authorized By:	
Attorney Docket Number:	V-0070US
Receipt Date:	28-NOV-2018
Filing Date:	
Time Stamp:	14:58:22
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Miscellaneous Incoming Letter	V0070US-12601275.pdf	862333 b9a690eee7105015896cc171365b45fca0e0df4d	no	9

Warnings:

Information:

2	Miscellaneous Incoming Letter	V0070US-12601275-srch.pdf	51232		no	1
			ae92ff0dd83d6d8edccd38f347c5cf3581d6 9b9d			

Warnings:

Information:

	Total Files Size (in bytes):	913565

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.
National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.
New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

REISSUE PATENT APPLICATION TRANSMITTAL

Address to:		
Mail Stop Reissue **Commissioner for Patents** **P.O. Box 1450** **Alexandria, VA 22313-1450**	Attorney Docket No.	V-0070US
	First Named Inventor	Venkat Konda
	Original Patent Number	8,269,523
	Original Patent Issue Date (Month/Day/Year)	09/18/2012
	Priority Mail Express® Label No.	

APPLICATION FOR REISSUE OF:
(Check applicable box)   [✔] Utility Patent   [ ] Design Patent   [ ] Plant Patent

APPLICATION ELEMENTS (37 CFR 1.173)	ACCOMPANYING APPLICATION PARTS
1. [ ] **Fee Transmittal Form** (PTO/SB/56)	11. [✔] **Statement of status and support for all changes to the claims.** See 37 CFR 1.173(c).
2. [✔] **Applicant asserts small entity status.** See 37 CFR 1.27	12. [ ] **Power of Attorney**
3. [ ] **Applicant certifies micro entity status.** See 37 CFR 1.29. Applicant must attach form PTO/SB/15A or B or equivalent.	13. [ ] **Information Disclosure Statement (IDS)** PTOSB/08 or PTO-1449
4. [✔] **Specification and Claims** in double column copy of patent format *(amended, if appropriate)*	[ ] Copies of citations attached
5. [✔] **Drawing(s)** *(proposed amendments, if appropriate)*	14. [ ] **English translation of Reissue Oath/Declaration** *(if applicable)*
6. [✔] **Reissue Oath/Declaration or Substitute Statement** (37 CFR 1.175) (PTO/AIA/05, 06, or 07)	15. [ ] **Return Receipt Postcard** (MPEP § 503) (Should be specifically itemized)
7. [✔] **Application Data Sheet** NOTE: Benefit claims under 37 CFR 1.78 and foreign priority claims under 37 CFR 1.55 MUST be set forth in an Application Data Sheet (ADS).	16. [✔] **Preliminary Amendment** (37 CFR 1.173; MPEP § 1453)
8. [✔] **Original U.S. Patent currently assigned?** [✔] Yes [ ] No *(If Yes, check applicable box(es))*	17. [ ] **Other:** _____
[✔] Written Consent of all Assigness (PTO/AIA/53)	_____
[ ] 37 CFR 3.73(c) Statement (PTO/AIA/96)	_____
9. [ ] **CD-ROM or CD-R** in duplicate, Computer Program (*Appendix*) or large table	_____
[ ] Landscape Table on CD	_____
10. **Nucleotide and/or Amino Acid Sequence Submission** *(if applicable, items a. – c. are required)*	_____
a. [ ] Computer Readable Form (CRF)	_____
b. [ ] Specification Sequence Listing on:	[ ] This is a continuation reissue or divisional reissue application (*i.e.*, a second or subsequent reissue application for the same issued patent). (*Check box if applicable.*)
i. [ ] CD-ROM (2 copies) or CD-R (2 copies); **or**	
ii. [ ] Paper	
c. [ ] Statements verifying identity of above copies	

18. CORRESPONDENCE ADDRESS

[✔] The address associated with Customer Number: 38139 _____ **OR** [ ] Correspondence address below

Name	
Address	
City		State		Zip Code	
Country		Telephone	
Email	

Signature	/Venkat Konda/	Date	11/27/2018
Name (Print/Type)	Venkat Konda	Registration No.	

This collection of information is required by 37 CFR 1.173. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Mail Stop Reissue, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**
If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Privacy Act Statement

The **Privacy Act of 1974 (P.L. 93-579)** requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (*i.e.,* GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

REISSUE APPLICATION: CONSENT OF ASSIGNEE; STATEMENT OF NON-ASSIGNMENT	Docket Number (Optional) V-0070US

This is part of the application for a reissue patent based on the original patent identified below.

Name of Patentee(s)

Konda Technologies Inc.

Patent Number 8,269,523	Date Patent Issued September 18, 2012

Title of Invention

VLSI Layouts of Fully Connected Generalized Networks

1. ☒ Filed herein is a statement under 37 CFR 3.73(c). (Form PTO/AIA/96)

2. ☐ Ownership of the patent is in the inventor(s), and no assignment of the patent is in effect.

One of boxes 1 or 2 above must be checked. If multiple assignees, complete this form for each assignee. If box 2 is checked, skip the next entry and go directly to "Name of Assignee."

The written consent of all assignees and inventors owning an undivided interest in the original patent is included in this application for reissue.

The assignee(s) owning an undivided interest in said original patent is/are Konda Technologies Inc. , and the assignee(s) consents to the accompanying application for reissue.

Name of assignee/inventor (if not assigned)

Signature /Venkat Konda/	Date 11-27-2018

Typed or printed name and title of person signing for assignee (if assigned)

Venkat Konda

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Privacy Act Statement

The **Privacy Act of 1974 (P.L. 93-579)** requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (*i.e.,* GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

STATEMENT UNDER 37 CFR 3.73(c)

Applicant/Patent Owner: Konda Technologies inc.

Application No./Patent No.: 8,269,523   Filed/Issue Date: September 18, 2012

Titled: VLSI Layouts of Fully Connected Generalized Networks

Konda Technologies inc. _____, a California C Corporation _____

(Name of Assignee)   (Type of Assignee, e.g., corporation, partnership, university, government agency, etc.)

states that, for the patent application/patent identified above, it is (choose **one** of options 1, 2, 3 or 4 below):

1. [✔] The assignee of the entire right, title, and interest.

2. [ ] An assignee of less than the entire right, title, and interest (check applicable box):

   [ ] The extent (by percentage) of its ownership interest is _____%. Additional Statement(s) by the owners holding the balance of the interest _must be submitted_ to account for 100% of the ownership interest.

   [ ] There are unspecified percentages of ownership. The other parties, including inventors, who together own the entire right, title and interest are:

   Additional Statement(s) by the owner(s) holding the balance of the interest _must be submitted_ to account for the entire right, title, and interest.

3. [ ] The assignee of an undivided interest in the entirety (a complete assignment from one of the joint inventors was made). The other parties, including inventors, who together own the entire right, title, and interest are:

   Additional Statement(s) by the owner(s) holding the balance of the interest _must be submitted_ to account for the entire right, title, and interest.

4. [ ] The recipient, via a court proceeding or the like (_e.g._, bankruptcy, probate), of an undivided interest in the entirety (a complete transfer of ownership interest was made). The certified document(s) showing the transfer is attached.

The interest identified in option 1, 2 or 3 above (not option 4) is evidenced by either (choose **one** of options A or B below):

A. [ ] An assignment from the inventor(s) of the patent application/patent identified above. The assignment was recorded in the United States Patent and Trademark Office at Reel _____, Frame _____, or for which a copy thereof is attached.

B. [ ] A chain of title from the inventor(s), of the patent application/patent identified above, to the current assignee as follows:

   1. From: _____ To: _____

      The document was recorded in the United States Patent and Trademark Office at

      Reel _____, Frame _____, or for which a copy thereof is attached.

   2. From: _____ To: _____

      The document was recorded in the United States Patent and Trademark Office at

      Reel _____, Frame _____, or for which a copy thereof is attached.

[Page 1 of 2]

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Approved for use through 01/31/2013. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

STATEMENT UNDER 37 CFR 3.73(c)

3. From: _____ To: _____

      The document was recorded in the United States Patent and Trademark Office at

      Reel _____, Frame _____, or for which a copy thereof is attached.

4. From: _____ To: _____

      The document was recorded in the United States Patent and Trademark Office at

      Reel _____, Frame _____, or for which a copy thereof is attached.

5. From: _____ To: _____

      The document was recorded in the United States Patent and Trademark Office at

      Reel _____, Frame _____, or for which a copy thereof is attached.

6. From: _____ To: _____

      The document was recorded in the United States Patent and Trademark Office at

      Reel _____, Frame _____, or for which a copy thereof is attached.

☐   Additional documents in the chain of title are listed on a supplemental sheet(s).

☐   As required by 37 CFR 3.73(c)(1)(i), the documentary evidence of the chain of title from the original owner to the assignee was, or concurrently is being, submitted for recordation pursuant to 37 CFR 3.11.

     [NOTE: A separate copy (i.e., a true copy of the original assignment document(s)) must be submitted to Assignment Division in accordance with 37 CFR Part 3, to record the assignment in the records of the USPTO. See MPEP 302.08]

The undersigned (whose title is supplied below) is authorized to act on behalf of the assignee.

/Venkat Konda/	11-27-2018
Signature	Date
Venkat Konda	Founder/CEO
Printed or Typed Name	Title or Registration Number

[Page 2 of 2]

Privacy Act Statement

The **Privacy Act of 1974 (P.L. 93-579)** requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (*i.e.*, GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

Electronic Patent Application Fee Transmittal

Application Number:	
Filing Date:	
Title of Invention:	VLSI LAYOUTS OF FULLY CONNECTED GENERALIZED NETWORKS
First Named Inventor/Applicant Name:	venkat konda
Filer:	Venkat Konda
Attorney Docket Number:	V-0070US

Filed as Small Entity

Filing Fees for   Reissue (Utility)

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
UTILITY REISSUE BASIC	2014	1	150	150
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Extension-of-Time:				
Miscellaneous:				
	Total in USD ($)			**150**

Electronic Acknowledgement Receipt

EFS ID:	34414552
Application Number:	16202067
International Application Number:	
Confirmation Number:	8134
Title of Invention:	VLSI LAYOUTS OF FULLY CONNECTED GENERALIZED NETWORKS
First Named Inventor/Applicant Name:	venkat konda
Customer Number:	38139
Filer:	Venkat Konda
Filer Authorized By:	
Attorney Docket Number:	V-0070US
Receipt Date:	27-NOV-2018
Filing Date:	
Time Stamp:	22:44:20
Application Type:	Reissue (Utility)

Payment information:

Submitted with Payment	yes
Payment Type	CARD
Payment was successfully received in RAM	$ 150
RAM confirmation Number	112818INTEFSW22474700
Deposit Account	
Authorized User	
The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows:	

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Preliminary Amendment	V0070US-PrelimAmendA.pdf	190468 bffef1032455046c781801e6a9adc50b3bfad902	no	15
Warnings:					
Information:					
2	Abstract	V-0070US-Abstract.pdf	89030 04306185b1c36eac11ae806af9f61f5f01b18946	no	1
Warnings:					
Information:					
3	Specification	V-0070US-Spec.pdf	2063673 9bff3d019bb67ca3ee17d784a4504fdf1f9c236f	no	17
Warnings:					
Information:					
4	Appendix to the Specification	V-0070US-Apndx-spec.pdf	11292 300c5b523792ddf9b228563ee75eec1fa6b720d8	no	1
Warnings:					
Information:					
5	Claims	V-0070US-Claims.pdf	400569 86b3b006105feb20761b7493096c8f2b47c6ac84	no	4
Warnings:					
Information:					
6	Drawings-only black and white line drawings	V-0070US-Drawings.pdf	1631019 0f7974c72e7cc3692f946bf5f8aa65db1b60d865	no	39
Warnings:					
Information:					

7	Reissue dec filed in accordance with MPEP 1414	aia0005.pdf	216269 055f422e3e07b42baae8860e96cd31a0d9c1c3fa	no	3

Warnings:

Information:

8	Reissue dec filed in accordance with MPEP 1414	aia0006.pdf	346707 dd47005cd34a0388aeff659dc5525709881a5b53	no	3

Warnings:

Information:

9	Application Data Sheet	aia0014.pdf	1878647 95679691ba6251ae3ff657379a8bab1ed8702356	no	8

Warnings:

Information:

10	Transmittal Reissue Application	aia0050.pdf	291961 29ddd5484cde7aec837afe1efeb8dcd93e59cda9	no	2

Warnings:

Information:

11	Consent of Assignee accompanying the declaration	aia0053.pdf	337786 b8d069208dfb97f83639d3541fe768e7982cb421	no	2

Warnings:

Information:

12	Assignee showing of ownership per 37 CFR 3.73	aia0096.pdf	114993 7ac0d3d80132185b61e61ca48cd510a1f43562af	no	3

Warnings:

Information:

13	Fee Worksheet (SB06)	fee-info.pdf	29356 d3f02680c0da4d8f7b3c5bbf8af9fb840bd8ce47	no	2

Warnings:

Information:

	Total Files Size (in bytes):	7601770

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.
National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.
New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

US 8,269,523 B2

1

VLSI LAYOUTS OF FULLY CONNECTED GENERALIZED NETWORKS

CROSS REFERENCE TO RELATED APPLICATIONS

This application is related to and claims priority of the PCT Application Serial No. PCT/US08/64605 entitled "VLSI LAYOUTS OF FULLY CONNECTED GENERALIZED NETWORKS" by Venkat Konda assigned to the same assignee as the current application, filed May 22, 2008, and the U.S. Provisional Patent Application Ser. No. 60/940,394 entitled "VLSI LAYOUTS OF FULLY CONNECTED GENERALIZED NETWORKS" by Venkat Konda assigned to the same assignee as the current application, filed May 25, 2007.

This application is related to and incorporates by reference in its entirety the U.S. application Ser. No. 12/530,207 entitled "FULLY CONNECTED GENERALIZED MULTI-STAGE NETWORKS" by Venkat Konda assigned to the same assignee as the current application, filed Sep. 6, 2009, the PCT Application Serial No. PCT/US08/56064 entitled "FULLY CONNECTED GENERALIZED MULTI-STAGE NETWORKS" by Venkat Konda assigned to the same assignee as the current application, filed Mar. 6, 2008, the U.S. Provisional Patent Application Ser. No. 60/905,526 entitled "LARGE SCALE CROSSPOINT REDUCTION WITH NONBLOCKING UNICAST & MULTICAST IN ARBITRARILY LARGE MULTI-STAGE NETWORKS" by Venkat Konda assigned to the same assignee as the current application, filed Mar. 6, 2007, and the U.S. Provisional Patent Application Ser. No. 60/940,383 entitled "FULLY CONNECTED GENERALIZED MULTI-STAGE NETWORKS" by Venkat Konda assigned to the same assignee as the current application, filed May 25, 2007.

This application is related to and incorporates by reference in its entirety the U.S. patent application Ser. No. 12/601,273 entitled "FULLY CONNECTED GENERALIZED BUTTERFLY FAT TREE NETWORKS" by Venkat Konda assigned to the same assignee as the current application filed concurrently, the PCT Application Serial No. PCT/US08/64603 entitled "FULLY CONNECTED GENERALIZED BUTTERFLY FAT TREE NETWORKS" by Venkat Konda assigned to the same assignee as the current application, filed May 22, 2008, the U.S. Provisional Patent Application Ser. No. 60/940,387 entitled "FULLY CONNECTED GENERALIZED BUTTERFLY FAT TREE NETWORKS" by Venkat Konda assigned to the same assignee as the current application, filed May 25, 2007, and the U.S. Provisional Patent Application Ser. No. 60/940,390 entitled "FULLY CONNECTED GENERALIZED MULTI-LINK BUTTERFLY FAT TREE NETWORKS" by Venkat Konda assigned to the same assignee as the current application, filed May 25, 2007.

This application is related to and incorporates by reference in its entirety the U.S. patent application Ser. No. 12/601,274 entitled "FULLY CONNECTED GENERALIZED MULTI-LINK MULTI-STAGE NETWORKS" by Venkat Konda assigned to the same assignee as the current application filed concurrently, the PCT Application Serial No. PCT/US08/64604 entitled "FULLY CONNECTED GENERALIZED MULTI-LINK MULTI-STAGE NETWORKS" by Venkat Konda assigned to the same assignee as the current application, filed May 22, 2008, the U.S. Provisional Patent Application Ser. No. 60/940,389 entitled "FULLY CONNECTED GENERALIZED REARRANGEABLY NONBLOCKING MULTI-LINK MULTI-STAGE NETWORKS" by Venkat Konda assigned to the same assignee as the current application, filed May 25, 2007, the U.S. Provisional Patent Appli-

2

cation Ser. No. 60/940,391 entitled "FULLY CONNECTED GENERALIZED FOLDED MULTI-STAGE NETWORKS" by Venkat Konda assigned to the same assignee as the current application, filed May 25, 2007 and the U.S. Provisional Patent Application Ser. No. 60/940,392 entitled "FULLY CONNECTED GENERALIZED STRICTLY NON-BLOCKING MULTI-LINK MULTI-STAGE NETWORKS" by Venkat Konda assigned to the same assignee as the current application, filed May 25, 2007.

This application is related to and incorporates by reference in its entirety the U.S. Provisional Patent Application Ser. No. 61/252, 603 entitled "VLSI LAYOUTS OF FULLY CONNECTED NETWORKS WITH LOCALITY EXPLOITATION" by Venkat Konda assigned to the same assignee as the current application, filed Oct. 16, 2009.

This application is related to and incorporates by reference in its entirety the U.S. Provisional Patent Application Ser. No. 61/252, 609 entitled "VLSI LAYOUTS OF FULLY CONNECTED GENERALIZED AND PYRAMID NETWORKS" by Venkat Konda assigned to the same assignee as the current application, filed Oct. 16, 2009.

BACKGROUND OF INVENTION

Multi-stage interconnection networks such as Benes networks and butterfly fat tree networks are widely useful in telecommunications, parallel and distributed computing. However VLSI layouts, known in the prior art, of these interconnection networks in an integrated circuit are inefficient and complicated.

Other multi-stage interconnection networks including butterfly fat tree networks, Banyan networks, Batcher-Banyan networks, Baseline networks, Delta networks, Omega networks and Flip networks have been widely studied particularly for self routing packet switching applications. Also Benes Networks with radix of two have been widely studied and it is known that Benes Networks of radix two are shown to be built with back to back baseline networks which are rearrangeably nonblocking for unicast connections.

The most commonly used VLSI layout in an integrated circuit is based on a two-dimensional grid model comprising only horizontal and vertical tracks. An intuitive interconnection network that utilizes two-dimensional grid model is 2D Mesh Network and its variations such as segmented mesh networks. Hence routing networks used in VLSI layouts are typically 2D mesh networks and its variations. However Mesh Networks require large scale cross points typically with a growth rate of $O(N^2)$ where N is the number of computing elements, ports, or logic elements depending on the application.

Multi-stage interconnection with a growth rate of $O(N \times \log N)$ requires significantly small number of cross points. U.S. Pat. No. 6,185,220 entitled "Grid Layouts of Switching and Sorting Networks" granted to Muthukrishnan et al. describes a VLSI layout using existing VLSI grid model for Benes and Butterfly networks. U.S. Pat. No. 6,940,308 entitled "Interconnection Network for a Field Programmable Gate Array" granted to Wong describes a VLSI layout where switches belonging to lower stage of Benes Network are layed out close to the logic cells and switches belonging to higher stages are layed out towards the center of the layout.

Due to the inefficient and in some cases impractical VLSI layout of Benes and butterfly fat tree networks on a semiconductor chip, today mesh networks and segmented mesh networks are widely used in the practical applications such as field programmable gate arrays (FPGAs), programmable logic devices (PLDs), and parallel computing interconnects.

US 8,269,523 B2

<table>
<tr><td>3</td><td>4</td></tr>
</table>

The prior art VLSI layouts of Benes and butterfly fat tree networks and VLSI layouts of mesh networks and segmented mesh networks require large area to implement the switches on the chip, large number of wires, longer wires, with increased power consumption, increased latency of the signals which effect the maximum clock speed of operation. Some networks may not even be implemented practically on a chip due to the lack of efficient layouts.

SUMMARY OF INVENTION

When large scale sub-integrated circuit blocks with inlet and outlet links are layed out in an integrated circuit device in a two-dimensional grid arrangement, (for example in an FPGA where the sub-integrated circuit blocks are Lookup Tables) the most intuitive routing network is a network that uses horizontal and vertical links only (the most often used such a network is one of the variations of a 2D Mesh network). A direct embedding of a generalized multi-stage network on to a 2D Mesh network is neither simple nor efficient.

In accordance with the invention, VLSI layouts of generalized multi-stage networks for broadcast, unicast and multicast connections are presented using only horizontal and vertical links The VLSI layouts employ shuffle exchange links where outlet links of cross links from switches in a stage in one sub-integrated circuit block are connected to inlet links of switches in the succeeding stage in another sub-integrated circuit block so that said cross links are either vertical links or horizontal and vice versa. In one embodiment the sub-integrated circuit blocks are arranged in a hypercube arrangement in a two-dimensional plane. The VLSI layouts exploit the benefits of significantly lower cross points, lower signal latency, lower power and full connectivity with significantly fast compilation.

The VLSI layouts presented are applicable to generalized multi-stage networks $V(N_1, N_2, d, s)$, generalized folded multi-stage networks $V_{fold}(N_1, N_2, d, s)$, generalized butterfly fat tree networks $V_{bft}(N_1, N_2, d, s)$, generalized multi-link multi-stage networks $V_{mlink}(N_1, N_2, d, s)$, generalized folded multi-link multi-stage networks $V_{fold-mlink}(N_1, N_2, d, s)$, generalized multi-link butterfly fat tree networks $V_{mlink-bft}(N_1, N_2, d, s)$, and generalized hypercube networks $V_{hcube}(N_1, N_2, d, s)$ for s=1, 2, 3 or any number in general. The embodiments of VLSI layouts are useful in wide target applications such as FPGAs, CPLDs, pSoCs, ASIC placement and route tools, networking applications, parallel & distributed computing, and reconfigurable computing.

BRIEF DESCRIPTION OF DRAWINGS

FIG. 1A is a diagram 100A of an exemplary symmetrical multi-link multi-stage network $V_{fold-mlink}(N, d, s)$ having inverse Benes connection topology of nine stages with N=32, d=2 and s=2, strictly nonblocking network for unicast connections and rearrangeably nonblocking network for arbitrary fan-out multicast connections, in accordance with the invention.

FIG. 1B is a diagram 100B of the equivalent symmetrical folded multi-link multi-stage network $V_{fold-mlink}(N, d, s)$ of the network 100A shown in FIG. 1A, having inverse Benes connection topology of five stages with N=32, d=2 and s=2, strictly nonblocking network for unicast connections and rearrangeably nonblocking network for arbitrary fan-out multicast connections, in accordance with the invention.

FIG. 1C is a diagram 100C layout of the network $V_{fold-mlink}$ (N, d, s) shown in FIG. 1B, in one embodiment, illustrating the connection links belonging with in each block only.

FIG. 1D is a diagram 100D layout of the network $V_{fold-mlink}(N, d, s)$ shown in FIG. 1B, in one embodiment, illustrating the connection links ML(1,i) for i=[1, 64] and ML(8,i) for i=[1,64].

FIG. 1E is a diagram 100E layout of the network $V_{fold-mlink}$ (N, d, s) shown in FIG. 1B, in one embodiment, illustrating the connection links ML(2,i) for i=[1, 64] and ML(7,i) for i=[1,64].

FIG. 1F is a diagram 100F layout of the network $V_{fold-mlink}$ (N, d, s) shown in FIG. 1B, in one embodiment, illustrating the connection links ML(3,i) for i=[1, 64] and ML(6,i) for i=[1,64].

FIG. 1G is a diagram 100G layout of the network $V_{fold-mlink}(N, d, s)$ shown in FIG. 1B, in one embodiment, illustrating the connection links ML(4,i) for i=[1, 64] and ML(5,i) for i=[1,64].

FIG. 1H is a diagram 100H layout of a network $V_{fold-mlink}$ (N, d, s) where N=128, d=2, and s=2, in one embodiment, illustrating the connection links belonging with in each block only.

FIG. 1I is a diagram 100I detailed connections of BLOCK 1_2 in the network layout 100C in one embodiment, illustrating the connection links going in and coming out when the layout 100C is implementing V(N, d, s) or $V_{fold}(N, d, s)$.

FIG. 1J is a diagram 100J detailed connections of BLOCK 1_2 in the network layout 100C in one embodiment, illustrating the connection links going in and coming out when the layout 100C is implementing V(N, d, s) or $V_{fold}(N, d, s)$.

FIG. 1K is a diagram 100K detailed connections of BLOCK 1_2 in the network layout 100C in one embodiment, illustrating the connection links going in and coming out when the layout 100C is implementing V(N, d, s) or $V_{fold}(N, d, s)$.

FIG. 1K1 is a diagram 100M1 detailed connections of BLOCK 1_2 in the network layout 100C in one embodiment, illustrating the connection links going in and coming out when the layout 100C is implementing V(N, d, s) or $V_{fold}(N, d, s)$ for s=1.

FIG. 1L is a diagram 100L detailed connections of BLOCK 1_2 in the network layout 100C in one embodiment, illustrating the connection links going in and coming out when the layout 100C is implementing V(N, d, s) or $V_{fold}(N, d, s)$.

FIG. 1L1 is a diagram 100L1 detailed connections of BLOCK 1_2 in the network layout 100C in one embodiment, illustrating the connection links going in and coming out when the layout 100C is implementing V(N, d, s) or $V_{fold}(N, d, s)$ for s=1.

FIG. 2A1 is a diagram 200A1 of an exemplary symmetrical multi-link multi-stage network $V_{fold-mlink}(N, d, s)$ having inverse Benes connection topology of one stage with N=2, d=2 and s=2, strictly nonblocking network for unicast connections and rearrangeably nonblocking network for arbitrary fan-out multicast connections, in accordance with the invention. FIG. 2A2 is a diagram 200A2 of the equivalent symmetrical folded multi-link multi-stage network $V_{fold-mlink}$ (N, d, s) of the network 200A1 shown in FIG. 2A1, having inverse Benes connection topology of one stage with N=2, d=2 and s=2, strictly nonblocking network for unicast connections and rearrangeably nonblocking network for arbitrary fan-out multicast connections, in accordance with the invention. FIG. 2A3 is a diagram 200A3 layout of the network $V_{fold-mlink}(N, d, s)$ shown in FIG. 2A2, in one embodiment, illustrating all the connection links.

FIG. 2B1 is a diagram 200B1 of an exemplary symmetrical multi-link multi-stage network $V_{fold-mlink}(N, d, s)$ having inverse Benes connection topology of one stage with N=4, d=2 and s=2, strictly nonblocking network for unicast con-

US 8,269,523 B2

5   6

nections and rearrangeably nonblocking network for arbitrary fan-out multicast connections, in accordance with the invention. FIG. 2B2 is a diagram 200B2 of the equivalent symmetrical folded multi-link multi-stage network $V_{fold-mlink}$ (N, d, s) of the network 200B1 shown in FIG. 2B1, having inverse Benes connection topology of one stage with N=4, d=2 and s=2, strictly nonblocking network for unicast connections and rearrangeably nonblocking network for arbitrary fan-out multicast connections, in accordance with the invention. FIG. 2B3 is a diagram 200B3 layout of the network $V_{fold-mlink}$(N, d, s) shown in FIG. 2B2, in one embodiment, illustrating the connection links belonging with in each block only. FIG. 2B4 is a diagram 200B4 layout of the network $V_{fold-mlink}$(N, d, s) shown in FIG. 2B2, in one embodiment, illustrating the connection links ML(1,i) for i=[1, 8] and ML(2,i) for i=[1,8].

FIG. 2C11 is a diagram 200C11 of an exemplary symmetrical multi-link multi-stage network $V_{fold-mlink}$(N, d, s) having inverse Benes connection topology of one stage with N=8, d=2 and s=2, strictly nonblocking network for unicast connections and rearrangeably nonblocking network for arbitrary fan-out multicast connections, in accordance with the invention. FIG. 2C12 is a diagram 200C12 of the equivalent symmetrical folded multi-link multi-stage network $V_{fold-mlink}$ (N, d, s) of the network 200C11 shown in FIG. 2C11, having inverse Benes connection topology of one stage with N=8, d=2 and s=2, strictly nonblocking network for unicast connections and rearrangeably nonblocking network for arbitrary fan-out multicast connections, in accordance with the invention.

FIG. 2C21 is a diagram 200C21 layout of the network $V_{fold-mlink}$(N, d, s) shown in FIG. 2C12, in one embodiment, illustrating the connection links belonging with in each block only. FIG. 2C22 is a diagram 200C22 layout of the network $V_{fold-mlink}$(N, d, s) shown in FIG. 2C12, in one embodiment, illustrating the connection links ML(1,i) for i=[1, 16] and ML(4,i) for i=[1,16]. FIG. 2C23 is a diagram 200C23 layout of the network $V_{fold-mlink}$(N, d, s) shown in FIG. 2C12, in one embodiment, illustrating the connection links ML(2,i) for i=[1, 16] and ML(3,i) for i=[1,16].

FIG. 2D1 is a diagram 200D1 of an exemplary symmetrical multi-link multi-stage network $V_{fold-mlink}$(N, d, s) having inverse Benes connection topology of one stage with N=16, d=2 and s=2, strictly nonblocking network for unicast connections and rearrangeably nonblocking network for arbitrary fan-out multicast connections, in accordance with the invention.

FIG. 2D2 is a diagram 200D2 of the equivalent symmetrical folded multi-link multi-stage network $V_{fold-mlink}$(N, d, s) of the network 200D1 shown in FIG. 2D1, having inverse Benes connection topology of one stage with N=16, d=2 and s=2, strictly nonblocking network for unicast connections and rearrangeably nonblocking network for arbitrary fan-out multicast connections, in accordance with the invention.

FIG. 2D3 is a diagram 200D3 layout of the network $V_{fold-mlink}$(N, d, s) shown in FIG. 2D2, in one embodiment, illustrating the connection links belonging with in each block only.

FIG. 2D4 is a diagram 200D4 layout of the network $V_{fold-mlink}$(N, d, s) shown in FIG. 2D2, in one embodiment, illustrating the connection links ML(1,i) for i=[1, 32] and ML(6,i) for i=[1,32].

FIG. 2D5 is a diagram 200D5 layout of the network $V_{fold-mlink}$(N, d, s) shown in FIG. 2D2, in one embodiment, illustrating the connection links ML(2,i) for i=[1, 32] and ML(5,i) for i=[1,32].

FIG. 2D6 is a diagram 200D6 layout of the network $V_{fold-mlink}$(N, d, s) shown in FIG. 2D2, in one embodiment, illustrating the connection links ML(3,i) for i=[1, 32] and ML(4,i) for i=[1,32].

FIG. 3A is a diagram 300A of an exemplary symmetrical multi-link multi-stage network V_{hcube}(N, d, s) having inverse Benes connection topology of nine stages with N=32, d=2 and s=2, strictly nonblocking network for unicast connections and rearrangeably nonblocking network for arbitrary fan-out multicast connections, in accordance with the invention.

FIG. 3B is a diagram 300B of the equivalent symmetrical folded multi-link multi-stage network V_{hcube}(N, d, s) of the network 300A shown in FIG. 3A, having inverse Benes connection topology of five stages with N=32, d=2 and s=2, strictly nonblocking network for unicast connections and rearrangeably nonblocking network for arbitrary fan-out multicast connections, in accordance with the invention.

FIG. 3C is a diagram 300C layout of the network V_{hcube}(N, d, s) shown in FIG. 3B, in one embodiment, illustrating the connection links belonging with in each block only.

FIG. 3D is a diagram 100D layout of the network V_{hcube}(N, d, s) shown in FIG. 3B, in one embodiment, illustrating the connection links ML(1,i) for i=[1, 64] and ML(8,i) for i=[1, 64].

FIG. 3E is a diagram 300E layout of the network V_{hcube}(N, d, s) shown in FIG. 3B, in one embodiment, illustrating the connection links ML(2,i) for i=[1, 64] and ML(7,i) for i=[1, 64].

FIG. 3F is a diagram 300F layout of the network V_{hcube}(N, d, s) shown in FIG. 3B, in one embodiment, illustrating the connection links ML(3,i) for i=[1, 64] and ML(6,i) for i=[1, 64].

FIG. 3G is a diagram 300G layout of the network V_{hcube}(N, d, s) shown in FIG. 3B, in one embodiment, illustrating the connection links ML(4,i) for i=[1, 64] and ML(5,i) for i=[1, 64].

FIG. 3H is a diagram 300H layout of a network V_{hcube}(N, d, s) where N=128, d=2, and s=2, in one embodiment, illustrating the connection links belonging with in each block only.

FIG. 4A is a diagram 400A layout of the network $V_{fold-mlink}$(N, d, s) shown in FIG. 1B, in one embodiment, illustrating the connection links belonging with in each block only.

FIG. 4B is a diagram 400B layout of the network $V_{fold-mlink}$(N, d, s) shown in FIG. 1B, in one embodiment, illustrating the connection links ML(1,i) for i=[1, 64] and ML(8,i) for i=[1,64].

FIG. 4C is a diagram 400C layout of the network $V_{fold-mlink}$(N, d, s) shown in FIG. 4C, in one embodiment, illustrating the connection links ML(2,i) for i=[1, 64] and ML(7,i) for i=[1,64].

FIG. 4D is a diagram 400D layout of the network $V_{fold-mlink}$(N, d, s) shown in FIG. 4D, in one embodiment, illustrating the connection links ML(3,i) for i=[1, 64] and ML(6,i) for i=[1,64].

FIG. 4E is a diagram 400E layout of the network $V_{fold-mlink}$(N, d, s) shown in FIG. 4E, in one embodiment, illustrating the connection links ML(4,i) for i=[1, 64] and ML(5,i) for i=[1,64].

FIG. 4C1 is a diagram 400C1 layout of the network $V_{fold-mlink}$(N, d, s) shown in FIG. 1B, in one embodiment, illustrating the connection links belonging with in each block only.

FIG. 5A1 is a diagram 500A1 of an exemplary prior art implementation of a two by two switch; FIG. 5A2 is a dia-

US 8,269,523 B2

7

gram **500A2** for programmable integrated circuit prior art implementation of the diagram **500A1** of FIG. **5A1**; FIG. **5A3** is a diagram **500A3** for one-time programmable integrated circuit prior art implementation of the diagram **500A1** of FIG. **5A1**; FIG. **5A4** is a diagram **500A4** for integrated circuit placement and route implementation of the diagram **500A1** of FIG. **5A1**.

DETAILED DESCRIPTION OF THE INVENTION

The present invention is concerned with the VLSI layouts of arbitrarily large switching networks for broadcast, unicast and multicast connections. Particularly switching networks considered in the current invention include: generalized multi-stage networks $V(N_1, N_2, d, s)$, generalized folded multi-stage networks $V_{fold}(N_1, N_2, d, s)$, generalized butterfly fat tree networks $V_{bft}(N_1, N_2, d, s)$, generalized multi-link multi-stage networks $V_{mlink}(N_1, N_2, d, s)$, generalized folded multi-link multi-stage networks $V_{fold-mlink}(N_1, N_2, d, s)$, generalized multi-link butterfly fat tree networks $V_{mlink-bft}(N_1, N_2, d, s)$, and generalized hypercube networks $V_{hcube}(N_1, N_2, d, s)$ for s=1, 2, 3 or any number in general.

Efficient VLSI layout of networks on a semiconductor chip are very important and greatly influence many important design parameters such as the area taken up by the network on the chip, total number of wires, length of the wires, latency of the signals, capacitance and hence the maximum clock speed of operation. Some networks may not even be implemented practically on a chip due to the lack of efficient layouts. The different varieties of multi-stage networks described above have not been implemented previously on the semiconductor chips efficiently. For example in Field Programmable Gate Array (FPGA) designs, multi-stage networks described in the current invention have not been successfully implemented primarily due to the lack of efficient VLSI layouts. Current commercial FPGA products such as Xilinx Vertex, Altera's Stratix implement island-style architecture using mesh and segmented mesh routing interconnects using either full cross-bars or sparse crossbars. These routing interconnects consume large silicon area for crosspoints, long wires, large signal propagation delay and hence consume lot of power.

The current invention discloses the VLSI layouts of numerous types of multi-stage networks which are very efficient. Moreover they can be embedded on to mesh and segmented mesh routing interconnects of current commercial FPGA products. The VLSI layouts disclosed in the current invention are applicable to including the numerous generalized multi-stage networks disclosed in the following patent applications, filed concurrently:

1) Strictly and rearrangeably nonblocking for arbitrary fan-out multicast and unicast for generalized multi-stage networks $V(N_1, N_2, d, s)$ with numerous connection topologies and the scheduling methods are described in detail in the PCT Application Serial No. PCT/US08/56064 that is incorporated by reference above.

2) Strictly and rearrangeably nonblocking for arbitrary fan-out multicast and unicast, and strictly nonblocking for unicast for generalized butterfly fat tree networks $V_{bft}(N_1, N_2, d, s)$ with numerous connection topologies and the scheduling methods are described in detail in PCT Application Serial No. PCT/US08/64603 that is incorporated by reference above.

3) Rearrangeably nonblocking for arbitrary fan-out multicast and unicast, and strictly nonblocking for unicast for generalized multi-link multi-stage networks $V_{mlink}(N_1, N_2, d, s)$ and generalized folded multi-link multi-stage networks $V_{fold-mlink}(N_1, N_2, d, s)$ with numerous connection topologies

8

and the scheduling methods are described in detail in PCT Application Serial No. PCT/US08/64604 that is incorporated by reference above.

4) Strictly and rearrangeably nonblocking for arbitrary fan-out multicast and unicast for generalized multi-link butterfly fat tree networks $V_{mlink-bft}(N_1, N_2, d, s)$ with numerous connection topologies and the scheduling methods are described in detail in PCT Application Serial No. PCT/US08/64603 that is incorporated by reference above.

5) Strictly and rearrangeably nonblocking for arbitrary fan-out multicast and unicast for generalized folded multi-stage networks $V_{fold}(N_1, N_2, d, s)$ with numerous connection topologies and the scheduling methods are described in detail in PCT Application Serial No. PCT/US08/64604 that is incorporated by reference above.

6) Strictly nonblocking for arbitrary fan-out multicast for generalized multi-link multi-stage networks $V_{mlink}(N_1, N_2, d, s)$ and generalized folded multi-link multi-stage networks $V_{fold-mlink}(N_1, N_2, d, s)$ with numerous connection topologies and the scheduling methods are described in detail in PCT Application Serial No. PCT/US08/64604 that is incorporated by reference above.

7) VLSI layouts of numerous types of multi-stage networks with locality exploitation are described in U.S. Provisional Patent Application Ser. No. 61/252,603 that is incorporated by reference above.

8) VLSI layouts of numerous types of multistage pyramid networks are described in U.S. Provisional Patent Application Ser. No. 61/252,609 that is incorporated by reference above.

In addition the layouts of the current invention are also applicable to generalized multi-stage pyramid networks $V_p(N_1, N_2, d, s)$, generalized folded multi-stage pyramid networks $V_{fold-p}(N_1, N_2, d, s)$, generalized butterfly fat pyramid networks $V_{bfp}(N_1, N_2, d, s)$, generalized multi-link multi-stage pyramid networks $V_{mlink-p}(N_1, N_2, d, s)$, generalized folded multi-link multi-stage pyramid networks $V_{fold-mlink-p}(N_1, N_2, d, s)$, generalized multi-link butterfly fat pyramid networks $V_{mlink-bfp}(N_1, N_2, d, s)$, and generalized hypercube networks $V_{hcube}(N_1, N_2, d, s)$ for s=1, 2, 3 or any number in general.

Symmetric RNB Generalized Multi-Link Multi-Stage Network $V_{mlink}(N_1, N_2, d, s)$:

Referring to diagram **100A** in FIG. **1A**, in one embodiment, an exemplary generalized multi-link multi-stage network $V_{mlink}(N_1, N_2, d, s)$ where $N_1=N_2=32$; d=2; and s=2 with nine stages of one hundred and forty four switches for satisfying communication requests, such as setting up a telephone call or a data call, or a connection between configurable logic blocks, between an input stage **110** and output stage **120** via middle stages **130**, **140**, **150**, **160**, **170**, **180** and **190** is shown where input stage **110** consists of sixteen, two by four switches IS1-IS16 and output stage **120** consists of sixteen, four by two switches OS1-OS16. And all the middle stages namely the middle stage **130** consists of sixteen, four by four switches MS(**1,1**)-MS(**1,16**), middle stage **140** consists of sixteen, four by four switches MS(**2,1**)-MS(**2,16**), middle stage **150** consists of sixteen, four by four switches MS(**3,1**)-MS(**3,16**), middle stage **160** consists of sixteen, four by four switches MS(**4,1**)-MS(**4,16**), middle stage **170** consists of sixteen, four by four switches MS(**5,1**)-MS(**5,16**), middle stage **180** consists of sixteen, four by four switches MS(**6,1**)-MS(**6,16**), and middle stage **190** consists of sixteen, four by four switches MS(**7,1**)-MS(**7,16**).

As disclosed in PCT Application Serial No. PCT/US08/64604 that is incorporated by reference above, such a network can be operated in rearrangeably non-blocking manner for

US 8,269,523 B2

9

10

arbitrary fan-out multicast connections and also can be operated in strictly non-blocking manner for unicast connections.

In one embodiment of this network each of the input switches IS1-IS4 and output switches OS1-OS4 are crossbar switches. The number of switches of input stage **110** and of output stage **120** can be denoted in general with the variable

$$\frac{N}{d},$$

where N is the total number of inlet links or outlet links. The number of middle switches in each middle stage is denoted by

$$\frac{N}{d}.$$

The size of each input switch IS1-IS4 can be denoted in general with the notation d*2d and each output switch OS1-OS4 can be denoted in general with the notation 2d*d. Likewise, the size of each switch in any of the middle stages can be denoted as 2d*2d. A switch as used herein can be either a crossbar switch, or a network of switches each of which in turn may be a crossbar switch or a network of switches. A symmetric multi-stage network can be represented with the notation $V_{mlink}(N, d, s)$, where N represents the total number of inlet links of all input switches (for example the links IL1-IL32), d represents the inlet links of each input switch or outlet links of each output switch, and s is the ratio of number of outgoing links from each input switch to the inlet links of each input switch.

Each of the

$$\frac{N}{d}$$

input switches IS1-IS16 are connected to exactly d switches in middle stage **130** through two links each for a total of 2×d links (for example input switch IS1 is connected to middle switch MS(**1,1**) through the links ML(**1,1**), ML(**1,2**), and also connected to middle switch MS(**1,2**) through the links ML(**1,3**) and ML(**1,4**)). The middle links which connect switches in the same row in two successive middle stages are called hereinafter straight middle links; and the middle links which connect switches in different rows in two successive middle stages are called hereinafter cross middle links. For example, the middle links ML(**1,1**) and ML(**1,2**) connect input switch IS1 and middle switch MS(**1,1**), so middle links ML(**1,1**) and ML(**1,2**) are straight middle links; where as the middle links ML(**1,3**) and ML(**1,4**) connect input switch IS1 and middle switch MS(**1,2**), since input switch IS1 and middle switch MS(**1,2**) belong to two different rows in diagram **100A** of FIG. 1A, middle links ML(**1,3**) and ML(**1,4**) are cross middle links.

Each of the

$$\frac{N}{d}$$

middle switches MS(**1,1**)-MS(**1,16**) in the middle stage **130** are connected from exactly d input switches through two links each for a total of 2×d links (for example the links ML(**1,1**)

and ML(**1,2**) are connected to the middle switch MS(**1,1**) from input switch IS1, and the links ML(**1,7**) and ML(**1,8**) are connected to the middle switch MS(**1,1**) from input switch IS2) and also are connected to exactly d switches in middle stage **140** through two links each for a total of 2×d links (for example the links ML(**2,1**) and ML(**2,2**) are connected from middle switch MS(**1,1**) to middle switch MS(**2,1**), and the links ML(**2,3**) and ML(**2,4**) are connected from middle switch MS(**1,1**) to middle switch MS(**2,3**)).

Each of the

$$\frac{N}{d}$$

middle switches MS(**2,1**)-MS(**2,16**) in the middle stage **140** are connected from exactly d input switches through two links each for a total of 2×d links (for example the links ML(**2,1**) and ML(**2,2**) are connected to the middle switch MS(**2,1**) from input switch MS(**1,1**), and the links ML(**1,11**) and ML(**1,12**) are connected to the middle switch MS(**2,1**) from input switch MS(**1,3**)) and also are connected to exactly d switches in middle stage **150** through two links each for a total of 2×d links (for example the links ML(**3,1**) and ML(**3,2**) are connected from middle switch MS(**2,1**) to middle switch MS(**3,1**), and the links ML(**3,3**) and ML(**3,4**) are connected from middle switch MS(**2,1**) to middle switch MS(**3,5**)).

Each of the

$$\frac{N}{d}$$

middle switches MS(**3,1**)-MS(**3,16**) in the middle stage **150** are connected from exactly d input switches through two links each for a total of 2×d links (for example the links ML(**3,1**) and ML(**3,2**) are connected to the middle switch MS(**3,1**) from input switch MS(**2,1**), and the links ML(**2,19**) and ML(**2,20**) are connected to the middle switch MS(**3,1**) from input switch MS(**2,5**)) and also are connected to exactly d switches in middle stage **160** through two links each for a total of 2×d links (for example the links ML(**4,1**) and ML(**4,2**) are connected from middle switch MS(**3,1**) to middle switch MS(**4,1**), and the links ML(**4,3**) and ML(**4,4**) are connected from middle switch MS(**3,1**) to middle switch MS(**4,9**)).

Each of the

$$\frac{N}{d}$$

middle switches MS(**4,1**)-MS(**4,16**) in the middle stage **160** are connected from exactly d input switches through two links each for a total of 2×d links (for example the links ML(**4,1**) and ML(**4,2**) are connected to the middle switch MS(**4,1**) from input switch MS(**3,1**), and the links ML(**4,35**) and ML(**4,36**) are connected to the middle switch MS(**4,1**) from input switch MS(**3,9**)) and also are connected to exactly d switches in middle stage **170** through two links each for a total of 2×d links (for example the links ML(**5,1**) and ML(**5,2**) are connected from middle switch MS(**4,1**) to middle switch MS(**5,1**), and the links ML(**5,3**) and ML(**5,4**) are connected from middle switch MS(**4,1**) to middle switch MS(**5,9**)).

US 8,269,523 B2

11

Each of the

$$\frac{N}{d}$$

middle switches MS(**5,1**)-MS(**5,16**) in the middle stage **170** are connected from exactly d input switches through two links each for a total of 2×d links (for example the links ML(**5,1**) and ML(**5,2**) are connected to the middle switch MS(**5,1**) from input switch MS(**4,1**), and the links ML(**5,35**) and ML(**5,36**) are connected to the middle switch MS(**5,1**) from input switch MS(**4,9**)) and also are connected to exactly d switches in middle stage **180** through two links each for a total of 2×d links (for example the links ML(**6,1**) and ML(**6,2**) are connected from middle switch MS(**5,1**) to middle switch MS(**6,1**), and the links ML(**6,3**) and ML(**6,4**) are connected from middle switch MS(**5,1**) to middle switch MS(**6,5**)).

Each of the

$$\frac{N}{d}$$

middle switches MS(**6,1**)-MS(**6,16**) in the middle stage **180** are connected from exactly d input switches through two links each for a total of 2×d links (for example the links ML(**6,1**) and ML(**6,2**) are connected to the middle switch MS(**6,1**) from input switch MS(**5,1**), and the links ML(**6,19**) and ML(**6,20**) are connected to the middle switch MS(**6,1**) from input switch MS(**5,5**)) and also are connected to exactly d switches in middle stage **190** through two links each for a total of 2×d links (for example the links ML(**7,1**) and ML(**7,2**) are connected from middle switch MS(**6,1**) to middle switch MS(**7,1**), and the links ML(**7,3**) and ML(**7,4**) are connected from middle switch MS(**6,1**) to middle switch MS(**7,3**)).

Each of the

$$\frac{N}{d}$$

middle switches MS(**7,1**)-MS(**7,16**) in the middle stage **190** are connected from exactly d input switches through two links each for a total of 2×d links (for example the links ML(**7,1**) and ML(**7,2**) are connected to the middle switch MS(**7,1**) from input switch MS(**6,1**), and the links ML(**7,11**) and ML(**7,12**) are connected to the middle switch MS(**7,1**) from input switch MS(**6,3**)) and also are connected to exactly d switches in middle stage **120** through two links each for a total of 2×d links (for example the links ML(**8,1**) and ML(**8,2**) are connected from middle switch MS(**7,1**) to middle switch MS(**8,1**), and the links ML(**8,3**) and ML(**8,4**) are connected from middle switch MS(**7,1**) to middle switch OS2).

Each of the

$$\frac{N}{d}$$

middle switches OS1-OS16 in the middle stage **120** are connected from exactly d input switches through two links each for a total of 2×d links (for example the links ML(**8,1**) and ML(**8,2**) are connected to the output switch OS1 from input

12

switch MS(**7,1**), and the links ML(**8,7**) and ML(**7,8**) are connected to the output switch OS1 from input switch MS(**7,2**)).

Finally the connection topology of the network **100A** shown in FIG. **1A** is known to be back to back inverse Benes connection topology.

Referring to diagram **100B** in FIG. **1B**, is a folded version of the multi-link multi-stage network **100A** shown in FIG. **1A**. The network **100B** in FIG. **1B** shows input stage **110** and output stage **120** are placed together. That is input switch IS**1** and output switch OS**1** are placed together, input switch IS**2** and output switch OS**2** are placed together, and similarly input switch IS**16** and output switch OS**16** are placed together. All the right going middle links (hereinafter "forward connecting links") {i.e., inlet links IL**1**-IL**32** and middle links ML(**1,1**)-ML(**1,64**)} correspond to input switches IS**1**-IS**16**, and all the left going middle links (hereinafter "backward connecting links") {i.e., middle links ML(**8,1**)-ML(**8, 64**) and outlet links OL**1**-OL**32**} correspond to output switches OS**1**-OS**16**.

Middle stage **130** and middle stage **190** are placed together. That is middle switches MS(**1,1**) and MS(**7,1**) are placed together, middle switches MS(**1,2**) and MS(**7,2**) are placed together, and similarly middle switches MS(**1,16**) and MS(**7, 16**) are placed together. All the right going middle links {i.e., middle links ML(**1,1**)-ML(**1,64**) and middle links ML(**2,1**)-ML(**2,64**)} correspond to middle switches MS(**1,1**)-MS(**1, 16**), and all the left going middle links {i.e., middle links ML(**7,1**)-ML(**7,64**) and middle links ML(**8,1**) and ML(**8,64**)} correspond to middle switches MS(**7,1**)-MS(**7,16**).

Middle stage **140** and middle stage **180** are placed together. That is middle switches MS(**2,1**) and MS(**6,1**) are placed together, middle switches MS(**2,2**) and MS(**6,2**) are placed together, and similarly middle switches MS(**2,16**) and MS(**6, 16**) are placed together. All the right going middle links {i.e., middle links ML(**2,1**)-ML(**2,64**) and middle links ML(**3,1**)-ML(**3,64**)} correspond to middle switches MS(**2,1**)-MS(**2, 16**), and all the left going middle links {i.e., middle links ML(**6,1**)-ML(**6,64**) and middle links ML(**7,1**) and ML(**7,64**)} correspond to middle switches MS(**6,1**)-MS(**6,16**).

Middle stage **150** and middle stage **170** are placed together. That is middle switches MS(**3,1**) and MS(**5,1**) are placed together, middle switches MS(**3,2**) and MS(**5,2**) are placed together, and similarly middle switches MS(**3,16**) and MS(**5, 16**) are placed together. All the right going middle links {i.e., middle links ML(**3,1**)-ML(**3,64**) and middle links ML(**4,1**)-ML(**4,64**)} correspond to middle switches MS(**3,1**)-MS(**3, 16**), and all the left going middle links {i.e., middle links ML(**5,1**)-ML(**5,64**) and middle links ML(**6,1**) and ML(**6,64**)} correspond to middle switches MS(**5,1**)-MS(**5,16**).

Middle stage **160** is placed alone. All the right going middle links are the middle links ML(**4,1**)-ML(**4,64**) and all the left going middle links are middle links ML(**5,1**)-ML(**5, 64**).

In one embodiment, in the network **100B** of FIG. **1B**, the switches that are placed together are implemented as separate switches then the network **100B** is the generalized folded multi-link multi-stage network $V_{fold-mlink}(N_1, N_2, d, s)$ where $N_1=N_2=32$; d=2; and s=2 with nine stages as disclosed in PCT Application Serial No. PCT/US08/64604 that is incorporated by reference above. That is the switches that are placed together in input stage **110** and output stage **120** are implemented as a two by four switch and a four by two switch. For example the switch input switch IS**1** and output switch OS**1** are placed together; so input switch IS**1** is implemented as two by four switch with the inlet links IL**1** and IL**2** being the inputs of the input switch IS**1** and middle links ML(**1,1**)-ML(**1,4**) being the outputs of the input switch IS**1**; and output

US 8,269,523 B2

13

14

switch OS1 is implemented as four by two switch with the middle links ML(**8,1**), ML(**8,2**), ML(**8,7**) and ML(**8,8**) being the inputs of the output switch OS1 and outlet links OL1-OL2 being the outputs of the output switch OS1. Similarly in this embodiment of network **100**B all the switches that are placed together in each middle stage are implemented as separate switches.

Hypercube Topology Layout Schemes:

Referring to layout **100**C of FIG. **1**C, in one embodiment, there are sixteen blocks namely Block **1_2**, Block **3_4**, Block **5_6**, Block **7_8**, Block **9_10**, Block **11_12**, Block **13_14**, Block **15_16**, Block **17_18**, Block **19_20**, Block **21_22**, Block **23_24**, Block **25_26**, Block **27_28**, Block **29_30**, and Block **31_32**. Each block implements all the switches in one row of the network **100**B of FIG. **1**B, one of the key aspects of the current invention. For example Block **1_2** implements the input switch IS**1**, output Switch OS**1**, middle switch MS(**1,1**), middle switch MS(**7,1**), middle switch MS(**2,1**), middle switch MS(**6,1**), middle switch MS(**3,1**), middle switch MS(**5,1**), and middle switch MS(**4,1**). For the simplification of illustration, Input switch IS**1** and output switch OS**1** together are denoted as switch **1**; Middle switch MS(**1,1**) and middle switch MS(**7,1**) together are denoted by switch **2**; Middle switch MS(**2,1**) and middle switch MS(**6,1**) together are denoted by switch **3**; Middle switch MS(**3,1**) and middle switch MS(**5,1**) together are denoted by switch **4**; Middle switch MS(**4,1**) is denoted by switch **5**.

All the straight middle links are illustrated in layout **100**C of FIG. **1**C. For example in Block **1_2**, inlet links IL1-IL**2**, outlet links OL1-OL**2**, middle link ML(**1,1**), middle link ML(**1,2**), middle link ML(**8,1**), middle link ML(**8,2**), middle link ML(**2,1**), middle link ML(**2,2**), middle link ML(**7,1**), middle link ML(**7,2**), middle link ML(**3,1**), middle link ML(**3,2**), middle link ML(**6,1**), middle link ML(**6,2**), middle link ML(**4,1**), middle link ML(**4,2**), middle link ML(**5,1**) and middle link ML(**5,2**) are illustrated in layout **100**C of FIG. **1**C.

Even though it is not illustrated in layout **100**C of FIG. **1**C, in each block, in addition to the switches there may be Configurable Logic Blocks (CLB) or any arbitrary digital circuit (hereinafter "sub-integrated circuit block") depending on the applications in different embodiments. There are four quadrants in the layout **100**C of FIG. **1**C namely top-left, bottom-left, top-right and bottom-right quadrants. Top-left quadrant implements Block **1_2**, Block **3_4**, Block **5_6**, and Block **7_8**. Bottom-left quadrant implements Block **9_10**, Block **11_12**, Block **13_14**, and Block **15_16**. Top-right quadrant implements Block **17_18**, Block **19_20**, Block **21_22**, and Block **23_24**. Bottom-right quadrant implements Block **25_26**, Block **27_28**, Block **29_30**, and Block **31_32**. There are two halves in layout **100**C of FIG. **1**C namely left-half and right-half. Left-half consists of top-left and bottom-left quadrants. Right-half consists of top-right and bottom-right quadrants.

Recursively in each quadrant there are four sub-quadrants. For example in top-left quadrant there are four sub-quadrants namely top-left sub-quadrant, bottom-left sub-quadrant, top-right sub-quadrant and bottom-right sub-quadrant. Top-left sub-quadrant of top-left quadrant implements Block **1_2**. Bottom-left sub-quadrant of top-left quadrant implements Block **3_4**. Top-right sub-quadrant of top-left quadrant implements Block **5_6**. Finally bottom-right sub-quadrant of top-left quadrant implements Block **7_8**. Similarly there are two sub-halves in each quadrant. For example in top-left quadrant there are two sub-halves namely left-sub-half and right-sub-half. Left-sub-half of top-left quadrant implements Block **1_2** and Block **3_4**. Right-sub-half of top-left quadrant

implements Block **5_6** and Block **7_8**. Finally applicant notes that in each quadrant or half the blocks are arranged as a general binary hypercube. Recursively in larger multi-stage network $V_{fold-mlink}(N_1, N_2, d, s)$ where $N_1=N_2>32$, the layout in this embodiment in accordance with the current invention, will be such that the super-quadrants will also be arranged in d-ary hypercube manner. (In the embodiment of the layout **100**C of FIG. **1**C, it is binary hypercube manner since d=2, in the network $V_{fold-mlink}(N_1, N_2, d, s)$ **100**B of FIG. **1**B).

Layout **100**D of FIG. **1**D illustrates the inter-block links between switches **1** and **2** of each block. For example middle links ML(**1,3**), ML(**1,4**), ML(**8,7**), and ML(**8,8**) are connected between switch **1** of Block **1_2** and switch **2** of Block **3_4**. Similarly middle links ML(**1,7**), ML(**1,8**), ML(**8,3**), and ML(**8,4**) are connected between switch **2** of Block **1_2** and switch **1** of Block **3_4**. Applicant notes that the inter-block links illustrated in layout **100**D of FIG. **1**D can be implemented as vertical tracks in one embodiment. Also in one embodiment inter-block links are implemented as two different tracks (for example middle links ML(**1,4**) and ML(**8,8**) are implemented as two different tracks); or in an alternative embodiment inter-block links are implemented as a time division multiplexed single track (for example middle links ML(**1,4**) and ML(**8,8**) are implemented as a time division multiplexed single track).

Layout **100**E of FIG. **1**E illustrates the inter-block links between switches **2** and **3** of each block. For example middle links ML(**2,3**), ML(**2,4**), ML(**7,11**), and ML(**7,12**) are connected between switch **2** of Block **1_2** and switch **3** of Block **3_4**. Similarly middle links ML(**2,11**), ML(**2,12**), ML(**7,3**), and ML(**7,4**) are connected between switch **3** of Block **1_2** and switch **2** of Block **3_4**. Applicant notes that the inter-block links illustrated in layout **100**E of FIG. **1**E can be implemented as horizontal tracks in one embodiment. Also in one embodiment inter-block links are implemented as two different tracks (for example middle links ML(**2,12**) and ML(**7,4**) are implemented as two different tracks); or in an alternative embodiment inter-block links are implemented as a time division multiplexed single track (for example middle links ML(**2,12**) and ML(**7,4**) are implemented as a time division multiplexed single track).

Layout **100**F of FIG. **1**F illustrates the inter-block links between switches **3** and **4** of each block. For example middle links ML(**3,3**), ML(**3,4**), ML(**6,19**), and ML(**6,20**) are connected between switch **3** of Block **1_2** and switch **4** of Block **3_4**. Similarly middle links ML(**3,19**), ML(**3,20**), ML(**6,3**), and ML(**6,4**) are connected between switch **4** of Block **1_2** and switch **3** of Block **3_4**. Applicant notes that the inter-block links illustrated in layout **100**F of FIG. **1**F can be implemented as vertical tracks in one embodiment. Also in one embodiment inter-block links are implemented as two different tracks (for example middle links ML(**3,4**) and ML(**6,20**) are implemented as two different tracks); or in an alternative embodiment inter-block links are implemented as a time division multiplexed single track (for example middle links ML(**3,4**) and ML(**6,20**) are implemented as a time division multiplexed single track).

Layout **100**G of FIG. **1**G illustrates the inter-block links between switches **4** and **5** of each block. For example middle links ML(**4,3**), ML(**4,4**), ML(**5,35**), and ML(**5,36**) are connected between switch **4** of Block **1_2** and switch **5** of Block **3_4**. Similarly middle links ML(**4,35**), ML(**4,36**), ML(**5,3**), and ML(**5,4**) are connected between switch **5** of Block **1_2** and switch **4** of Block **3_4**. Applicant notes that the inter-block links illustrated in layout **100**G of FIG. **1**G can be implemented as horizontal tracks in one embodiment. Also in one embodiment inter-block links are implemented as two

US 8,269,523 B2

15   16

different tracks (for example middle links ML(**4,4**) and ML(**5,36**) are implemented as two different tracks); or in an alternative embodiment inter-block links are implemented as a time division multiplexed single track (for example middle links ML(**4,4**) and ML(**5,36**) are implemented as a time division multiplexed single track).

The complete layout for the network 100B of FIG. 1B is given by combining the links in layout diagrams of 100C, 100D, 100E, 100F, and 100G. Applicant notes that in the layout 100C of FIG. 1C, the inter-block links between switch 1 and switch 2 of corresponding blocks are vertical tracks as shown in layout 100D of FIG. 1D; the inter-block links between switch 2 and switch 3 of corresponding blocks are horizontal tracks as shown in layout 100E of FIG. 1E; the inter-block links between switch 3 and switch 4 of corresponding blocks are vertical tracks as shown in layout 100F of FIG. 1F; and finally the inter-block links between switch 4 and switch 5 of corresponding blocks are horizontal tracks as shown in layout 100G of FIG. 1G. The pattern is alternate vertical tracks and horizontal tracks. It continues recursively for larger networks of N>32 as will be illustrated later.

Some of the key aspects of the current invention are discussed. 1) All the switches in one row of the multi-stage network 100B are implemented in a single block. 2) The blocks are placed in such a way that all the inter-block links are either horizontal tracks or vertical tracks; 3) Since all the inter-block links are either horizontal or vertical tracks, all the inter-block links can be mapped on to island-style architectures in current commercial FPGA's; 4) The length of the longest wire is about half of the width (or length) of the complete layout (For example middle link ML(**4,4**) is about half the width of the complete layout).

In accordance with the current invention, the layout 100C in FIG. 1C can be recursively extended for any arbitrarily large generalized folded multi-link multi-stage network $V_{fold-mlink}(N_1, N_2, d, s)$ the sub-quadrants, quadrants, and super-quadrants are arranged in d-ary hypercube manner and also the inter-blocks are accordingly connected in d-ary hypercube topology. Even though all the embodiments in the current invention are illustrated for $N_1 = N_2$, the embodiments can be extended for $N_1 \neq N_2$. Referring to layout 100H of FIG. 1H, illustrates the extension of layout 100C for the network $V_{fold-mlink}(N_1, N_2, d, s)$ where $N_1 = N_2 = 128$; d=2; and s=2. There are four super-quadrants in layout 100H namely top-left super-quadrant, bottom-left super-quadrant, top-right super-quadrant, bottom-right super-quadrant. Total number of blocks in the layout 100H is sixty four. Top-left super-quadrant implements the blocks from block **1_2** to block **31_32**. Each block in all the super-quadrants has two more switches namely switch 6 and switch 7 in addition to the switches [1-5] illustrated in layout 100C of FIG. 1C. The inter-block link connection topology is the exactly the same between the switches 1 and 2; switches 2 and 3; switches 3 and 4; switches 4 and 5 as it is shown in the layouts of FIG. 1D, FIG. 1E, FIG. 1F, and FIG. 1G respectively.

Bottom-left super-quadrant implements the blocks from block **33_34** to block **63_64**. Top-right super-quadrant implements the blocks from block **65_66** to block **95_96**. And bottom-right super-quadrant implements the blocks from block **97_98** to block **127_128**. In all these three super-quadrants also, the inter-block link connection topology is the exactly the same between the switches 1 and 2; switches 2 and 3; switches 3 and 4; switches 4 and 5 as that of the top-left super-quadrant.

Recursively in accordance with the current invention, the inter-block links connecting the switch 5 and switch 6 will be vertical tracks between the corresponding switches of top-left

super-quadrant and bottom-left super-quadrant. And similarly the inter-block links connecting the switch 5 and switch 6 will be vertical tracks between the corresponding switches of top-right super-quadrant and bottom-right super-quadrant. The inter-block links connecting the switch 6 and switch 7 will be horizontal tracks between the corresponding switches of top-left super-quadrant and top-right super-quadrant. And similarly the inter-block links connecting the switch 6 and switch 7 will be horizontal tracks between the corresponding switches of bottom-left super-quadrant and bottom-right super-quadrant.

Referring to diagram 100I of FIG. 1I illustrates a high-level implementation of Block **1_2** (Each of the other blocks have similar implementation) of the layout 100C of FIG. 1C which represents a generalized folded multi-link multi-stage network $V_{fold-mlink}(N_1, N_2, d, s)$ where $N_1 = N_2 = 32$; d=2; and s=2. Block **1_2** in 100I illustrates both the intra-block and inter-block links connected to Block **1_2**. The layout diagram 100I corresponds to the embodiment where the switches that are placed together are implemented as separate switches in the network 100B of FIG. 1B. As noted before then the network 100B is the generalized folded multi-link multi-stage network $V_{fold-mlink}(N_1, N_2, d, s)$ where $N_1 = N_2 = 32$; d=2; and s=2 with nine stages as disclosed in PCT Application Serial No. PCT/US08/64604 that is incorporated by reference above.

That is the switches that are placed together in Block **1_2** as shown in FIG. 1I are namely input switch IS1 and output switch OS1 belonging to switch 1, illustrated by dotted lines, (as noted before switch 1 is for illustration purposes only, in practice the switches implemented are input switch IS1 and output switch OS1); middle switch MS(**1,1**) and middle switch MS(**7,1**) belonging to switch 2; middle switch MS(**2,1**) and middle switch MS(**6,1**) belonging to switch 3; middle switch MS(**3,1**) and middle switch MS(**5,1**) belonging to switch 4; And middle switch MS(**4,1**) belonging to switch 5.

Input switch IS1 is implemented as two by four switch with the inlet links IL1 and IL2 being the inputs of the input switch IS1 and middle links ML(**1,1**)-ML(**1,4**) being the outputs of the input switch IS1; and output switch OS1 is implemented as four by two switch with the middle links ML(**8,1**)-ML(**8,4**) being the inputs of the output switch OS1 and outlet links OL1-OL2 being the outputs of the output switch OS1.

Middle switch MS(**1,1**) is implemented as four by four switch with the middle links ML(**1,1**), ML(**1,2**), ML(**1,7**) and ML(**1,8**) being the inputs and middle links ML(**2,1**)-ML(**2,4**) being the outputs; and middle switch MS(**7,1**) is implemented as four by four switch with the middle links ML(**7,1**), ML(**7, 2**), ML(**7,11**) and ML(**7,12**) being the inputs and middle links ML(**8,1**)-ML(**8,4**) being the outputs. Similarly all the other middle switches are also implemented as four by four switches as illustrated in 100I of FIG. 1I.

Now the VLSI layouts of generalized multi-link multi-stage network $V_{mlink}(N_1, N_2, d, s)$ where $N_1 = N_2 < 32$; d=2; s=2 and its corresponding version of folded generalized multi-link multi-stage network $V_{fold-mlink}(N_1, N_2, d, s)$ where $N_1 = N_2 < 32$; d=2; s=2 are discussed. Referring to diagram 200A1 of FIG. 2A1 is generalized multi-link multi-stage network $V_{mlink}(N_1, N_2, d, s)$ where $N_1 = N_2 = 2$; d=2. Diagram 200A2 of FIG. 2A2 illustrates the corresponding folded generalized multi-link multi-stage network $V_{fold-mlink}(N_1, N_2, d, s)$ where $N_1 = N_2 = 2$; d=2, version of the diagram 200A1 of FIG. 2A1. Layout 200A3 of FIG. 2A3 illustrates the VLSI layout of the network 200A2 of FIG. 2A2. There is only one block i.e., Block **1_2** comprising switch 1. Just like in the layout 100C of FIG. 1C, switch 1 consists of input switch IS1 and output switch OS1.

US 8,269,523 B2

17   18

Referring to diagram **200B1** of FIG. **2B1** is generalized multi-link multi-stage network $V_{mlink}(N_1, N_2, d, s)$ where $N_1=N_2=4$; $d=2$; $s=2$. Diagram **200B2** of FIG. **2B2** illustrates the corresponding folded generalized multi-link multi-stage network $V_{fold-mlink}(N_1, N_2, d, s)$ where $N_1=N_2=4$; $d=2$; $s=2$, version of the diagram **200B1** of FIG. **2B1**. Layout **200B3** of FIG. **2B3** illustrates the VLSI layout of the network **200B2** of FIG. **2B2**. There are two blocks i.e., Block **1_2** and Block **3_4** each comprising switch **1** and switch **2**. Switch **1** in each block consists of the corresponding input switch and output switch. For example switch **1** in Block **1_2** consists of input switch IS1 and output switch OS1. Similarly switch **2** in Block **1_2** consists of middle switch (**1,1**). Layout **200B4** of FIG. **2B4** illustrates the inter-block links of the VLSI layout diagram **200B3** of FIG. **2B3**. For example middle links ML(**1, 4**) and ML(**2,8**). It must be noted that all the inter-block links are vertical tracks in this layout. (Alternatively all the inter-blocks can also be implemented as horizontal tracks).

Referring to diagram **200C11** of FIG. **2C11** is generalized multi-link multi-stage network $V_{mlink}(N_1, N_2, d, s)$ where $N_1=N_2=8$; $d=2$; $s=2$. Diagram **200C12** of FIG. **2C12** illustrates the corresponding folded generalized multi-link multi-stage network $V_{fold-mlink}(N_1, N_2, d, s)$ where $N_1=N_2=8$; $d=2$; $s=2$, version of the diagram **200C11** of FIG. **2C11**. Layout **200C21** of FIG. **2C21** illustrates the VLSI layout of the network **200C12** of FIG. **2C12**. There are four blocks i.e., Block **1_2**, Block **3_4**, Block **5_6**, and Block **7_8** each comprising switch **1**, switch **2** and switch **3**. For example switch **1** in Block **1_2** consists of input switch IS1 and output switch OS1; Switch **2** in Block **1_2** consists of MS(**1,1**) and MS(**3,1**). Switch **3** in Block **1_2** consists of MS(**2,1**).

Layout **200C22** of FIG. **2C22** illustrates the inter-block links between the switch **1** and switch **2** of the VLSI layout diagram **200C21** of FIG. **2C21**. For example middle links ML(**1,4**) and ML(**4,8**) are connected between Block **1_2** and Block **3_4**. It must be noted that all the inter-block links between switch **1** and switch **2** of all blocks are vertical tracks in this layout. Layout **200C23** of FIG. **2C23** illustrates the inter-block links between the switch **2** and switch **3** of the VLSI layout diagram **200C21** of FIG. **2C21**. For example middle links ML(**2,12**) and ML(**3,4**) are connected between Block **1_2** and Block **5_6**. It must be noted that all the inter-block links between switch **2** and switch **3** of all blocks are horizontal tracks in this layout

Referring to diagram **200D1** of FIG. **2D1** is generalized multi-link multi-stage network $V_{mlink}(N_1, N_2, d, s)$ where $N_1=N_2=16$; $d=2$; $s=2$. Diagram **200D2** of FIG. **2D2** illustrates the corresponding folded generalized multi-link multi-stage network $V_{fold-mlink}(N_1, N_2, d, s)$ where $N_1=N_2=16$; $d=2$; $s=2$, version of the diagram **200D1** of FIG. **2D1**. Layout **200D3** of FIG. **2D3** illustrates the VLSI layout of the network **200D2** of FIG. **2D2**. There are eight blocks i.e., Block **1_2**, Block **3_4**, Block **5_6**, Block **7_8**, Block **9_10**, Block **11_12**, Block **13_14** and Block **15_16** each comprising switch **1**, switch **2**, switch **3** and switch **4**. For example switch **1** in Block **1_2** consists of input switch IS1 and output switch OS1; Switch **2** in Block **1_2** consists of MS(**1,1**) and MS(**5,1**). Switch **3** in Block **1_2** consists of MS(**2,1**) and MS(**4,1**), and switch **4** in Block **1_2** consists of MS(**3,1**).

Layout **200D4** of FIG. **2D4** illustrates the inter-block links between the switch **1** and switch **2** of the VLSI layout diagram **200D3** of FIG. **2D3**. For example middle links ML(**1,4**) and ML(**6,8**) are connected between Block **1_2** and Block **3_4**. It must be noted that all the inter-block links between switch **1** and switch **2** of all blocks are vertical tracks in this layout. Layout **200D5** of FIG. **2D5** illustrates the inter-block links between the switch **2** and switch **3** of the VLSI layout diagram

200D3 of FIG. **2D3**. For example middle links ML(**2,12**) and ML(**5,4**) are connected between Block **1_2** and Block **5_6**. It must be noted that all the inter-block links between switch **2** and switch **3** of all blocks are horizontal tracks in this layout. Layout **200D6** of FIG. **2D6** illustrates the inter-block links between the switch **3** and switch **4** of the VLSI layout diagram **200D3** of FIG. **2D3**. For example middle links ML(**3,4**) and ML(**4,20**) are connected between Block **1_2** and Block **9_10**. It must be noted that all the inter-block links between switch **3** and switch **4** of all blocks are vertical tracks in this layout.

Generalized Multi-link Butterfly Fat Tree Network Embodiment

In another embodiment in the network **100B** of FIG. **1B**, the switches that are placed together are implemented as combined switch then the network **100B** is the generalized multi-link butterfly fat tree network $V_{mlink-bft}(N_1, N_2, d, s)$ where $N_1=N_2=32$; $d=2$; and $s=2$ with five stages as disclosed in PCT Application Serial No. PCT/US08/64603 that is incorporated by reference above. That is the switches that are placed together in input stage **110** and output stage **120** are implemented as a six by six switch. For example the input switch IS1 and output switch OS1 are placed together; so input switch IS1 and output OS1 are implemented as a six by six switch with the inlet links IL1, IL2, ML(**8,1**), ML(**8,2**), ML(**8,7**) and ML(**8,8**) being the inputs of the combined switch (denoted as IS1&OS1) and middle links ML(**1,1**), ML(**1,2**), ML(**1,3**), ML(**1,4**), OL1 and OL2 being the outputs of the combined switch IS1&OS1. Similarly in this embodiment of network **100B** all the switches that are placed together are implemented as a combined switch.

Layout diagrams **100C** in FIG. **1C**, **100D** in FIG. **1D**, **100E** in FIG. **1E**, **100F** in FIG. **1G** are also applicable to generalized multi-link butterfly fat tree network $V_{mlink-bft}(N_1, N_2, d, s)$ where $N_1=N_2=32$; $d=2$; and $s=2$ with five stages. The layout **100C** in FIG. **1C** can be recursively extended for any arbitrarily large generalized multi-link butterfly fat tree network $V_{mlink-bft}(N_1, N_2, d, s)$. Accordingly layout **100H** of FIG. **1H** is also applicable to generalized multi-link butterfly fat tree network $V_{mlink-bft}(N_1, N_2, d, s)$.

Referring to diagram **100J** of FIG. **1J** illustrates a high-level implementation of Block **1_2** (Each of the other blocks have similar implementation) of the layout **100C** of FIG. **1C** which represents a generalized multi-link butterfly fat tree network $V_{mlink-bft}(N_1, N_2, d, s)$ where $N_1=N_2=32$; $d=2$; and $s=2$. Block **1_2** in **100J** illustrates both the intra-block and inter-block links. The layout diagram **100J** corresponds to the embodiment where the switches that are placed together are implemented as combined switch in the network **100B** of FIG. **1B**. As noted before then the network **100B** is the generalized multi-link butterfly fat tree network $V_{mlink-bft}(N_1, N_2, d, s)$ where $N_1=N_2=32$; $d=2$; and $s=2$ with five stages as disclosed in PCT Application Serial No. PCT/US08/64603 that is incorporated by reference above.

That is the switches that are placed together in Block **1_2** as shown in FIG. **1J** are namely the combined input and output switch IS1&OS1 belonging to switch **1**, illustrated by dotted lines, (as noted before switch **1** is for illustration purposes only, in practice the switch implemented is combined input and output switch IS1&OS1); middle switch MS(**1,1**) belonging to switch **2**; middle switch MS(**2,1**) belonging to switch **3**; middle switch MS(**3,1**) belonging to switch **4**; And middle switch MS(**4,1**) belonging to switch **5**.

Combined input and output switch IS1&OS1 is implemented as six by six switch with the inlet links IL1, IL2 and

US 8,269,523 B2

19

20

ML(8,1)-ML(8,4) being the inputs and middle links ML(1,1)-ML(1,4), and outlet links OL1-OL2 being the outputs.

Middle switch MS(1,1) is implemented as eight by eight switch with the middle links ML(1,1), ML(1,2), ML(1,7), ML(1,8), ML(7,1), ML(7,2), ML(7,11) and ML(7,12) being the inputs and middle links ML(2,1)-ML(2,4) and middle links ML(8,1)-ML(8,4) being the outputs. Similarly all the other middle switches are also implemented as eight by eight switches as illustrated in 100J of FIG. 1J. Applicant observes that in middle switch MS(1,1) any one of the right going middle links can be switched to any one of the left going middle links and hereinafter middle switch MS(1,1) provides U-turn links. In general, in the network $V_{mlink-bft}(N_1, N_2, d, s)$ each input switch, each output switch and each middle switch provides U-turn links.

In another embodiment, middle switch MS(1,1) (or the middle switches in any of the middle stage excepting the root middle stage) of Block 1_2 of $V_{mlink-bft}(N_1, N_2, d, s)$ can be implemented as a four by eight switch and a four by four switch to save cross points. This is because the left going middle links of these middle switches are never setup to the right going middle links. For example, in middle switch MS(1,1) of Block 1_2 as shown FIG. 1J, the left going middle links namely ML(7,1), ML(7,2), ML(7,11), and ML(7,12) are never switched to the right going middle links ML(2,1), ML(2,2), ML(2,3), and ML(2,4). And hence to implement MS(1,1) two switches namely: 1) a four by eight switch with the middle links ML(1,1), ML(1,2), ML(1,7), and ML(1,8) as inputs and the middle links ML(2,1), ML(2,2), ML(2,3), ML(2,4), ML(8,1), ML(8,2), ML(8,3), and ML(8,4) as outputs and 2) a four by four switch with the middle links ML(7,1), ML(7,2), ML(7,11), and ML(7,12) as inputs and the middle links ML(8,1), ML(8,2), ML(8,3), and ML(8,4) as outputs are sufficient without loosing any connectivity of the embodiment of MS(1,1) being implemented as an eight by eight switch as described before.)

Generalized Multi-Stage Network Embodiment

In one embodiment, in the network 100B of FIG. 1B, the switches that are placed together are implemented as two separate switches in input stage 110 and output stage 120; and as four separate switches in all the middle stages, then the network 100B is the generalized folded multi-stage network $V_{fold}(N_1, N_2, d, s)$ where $N_1=N_2=32$; d=2; and s=2 with nine stages as disclosed in PCT Application Serial No. PCT/US08/64604 that is incorporated by reference above. That is the switches that are placed together in input stage 110 and output stage 120 are implemented as a two by four switch and a four by two switch respectively. For example the switch input switch IS1 and output switch OS1 are placed together; so input switch IS1 is implemented as two by four switch with the inlet links IL1 and IL2 being the inputs and middle links ML(1,1)-ML(1,4) being the outputs; and output switch OS1 is implemented as four by two switch with the middle links ML(8,1), ML(8,4), ML(8,7) and ML(8,8) being the inputs and outlet links OL1-OL2 being the outputs.

The switches, corresponding to the middle stages that are placed together are implemented as four two by two switches. For example middle switches MS(1,1), MS(1,17), MS(7,1), and MS(7,17) are placed together; so middle switch MS(1,1) is implemented as two by two switch with middle links ML(1,1) and ML(1,7) being the inputs and middle links ML(2,1) and ML(2,3) being the outputs; middle switch MS(1,17) is implemented as two by two switch with the middle links ML(1,2) and ML(1,8) being the inputs and middle links ML(2,2) and ML(2,4) being the outputs; middle switch

MS(7,1) is implemented as two by two switch with middle links ML(7,1) and ML(7,11) being the inputs and middle links ML(8,1) and ML(8,3) being the outputs; And middle switch MS(7,17) is implemented as two by two switch with the middle links ML(7,2) and ML(7,12) being the inputs and middle links ML(8,2) and ML(8,4) being the outputs; Similarly in this embodiment of network 100B all the switches that are placed together are implemented as separate switches.

Layout diagrams 100C in FIG. 1C, 100D in FIG. 1D, 100E in FIG. 1E, 100F in FIG. 1G are also applicable to generalized folded multi-stage network $V_{fold}(N_1, N_2, d, s)$ where $N_1=N_2=32$; d=2; and s=2 with nine stages. The layout 100C in FIG. 1C can be recursively extended for any arbitrarily large generalized folded multi-stage network $V_{fold}(N_1, N_2, d, s)$. Accordingly layout 100H of FIG. 1H is also applicable to generalized folded multi-stage network $V_{fold}(N_1, N_2, d, s)$.

Referring to diagram 100K of FIG. 1K illustrates a high-level implementation of Block 1_2 (Each of the other blocks have similar implementation) of the layout 100C of FIG. 1C which represents a generalized folded multi-stage network $V_{fold}(N_1, N_2, d, s)$ where $N_1=N_2=32$; d=2; and s=2. Block 1_2 in 100K illustrates both the intra-block and inter-block links. The layout diagram 100K corresponds to the embodiment where the switches that are placed together are implemented as separate switches in the network 100B of FIG. 1B. As noted before then the network 100B is the generalized folded multi-stage network $V_{fold}(N_1, N_2, d, s)$ where $N_1=N_2=32$; d=2; and s=2 with nine stages as disclosed in PCT Application Serial No. PCT/US08/64604 that is incorporated by reference above.

That is the switches that are placed together in Block 1_2 as shown in FIG. 1K are namely the input switch IS1 and output switch OS1 belonging to switch 1, illustrated by dotted lines, (as noted before switch 1 is for illustration purposes only, in practice the switches implemented are input switch IS1 and output switch OS1); middle switches MS(1,1), MS(1,17), MS(7,1) and MS(7,17) belonging to switch 2; middle switches MS(2,1), MS(2,17), MS(6,1) and MS(6,17) belonging to switch 3; middle switches MS(3,1), MS(3,17), MS(5,1) and MS(5,17) belonging to switch 4; And middle switches MS(4,1), and MS(4,17) belonging to switch 5.

Input switch IS1 and output switch OS1 are placed together; so input switch IS1 is implemented as two by four switch with the inlet links IL1 and IL2 being the inputs and middle links ML(1,1)-ML(1,4) being the outputs; and output switch OS1 is implemented as four by two switch with the middle links ML(8,1), ML(8,4), ML(8,7) and ML(8,8) being the inputs and outlet links OL1-OL2 being the outputs.

Middle switches MS(1,1), MS(1,17), MS(7,1), and MS(7,17) are placed together; so middle switch MS(1,1) is implemented as two by two switch with middle links ML(1,1) and ML(1,7) being the inputs and middle links ML(2,1) and ML(2,3) being the outputs; middle switch MS(1,17) is implemented as two by two switch with the middle links ML(1,2) and ML(1,8) being the inputs and middle links ML(2,2) and ML(2,4) being the outputs; middle switch MS(7,1) is implemented as two by two switch with middle links ML(7,1) and ML(7,11) being the inputs and middle links ML(8,1) and ML(8,3) being the outputs; And middle switch MS(7,17) is implemented as two by two switch with the middle links ML(7,2) and ML(7,12) being the inputs and middle links ML(8,2) and ML(8,4) being the outputs Similarly all the other middle switches are also implemented as two by two switches as illustrated in 100K of FIG. 1K.

Generalized Multi-Stage Network Embodiment with S=1

In one embodiment, in the network 100B of FIG. 1B (where it is implemented with s=1), the switches that are

US 8,269,523 B2

21   22

placed together are implemented as two separate switches in input stage **110** and output stage **120**; and as two separate switches in all the middle stages, then the network **100B** is the generalized folded multi-stage network $V_{fold}(N_1, N_2, d, s)$ where $N_1=N_2=32$; d=2; and s=1 with nine stages as disclosed in PCT Application Serial No. PCT/US08/64604 that is incorporated by reference above. That is the switches that are placed together in input stage **110** and output stage **120** are implemented as a two by two switch and a two by two switch. For example the switch input switch IS1 and output switch OS1 are placed together; so input switch IS1 is implemented as two by two switch with the inlet links IL**1** and IL**2** being the inputs and middle links ML(**1,1**)-ML(**1,2**) being the outputs; and output switch OS1 is implemented as two by two switch with the middle links ML(**8,1**) and ML(**8,3**) being the inputs and outlet links OL**1**-OL**2** being the outputs.

The switches, corresponding to the middle stages that are placed together are implemented as two, two by two switches. For example middle switches MS(**1,1**) and MS(**7,1**) are placed together; so middle switch MS(**1,1**) is implemented as two by two switch with middle links ML(**1,1**) and ML(**1,3**) being the inputs and middle links ML(**2,1**) and ML(**2,2**) being the outputs; middle switch MS(**7,1**) is implemented as two by two switch with middle links ML(**7,1**) and ML(**7,5**) being the inputs and middle links ML(**8,1**) and ML(**8,2**) being the outputs; Similarly in this embodiment of network **100B** all the switches that are placed together are implemented as two separate switches.

Layout diagrams **100C** in FIG. 1C, **100D** in FIG. 1D, **100E** in FIG. 1E, **100F** in FIG. 1G are also applicable to generalized folded multi-stage network $V_{fold}(N_1, N_2, d, s)$ where $N_1=N_2=32$; d=2; and s=1 with nine stages. The layout **100C** in FIG. 1C can be recursively extended for any arbitrarily large generalized folded multi-stage network $V_{fold}(N_1, N_2, d, s)$. Accordingly layout **100H** of FIG. 1H is also applicable to generalized folded multi-stage network $V_{fold}(N_1, N_2, d, s)$.

Referring to diagram **100K1** of FIG. 1K1 illustrates a high-level implementation of Block **1_2** (Each of the other blocks have similar implementation) for the layout **100C** of FIG. 1C when s=1 which represents a generalized folded multi-stage network $V_{fold}(N_1, N_2, d, s)$ where $N_1=N_2=32$; d=2; and s=1 (All the double links are replaced by single links when s=1). Block **1_2** in **100K1** illustrates both the intra-block and inter-block links. The layout diagram **100K1** corresponds to the embodiment where the switches that are placed together are implemented as separate switches in the network **100B** of FIG. 1B when s=1. As noted before then the network **100B** is the generalized folded multi-stage network $V_{fold}(N_1, N_2, d, s)$ where $N_1=N_2=32$; d=2; and s=1 with nine stages as disclosed in PCT Application Serial No. PCT/US08/64604 that is incorporated by reference above.

That is the switches that are placed together in Block **1_2** as shown in FIG. 1K1 are namely the input switch IS1 and output switch OS1 belonging to switch **1**, illustrated by dotted lines, (as noted before switch **1** is for illustration purposes only, in practice the switches implemented are input switch IS1 and output switch OS1); middle switches MS(**1,1**) and MS(**7,1**) belonging to switch **2**; middle switches MS(**2,1**) and MS(**6,1**) belonging to switch **3**; middle switches MS(**3,1**) and MS(**5,1**) belonging to switch **4**; And middle switch MS(**4,1**) belonging to switch **5**.

Input switch IS1 and output switch OS1 are placed together; so input switch IS1 is implemented as two by two switch with the inlet links IL**1** and IL**2** being the inputs and middle links ML(**1,1**)-ML(**1,2**) being the outputs; and output switch OS1 is implemented as two by two switch with the

middle links ML(**8,1**) and ML(**8,3**) being the inputs and outlet links OL**1**-OL**2** being the outputs.

Middle switches MS(**1,1**) and MS(**7,1**) are placed together; so middle switch MS(**1,1**) is implemented as two by two switch with middle links ML(**1,1**) and ML(**1,3**) being the inputs and middle links ML(**2,1**) and ML(**2,2**) being the outputs; And middle switch MS(**7,1**) is implemented as two by two switch with middle links ML(**7,1**) and ML(**7,5**) being the inputs and middle links ML(**8,1**) and ML(**8,2**) being the outputs. Similarly all the other middle switches are also implemented as two by two switches as illustrated in **100K1** of FIG. **1K1**.

Generalized Butterfly Fat Tree Network Embodiment

In another embodiment in the network **100B** of FIG. 1B, the switches that are placed together are implemented as two combined switches then the network **100B** is the generalized butterfly fat tree network $V_{bft}(N_1, N_2, d, s)$ where $N_1=N_2=32$; d=2; and s=2 with five stages as disclosed in PCT Application Serial No. PCT/US08/64603 that is incorporated by reference above. That is the switches that are placed together in input stage **110** and output stage **120** are implemented as a six by six switch. For example the input switch IS1 and output switch OS1 are placed together; so input output switch IS1&OS1 are implemented as a six by six switch with the inlet links IL**1**, IL**2**, ML(**8,1**), ML(**8,2**), ML(**8,7**) and ML(**8,8**) being the inputs of the combined switch (denoted as IS1&OS1) and middle links ML(**1,1**), ML(**1,2**), ML(**1,3**), ML(**1,4**), OL**1** and OL**2** being the outputs of the combined switch IS1&OS1.

The switches, corresponding to the middle stages that are placed together are implemented as two four by four switches. For example middle switches MS(**1,1**) and MS(**1,17**) are placed together; so middle switch MS(**1,1**) is implemented as four by four switch with middle links ML(**1,1**), ML(**1,7**), ML(**7,1**) and ML(**7,11**) being the inputs and middle links ML(**2,1**), ML(**2,3**), ML(**8,1**) and ML(**8,3**) being the outputs; middle switch MS(**1,17**) is implemented as four by four switch with the middle links ML(**1,2**), ML(**1,8**), ML(**7,2**) and ML(**7,12**) being the inputs and middle links ML(**2,2**), ML(**2,4**), ML(**8,2**) and ML(**8,4**) being the outputs. Similarly in this embodiment of network **100B** all the switches that are placed together are implemented as a two combined switches.

Layout diagrams **100C** in FIG. 1C, **100D** in FIG. 1D, **100E** in FIG. 1E, **100F** in FIG. 1G are also applicable to generalized butterfly fat tree network $V_{bft}(N_1, N_2, d, s)$ where $N_1=N_2=32$; d=2; and s=2 with five stages. The layout **100C** in FIG. 1C can be recursively extended for any arbitrarily large generalized butterfly fat tree network $V_{bft}(N_1, N_2, d, s)$. Accordingly layout **100H** of FIG. 1H is also applicable to generalized butterfly fat tree network $V_{bft}(N_1, N_2, d, s)$.

Referring to diagram **100L** of FIG. 1L illustrates a high-level implementation of Block **1_2** (Each of the other blocks have similar implementation) of the layout **100C** of FIG. 1C which represents a generalized butterfly fat tree network $V_{bft}(N_1, N_2, d, s)$ where $N_1=N_2=32$; d=2; and s=2. Block **1_2** in **100L** illustrates both the intra-block and inter-block links. The layout diagram **100L** corresponds to the embodiment where the switches that are placed together are implemented as two combined switches in the network **100B** of FIG. 1B. As noted before then the network **100B** is the generalized butterfly fat tree network $V_{bft}(N_1, N_2, d, s)$ where $N_1=N_2=32$; d=2; and s=2 with five stages as disclosed in PCT Application Serial No. PCT/US08/64603 that is incorporated by reference above.

US 8,269,523 B2

23   24

That is the switches that are placed together in Block **1_2** as shown in FIG. **1L** are namely the combined input and output switch IS1&OS1 belonging to switch **1**, illustrated by dotted lines, (as noted before switch **1** is for illustration purposes only, in practice the switch implemented is combined input and output switch IS1&OS1); middle switch MS(**1,1**) and MS(**1,17**) belonging to switch **2**; middle switch MS(**2,1**) and MS(**2,17**) belonging to switch **3**; middle switch MS(**3,1**) and MS(**3,17**) belonging to switch **4**; And middle switch MS(**4,1**) belonging to switch **5**.

Combined input and output switch IS1&OS1 is implemented as six by six switch with the inlet links IL**1**, IL**2**, ML(**8,1**), ML(**8,2**), ML(**8,7**) and ML(**8,8**) being the inputs and middle links ML(**1,1**)-ML(**1,4**) and outlet links OL1-OL2 being the outputs.

Middle switch MS(**1,1**) is implemented as four by four switch with middle links ML(**1,1**), ML(**1,7**), ML(**7,1**) and ML(**7,11**) being the inputs and middle links ML(**2,1**), ML(**2,3**), ML(**8,1**) and ML(**8,3**) being the outputs; And middle switch MS(**1,17**) is implemented as four by four switch with the middle links ML(**1,2**), ML(**1,8**), ML(**7,2**) and ML(**7,12**) being the inputs and middle links ML(**2,2**), ML(**2,4**), ML(**8,2**) and ML(**8,4**) being the outputs. Similarly all the other middle switches are also implemented as two four by four switches as illustrated in **100L** of FIG. **1L**. Applicant observes that in middle switch MS(**1,1**) any one of the right going middle links can be switched to any one of the left going middle links and hereinafter middle switch MS(**1,1**) provides U-turn links. In general, in the network V_{bft}(N$_1$, N$_2$, d, s) each input switch, each output switch and each middle switch provides U-turn links.

In another embodiment, middle switch MS(**1,1**) (or the middle switches in any of the middle stage excepting the root middle stage) of Block **1_2** of V_{bft}(N$_1$, N$_2$, d, s) can be implemented as a two by four switch and a two by two switch to save cross points. This is because the left going middle links of these middle switches are never setup to the right going middle links. For example, in middle switch MS(**1,1**) of Block **1_2** as shown FIG. **1L**, the left going middle links namely ML(**7,1**) and ML(**7,11**) are never switched to the right going middle links ML(**2,1**) and ML(**2,3**). And hence to implement MS(**1,1**) two switches namely: 1) a two by four switch with the middle links ML(**1,1**) and ML(**1,7**) as inputs and the middle links ML(**2,1**), ML(**2,3**), ML(**8,1**), and ML(**8,3**) as outputs and 2) a two by two switch with the middle links ML(**7,1**) and ML(**7,11**) as inputs and the middle links ML(**8,1**) and ML(**8,3**) as outputs are sufficient without loosing any connectivity of the embodiment of MS(**1,1**) being implemented as an eight by eight switch as described before.)

Generalized Butterfly Fat Tree Network Embodiment
with S=1

In one embodiment, in the network **100B** of FIG. **1B** (where it is implemented with s=1), the switches that are placed together are implemented as a combined switch in input stage **110** and output stage **120**; and as a combined switch in all the middle stages, then the network **100B** is the generalized butterfly fat tree network V_{bft}(N$_1$, N$_2$, d, s) where N$_1$=N$_2$=32; d=2; and s=1 with five stages as disclosed in PCT Application Serial No. PCT/US08/64603 that is incorporated by reference above. That is the switches that are placed together in input stage **110** and output stage **120** are implemented as a four by four switch. For example the switch input switch IS1 and output switch OS1 are placed together; so input and output switch IS1&OS1 is implemented as four by four switch with the inlet links IL**1**, IL**2**, ML(**8,1**) and ML(**8,

3**) being the inputs and middle links ML(**1,1**)-ML(**1,2**) and outlet links OL1-OL2 being the outputs

The switches, corresponding to the middle stages that are placed together are implemented as a four by four switch. For example middle switches MS(**1,1**) is implemented as four by four switch with middle links ML(**1,1**), ML(**1,3**), ML(**7,1**) and ML(**7,5**) being the inputs and middle links ML(**2,1**), ML(**2,2**), ML(**8,1**) and ML(**8,2**) being the outputs.

Layout diagrams **100C** in FIG. **1C**, **100D** in FIG. **1D**, **100E** in FIG. **1E**, **100F** in FIG. **1G** are also applicable to generalized butterfly fat tree network V_{bft}(N$_1$, N$_2$, d, s) where N$_1$=N$_2$=32; d=2; and s=1 with five stages. The layout **100C** in FIG. **1C** can be recursively extended for any arbitrarily large generalized butterfly fat tree network V_{bft}(N$_1$, N$_2$, d, s). Accordingly layout **100H** of FIG. **1H** is also applicable to generalized butterfly fat tree network V_{bft}(N$_1$, N$_2$, d, s).

Referring to diagram **100L1** of FIG. **1L1** illustrates a high-level implementation of Block **1_2** (Each of the other blocks have similar implementation) for the layout **100C** of FIG. **1C** when s=1 which represents a generalized butterfly fat tree network V_{bft}(N$_1$, N$_2$, d, s) where N$_1$=N$_2$=32; d=2; and s=1 (All the double links are replaced by single links when s=1). Block **1_2** in **100K1** illustrates both the intra-block and inter-block links. The layout diagram **100L1** corresponds to the embodiment where the switches that are placed together are implemented as a combined switch in the network **100B** of FIG. **1B** when s=1. As noted before then the network **100B** is the generalized butterfly fat tree network V_{bft}(N$_1$, N$_2$, d, s) where N$_1$=N$_2$=32; d=2; and s=1 with nine stages as disclosed in PCT Application Serial No. PCT/US08/64603 that is incorporated by reference above.

That is the switches that are placed together in Block **1_2** as shown in FIG. **1L1** are namely the input and output switch IS1&OS1 belonging to switch **1**, illustrated by dotted lines, (as noted before switch **1** is for illustration purposes only, in practice the switches implemented are input switch IS1 and output switch OS1); middle switch MS(**1,1**) belonging to switch **2**; middle switch MS(**2,1**) belonging to switch **3**; middle switch MS(**3,1**) belonging to switch **4**; And middle switch MS(**4,1**) belonging to switch **5**.

Input and output switch IS1&OS1 are placed together; so input and output switch IS1&OS1 is implemented as four by four switch with the inlet links IL**1**, IL**2**, ML(**8,1**) and ML(**8,3**) being the inputs and middle links ML(**1,1**)-ML(**1,2**) and outlet links OL1-OL2 being the outputs.

Middle switch MS(**1,1**) is implemented as four by four switch with middle links ML(**1,1**), ML(**1,3**), ML(**7,1**) and ML(**7,5**) being the inputs and middle links ML(**2,1**), ML(**2,2**), ML(**8,1**) and ML(**8,2**) being the outputs. Similarly all the other middle switches are also implemented as four by four switches as illustrated in **100L1** of FIG. **1L1**.

In another embodiment, middle switch MS(**1,1**) (or the middle switches in any of the middle stage excepting the root middle stage) of Block **1_2** of $V_{mlink-bft}$(N$_1$, N$_2$, d, s) can be implemented as a two by four switch and a two by two switch to save cross points. This is because the left going middle links of these middle switches are never setup to the right going middle links For example, in middle switch MS(**1,1**) of Block **1_2** as shown FIG. **1L1**, the left going middle links namely ML(**7,1**) and ML(**7,5**) are never switched to the right going middle links ML(**2,1**) and ML(**2,2**). And hence to implement MS(**1,1**) two switches namely: 1) a two by four switch with the middle links ML(**1,1**) and ML(**1,3**) as inputs and the middle links ML(**2,1**), ML(**2,2**), ML(**8,1**), and ML(**8, 2**) as outputs and 2) a two by two switch with the middle links ML(**7,1**) and ML(**7,5**) as inputs and the middle links ML(**8,1**) and ML(**8,2**) as outputs are sufficient without loosing any

US 8,269,523 B2

25   26

connectivity of the embodiment of MS(**1**,**1**) being implemented as an eight by eight switch as described before.)

Hypercube-Like Topology Layout Schemes:

Referring to diagram **300A** in FIG. **3A**, in one embodiment, an exemplary generalized multi-link multi-stage network $V_{mlink}(N_1, N_2, d, s)$ where $N_1=N_2=32$; $d=2$; and $s=2$ with nine stages of one hundred and forty four switches for satisfying communication requests, such as setting up a telephone call or a data call, or a connection between configurable logic blocks, between an input stage **110** and output stage **120** via middle stages **130**, **140**, **150**, **170**, **170**, **180** and **190** is shown where input stage **110** consists of sixteen, two by four switches IS1-IS16 and output stage **120** consists of sixteen, four by two switches OS1-OS16.

As disclosed in PCT Application Serial No. PCT/US08/64604 that is incorporated by reference above, such a network can be operated in rearrangeably non-blocking manner for arbitrary fan-out multicast connections and also can be operated in strictly non-blocking manner for unicast connections.

The diagram **300A** in FIG. **3A** is exactly the same as the diagram **100A** in FIG. **1A** excepting the connection links between middle stage **150** and middle stage **160** as well as between middle stage **160** and middle stage **170**.

Each of the

$$\frac{N}{d}$$

middle switches are connected to exactly d switches in middle stage **160** through two links each for a total of 2×d links (for example the links ML(**4**,**1**) and ML(**4**,**2**) are connected from middle switch MS(**3**,**1**) to middle switch MS(**4**,**1**), and the links ML(**4**,**3**) and ML(**4**,**4**) are connected from middle switch MS(**3**,**1**) to middle switch MS(**4**,**15**)).

Each of the

$$\frac{N}{d}$$

middle switches MS(**4**,**1**)-MS(**4**,**16**) in the middle stage **160** are connected from exactly d input switches through two links each for a total of 2×d links (for example the links ML(**4**,**1**) and ML(**4**,**2**) are connected to the middle switch MS(**4**,**1**) from input switch MS(**3**,**1**), and the links ML(**4**,**59**) and ML(**4**,**60**) are connected to the middle switch MS(**4**,**1**) from input switch MS(**3**,**15**)) and also are connected to exactly d switches in middle stage **170** through two links each for a total of 2×d links (for example the links ML(**5**,**1**) and ML(**5**,**2**) are connected from middle switch MS(**4**,**1**) to middle switch MS(**5**,**1**), and the links ML(**5**,**3**) and ML(**5**,**4**) are connected from middle switch MS(**4**,**1**) to middle switch MS(**5**,**15**)).

Each of the

$$\frac{N}{d}$$

middle switches MS(**5**,**1**)-MS(**5**,**16**) in the middle stage **170** are connected from exactly d input switches through two links each for a total of 2×d links (for example the links ML(**5**,**1**) and ML(**5**,**2**) are connected to the middle switch MS(**5**,**1**) from input switch MS(**4**,**1**), and the links ML(**5**,**59**) and ML(**5**,**60**) are connected to the middle switch MS(**5**,**1**) from input switch MS(**4**,**15**)).

Finally the connection topology of the network **100A** shown in FIG. **1A** is also basically back to back inverse Benes connection topology but with a slight variation. All the cross middle links from middle switches MS(**3**,**1**)-MS(**3**,**8**) connect to middle switches MS(**4**,**9**)-MS(**4**,**16**) and all the cross middle links from middle switches MS(**3**,**9**)-MS(**3**,**16**) connect to middle switches MS(**4**,**1**)-MS(**4**,**8**). Applicant makes a key observation that there are many combinations of connections possible using this property. The difference in the connection topology between diagram **100A** of FIG. **1A** and diagram **300A** of FIG. **3A** is that the connections formed by cross middle links between middle stage **150** and middle stage **160** are made of two different combinations otherwise both the diagrams **100A** and **300A** implement back to back inverse Benes connection topology. Since these networks implement back to back inverse Benes topologies since there is difference in the connections of cross middle links between middle stage **150** and middle stage **160**, the same difference in the connections of cross middle links between **160** and middle stage **170** occurs.

Referring to diagram **300B** in FIG. **3B**, is a folded version of the multi-link multi-stage network **300A** shown in FIG. **3A**. The network **300B** in FIG. **3B** shows input stage **110** and output stage **120** are placed together. That is input switch IS1 and output switch OS1 are placed together, input switch IS2 and output switch OS2 are placed together, and similarly input switch IS16 and output switch OS16 are placed together. All the right going middle links {i.e., inlet links IL1-IL32 and middle links ML(**1**,**1**)-ML(**1**,**64**)} correspond to input switches IS1-IS16, and all the left going middle links {i.e., middle links ML(**7**,**1**)-ML(**7**,**64**) and outlet links OL1-OL32} correspond to output switches OS1-OS16.

Just the same way there is difference in the connection topology between diagram **100A** of FIG. **1A** and diagram **300A** of FIG. **3A** in the way the connections are formed by cross middle links between middle stage **150** and middle stage **160** and also between middle stage **160** and middle stage **170**, the exact similar difference is there between the diagram **100B** of FIG. **1B** and the diagram **300B** of FIG. **3B**, i.e., in the way the connections are formed by cross middle links between middle stage **150** and middle stage **160** and also between middle stage **160** and middle stage **170**.

In one embodiment, in the network **300B** of FIG. **3B**, the switches that are placed together are implemented as separate switches then the network **300B** is the generalized folded multi-link multi-stage network $V_{fold-mlink}(N_1, N_2, d, s)$ where $N_1=N_2=32$; $d=2$; and $s=2$ with nine stages as disclosed in PCT Application Serial No. PCT/US08/64604 that is incorporated by reference above. That is the switches that are placed together in input stage **110** and output stage **120** are implemented as a two by four switch and a four by two switch. For example the switch input switch IS1 and output switch OS1 are placed together; so input switch IS1 is implemented as two by four switch with the inlet links IL1 and IL2 being the inputs of the input switch IS1 and middle links ML(**1**,**1**)-ML(**1**,**4**) being the outputs of the input switch IS1; and output switch OS1 is implemented as four by two switch with the middle links ML(**8**,**1**), ML(**8**,**2**), ML(**8**,**7**) and ML(**8**,**8**) being the inputs of the output switch OS1 and outlet links OL1-OL2 being the outputs of the output switch OS1. Similarly in this embodiment of network **300B** all the switches that are placed together are implemented as separate switches.

Referring to layout **300C** of FIG. **3C**, in one embodiment, there are sixteen blocks namely Block **1**_**2**, Block **3**_**4**, Block **5**_**6**, Block **7**_**8**, Block **9**_**10**, Block **11**_**12**, Block **13**_**14**, Block **15**_**16**, Block **17**_**18**, Block **19**_**20**, Block **21**_**22**, Block **23**_**24**, Block **25**_**26**, Block **27**_**28**, Block **29**_**30**, and

US 8,269,523 B2

27   28

Block **31_32**. Each block implements all the switches in one row of the network **300B** of FIG. **3**B, one of the key aspects of the current invention. For example Block **1_2** implements the input switch IS**1**, output Switch OS**1**, middle switch MS(**1,1**), middle switch MS(**7,1**), middle switch MS(**2,1**), middle switch MS(**6,1**), middle switch MS(**3,1**), middle switch MS(**5,1**), and middle switch MS(**4,1**). For the simplification of illustration, Input switch IS**1** and output switch OS**1** together are denoted as switch **1**; Middle switch MS(**1,1**) and middle switch MS(**7,1**) together are denoted by switch **2**; Middle switch MS(**2,1**) and middle switch MS(**6,1**) together are denoted by switch **3**; Middle switch MS(**3,1**) and middle switch MS(**5,1**) together are denoted by switch **4**; And middle switch MS(**4,1**) is denoted by switch **5**.

All the straight middle links are illustrated in layout **300C** of FIG. **3**C. For example in Block **1_2**, inlet links IL**1**-IL**2**, outlet links OL**1**-OL**2**, middle link ML(**1,1**), middle link ML(**1,2**), middle link ML(**8,1**), middle link ML(**8,2**), middle link ML(**2,1**), middle link ML(**2,2**), middle link ML(**7,1**), middle link ML(**7,2**), middle link ML(**3,1**), middle link ML(**3,2**), middle link ML(**6,1**), middle link ML(**6,2**), middle link ML(**4,1**), middle link ML(**4,2**), middle link ML(**5,1**) and middle link ML(**5,2**) are illustrated in layout **300C** of FIG. **3**C.

Even though it is not illustrated in layout **300C** of FIG. **3**C, in each block, in addition to the switches there may be Configurable Logic Blocks (CLB) or any arbitrary digital circuit or sub-integrated circuit block depending on the applications in different embodiments. There are four quadrants in the layout **300C** of FIG. **3**C namely top-left, bottom-left, top-right and bottom-right quadrants. Top-left quadrant implements Block **1_2**, Block **3_4**, Block **5_6**, and Block **7_8**. Bottom-left quadrant implements Block **9_10**, Block **11_12**, Block **13_14**, and Block **15_16**. Top-right quadrant implements Block **25_26**, Block **27_28**, Block **29_30**, and Block **31_32**. Bottom-right quadrant implements Block **17_18**, Block **19_20**, Block **21_22**, and Block **23_24**. There are two halves in layout **300C** of FIG. **3**C namely left-half and right-half. Left-half consists of top-left and bottom-left quadrants. Right-half consists of top-right and bottom-right quadrants.

Recursively in each quadrant there are four sub-quadrants. For example in top-left quadrant there are four sub-quadrants namely top-left sub-quadrant, bottom-left sub-quadrant, top-right sub-quadrant and bottom-right sub-quadrant. Top-left sub-quadrant of top-left quadrant implements Block **1_2**. Bottom-left sub-quadrant of top-left quadrant implements Block **3_4**. Top-right sub-quadrant of top-left quadrant implements Block **7_8**. Finally bottom-right sub-quadrant of top-left quadrant implements Block **5_6**. Similarly there are two sub-halves in each quadrant. For example in top-left quadrant there are two sub-halves namely left-sub-half and right-sub-half. Left-sub-half of top-left quadrant implements Block **1_2** and Block **3_4**. Right-sub-half of top-left quadrant implements Block **7_8** and Block **5_6**. Recursively in larger multi-stage network $V_{fold-mlink}(N_1, N_2, d, s)$ where $N_1=N_2>32$, the layout in this embodiment in accordance with the current invention, will be such that the super-quadrants will also be arranged in a similar manner.

Layout **300D** of FIG. **3**D illustrates the inter-block links (in the layout **300C** of FIG. **3**C all the cross middle links are inter-block links) between switches **1** and **2** of each block. For example middle links ML(**1,3**), ML(**1,4**), ML(**8,7**), and ML(**8,8**) are connected between switch **1** of Block **1_2** and switch **2** of Block **3_4**. Similarly middle links ML(**1,7**), ML(**1,8**), ML(**8,3**), and ML(**8,4**) are connected between switch **2** of Block **1_2** and switch **1** of Block **3_4**. Applicant notes that the inter-block links illustrated in layout **100D** of

FIG. **1**D can be implemented as vertical tracks in one embodiment. Also in one embodiment inter-block links are implemented as two different tracks (for example middle links ML(**1,4**) and ML(**8,8**) are implemented as two different tracks); or in an alternative embodiment inter-block links are implemented as a time division multiplexed single track (for example middle links ML(**1,4**) and ML(**8,8**) are implemented as a time division multiplexed single track).

Layout **300E** of FIG. **3**E illustrates the inter-block links between switches **2** and **3** of each block. For example middle links ML(**2,3**), ML(**2,4**), ML(**7,11**), and ML(**7,12**) are connected between switch **2** of Block **1_2** and switch **3** of Block **3_4**. Similarly middle links ML(**2,11**), ML(**2,12**), ML(**7,3**), and ML(**7,4**) are connected between switch **3** of Block **1_2** and switch **2** of Block **3_4**. Applicant notes that the inter-block links illustrated in layout **300E** of FIG. **3**E can be implemented as diagonal tracks in one embodiment. Also in one embodiment inter-block links are implemented as two different tracks (for example middle links ML(**2,12**) and ML(**7,4**) are implemented as two different tracks); or in an alternative embodiment inter-block links are implemented as a time division multiplexed single track (for example middle links ML(**2,12**) and ML(**7,4**) are implemented as a time division multiplexed single track).

Layout **300F** of FIG. **3**F illustrates the inter-block links between switches **3** and **4** of each block. For example middle links ML(**3,3**), ML(**3,4**), ML(**6,19**), and ML(**6,20**) are connected between switch **3** of Block **1_2** and switch **4** of Block **3_4**. Similarly middle links ML(**3,19**), ML(**3,20**), ML(**6,3**), and ML(**6,4**) are connected between switch **4** of Block **1_2** and switch **3** of Block **3_4**. Applicant notes that the inter-block links illustrated in layout **300F** of FIG. **3**F can be implemented as vertical tracks in one embodiment. Also in one embodiment inter-block links are implemented as two different tracks (for example middle links ML(**3,4**) and ML(**6,20**) are implemented as two different tracks); or in an alternative embodiment inter-block links are implemented as a time division multiplexed single track (for example middle links ML(**3,4**) and ML(**6,20**) are implemented as a time division multiplexed single track).

Layout **300G** of FIG. **3**G illustrates the inter-block links between switches **4** and **5** of each block. For example middle links ML(**4,3**), ML(**4,4**), ML(**5,35**), and ML(**5,36**) are connected between switch **4** of Block **1_2** and switch **5** of Block **3_4**. Similarly middle links ML(**4,35**), ML(**4,36**), ML(**5,3**), and ML(**5,4**) are connected between switch **5** of Block **1_2** and switch **4** of Block **3_4**. Applicant notes that the inter-block links illustrated in layout **300G** of FIG. **3**G can be implemented as horizontal tracks in one embodiment. Also in one embodiment inter-block links are implemented as two different tracks (for example middle links ML(**4,4**) and ML(**5,36**) are implemented as two different tracks); or in an alternative embodiment inter-block links are implemented as a time division multiplexed single track (for example middle links ML(**4,4**) and ML(**5,36**) are implemented as a time division multiplexed single track).

The complete layout for the network **300B** of FIG. **3**B is given by combining the links in layout diagrams of **300C**, **300D**, **300E**, **300F**, and **300G**. Applicant notes that in the layout **300C** of FIG. **3**C, the inter-block links between switch **1** and switch **2** are vertical tracks as shown in layout **300D** of FIG. **3**D; the inter-block links between switch **2** and switch **3** are horizontal tracks as shown in layout **300E** of FIG. **3**E; the inter-block links between switch **3** and switch **4** are vertical tracks as shown in layout **300F** of FIG. **3**F; and finally the inter-block links between switch **4** and switch **5** are horizontal tracks as shown in layout **300G** of FIG. **3**G. The pattern is

US 8,269,523 B2

29   30

either vertical tracks, horizontal tracks or diagonal tracks. It continues recursively for larger networks of N>32 as will be illustrated later.

Some of the key aspects of the current invention related to layout diagram **300C** of FIG. **3C** are noted. 1) All the switches in one row of the multi-stage network **300B** are implemented in a single block. 2) The blocks are placed in such a way that all the inter-block links are either horizontal tracks, vertical tracks or diagonal tracks; 3) The length of the longest wire is about half of the width (or length) of the complete layout (For example middle link ML(**4,4**) is about half the width of the complete layout.);

The layout **300C** in FIG. **3C** can be recursively extended for any arbitrarily large generalized folded multi link multi-stage network $V_{fold-mlink}(N_1, N_2, d, s)$. Referring to layout **300H** of FIG. **3H**, illustrates the extension of layout **300C** for the network $V_{fold-mlink}(N_1, N_2, d, s)$ where $N_1=N_2=128$; d=2; and s=2. There are four super-quadrants in layout **300H** namely top-left super-quadrant, bottom-left super-quadrant, top-right super-quadrant, bottom-right super-quadrant. Total number of blocks in the layout **300H** is sixty four. Top-left super-quadrant implements the blocks from block **1_2** to block **31_32**. Each block in all the super-quadrants has two more switches namely switch **6** and switch **7** in addition to the switches [**1-5**] illustrated in layout **300C** of FIG. **3C**. The inter-block link connection topology is the exactly the same between the switches **1** and **2**; switches **2** and **3**; switches **3** and **4**; switches **4** and **5** as it is shown in the layouts of FIG. **3D**, FIG. **3E**, FIG. **3F**, and FIG. **3G** respectively.

Bottom-left super-quadrant implements the blocks from block **33_34** to block **63_64**. Top-right super-quadrant implements the blocks from block **65_66** to block **95_96**. And bottom-right super-quadrant implements the blocks from block **97_98** to block **127_128**. In all these three super-quadrants also, the inter-block link connection topology is the exactly the same between the switches **1** and **2**; switches **2** and **3**; switches **3** and **4**; switches **4** and **5** as that of the top-left super-quadrant.

Recursively in accordance with the current invention, the inter-block links connecting the switch **5** and switch **6** will be vertical tracks between the corresponding switches of top-left super-quadrant and bottom-left super-quadrant. And similarly the inter-block links connecting the switch **5** and switch **6** will be vertical tracks between the corresponding switches of top-right super-quadrant and bottom-right super-quadrant. The inter-block links connecting the switch **6** and switch **7** will be horizontal tracks between the corresponding switches of top-left super-quadrant and top-right super-quadrant. And similarly the inter-block links connecting the switch **6** and switch **7** will be horizontal tracks between the corresponding switches of bottom-left super-quadrant and bottom-right super-quadrant.

Ring Topology Layout Schemes:

Layout diagram **400C** of FIG. **4C** is another embodiment for the generalized folded multi-link multi-stage network $V_{fold-mlink}(N_1, N_2, d, s)$ diagram **100B** in FIG. **1B**.

Referring to layout **400C** of FIG. **4C**, there are sixteen blocks namely Block **1_2**, Block **3_4**, Block **5_6**, Block **7_8**, Block **9_10**, Block **11_12**, Block **13_14**, Block **15_16**, Block **17_18**, Block **19_20**, Block **21_22**, Block **23_24**, Block **25_26**, Block **27_28**, Block **29_30**, and Block **31_32**. Each block implements all the switches in one row of the network **100B** of FIG. **1B**, one of the key aspects of the current invention. For example Block **1_2** implements the input switch IS**1**, output Switch OS**1**, middle switch MS(**1,1**), middle switch MS(**7,1**), middle switch MS(**2,1**), middle switch MS(**6,1**), middle switch MS(**3,1**), middle switch MS(**5,1**), and middle

switch MS(**4,1**). For the simplification of illustration, Input switch IS**1** and output switch OS**1** together are denoted as switch **1**; Middle switch MS(**1,1**) and middle switch MS(**7,1**) together are denoted by switch **2**; Middle switch MS(**2,1**) and middle switch MS(**6,1**) together are denoted by switch **3**; Middle switch MS(**3,1**) and middle switch MS(**5,1**) together are denoted by switch **4**; And middle switch MS(**4,1**) is denoted by switch **5**.

All the straight middle links are illustrated in layout **400C** of FIG. **4C**. For example in Block **1_2**, inlet links IL1-IL**2**, outlet links OL1-OL**2**, middle link ML(**1,1**), middle link ML(**1,2**), middle link ML(**8,1**), middle link ML(**8,2**), middle link ML(**2,1**), middle link ML(**2,2**), middle link ML(**7,1**), middle link ML(**7,2**), middle link ML(**3,1**), middle link ML(**3,2**), middle link ML(**6,1**), middle link ML(**6,2**), middle link ML(**4,1**), middle link ML(**4,2**), middle link ML(**5,1**) and middle link ML(**5,2**) are illustrated in layout **400C** of FIG. **4C**.

Even though it is not illustrated in layout **400C** of FIG. **4C**, in each block, in addition to the switches there may be Configurable Logic Blocks (CLB) or any arbitrary digital circuit or sub-integrated circuit block depending on the applications in different embodiments. The topology of the layout **400C** in FIG. **4C** is a ring. For each of the neighboring rows in diagram **100B** of FIG. **1B** the corresponding blocks are also physically neighbors in layout diagram **400C** of FIG. **4C**. In addition the topmost row is also logically considered as neighbor to the bottommost row. For example Block **1_2** (implementing the switches belonging to a row in diagram **100B** of FIG. **1B**) has Block **3_4** as neighbor since Block **3_4** implements the switches in its neighboring row. Similarly Block **1_2** also has Block **31_32** as neighbor since Block **1_2** implements topmost row of switches and Block **31_32** implements bottommost row of switches in diagram **100B** of FIG. **1B**. The ring layout scheme illustrated in **400C** of FIG. **4C** can be generalized for a large multi-stage network $V_{fold-mlink}(N_1, N_2, d, s)$ where $N_1=N_2>32$, in accordance with the current invention.

Layout **400B** of FIG. **4B** illustrates the inter-block links (in the layout **400A** of FIG. **4A** all the cross middle links are inter-block links) between switches **1** and **2** of each block. For example middle links ML(**1,3**), ML(**1,4**), ML(**8,7**), and ML(**8,8**) are connected between switch **1** of Block **1_2** and switch **2** of Block **3_4**. Similarly middle links ML(**1,7**), ML(**1,8**), ML(**8,3**), and ML(**8,4**) are connected between switch **2** of Block **1_2** and switch **1** of Block **3_4**. Applicant notes that the inter-block links illustrated in layout **400B** of FIG. **4B** are implemented as vertical tracks or horizontal tracks or diagonal tracks. Also in one embodiment inter-block links are implemented as two different tracks (for example middle links ML(**1,4**) and ML(**8,8**) are implemented as two different tracks); or in an alternative embodiment inter-block links are implemented as a time division multiplexed single track (for example middle links ML(**1,4**) and ML(**8,8**) are implemented as a time division multiplexed single track).

Layout **400C** of FIG. **4C** illustrates the inter-block links between switches **2** and **3** of each block. For example middle links ML(**2,3**), ML(**2,4**), ML(**7,11**), and ML(**7,12**) are connected between switch **2** of Block **1_2** and switch **3** of Block **3_4**. Similarly middle links ML(**2,11**), ML(**2,12**), ML(**7,3**), and ML(**7,4**) are connected between switch **3** of Block **1_2** and switch **2** of Block **3_4**. Applicant notes that the inter-block links illustrated in layout **400C** of FIG. **4C** are implemented as vertical tracks or horizontal tracks or diagonal tracks. Also in one embodiment inter-block links are implemented as two different tracks (for example middle links ML(**2,12**) and ML(**7,4**) are implemented as two different tracks); or in an alternative embodiment inter-block links are

US 8,269,523 B2

31   32

implemented as a time division multiplexed single track (for example middle links ML(2,12) and ML(7,4) are implemented as a time division multiplexed single track).

Layout 400D of FIG. 4D illustrates the inter-block links between switches 3 and 4 of each block. For example middle links ML(3,3), ML(3,4), ML(6,19), and ML(6,20) are connected between switch 3 of Block 1_2 and switch 4 of Block 3_4. Similarly middle links ML(3,19), ML(3,20), ML(6,3), and ML(6,4) are connected between switch 4 of Block 1_2 and switch 3 of Block 3_4. Applicant notes that the inter-block links illustrated in layout 400D of FIG. 4D are implemented as vertical tracks or horizontal tracks or diagonal tracks. Also in one embodiment inter-block links are implemented as two different tracks (for example middle links ML(3,4) and ML(6,20) are implemented as two different tracks); or in an alternative embodiment inter-block links are implemented as a time division multiplexed single track (for example middle links ML(3,4) and ML(6,20) are implemented as a time division multiplexed single track).

Layout 400E of FIG. 4E illustrates the inter-block links between switches 4 and 5 of each block. For example middle links ML(4,3), ML(4,4), ML(5,35), and ML(5,36) are connected between switch 4 of Block 1_2 and switch 5 of Block 3_4. Similarly middle links ML(4,35), ML(4,36), ML(5,3), and ML(5,4) are connected between switch 5 of Block 1_2 and switch 4 of Block 3_4. Applicant notes that the inter-block links illustrated in layout 400E of FIG. 4E are implemented as vertical tracks or horizontal tracks or diagonal tracks. Also in one embodiment inter-block links are implemented as two different tracks (for example middle links ML(4,4) and ML(5,36) are implemented as two different tracks); or in an alternative embodiment inter-block links are implemented as a time division multiplexed single track (for example middle links ML(4,4) and ML(5,36) are implemented as a time division multiplexed single track).

The complete layout for the network 100B of FIG. 1B is given by combining the links in layout diagrams of 400A, 400B, 400C, 400D, and 400E.

Some of the key aspects of the current invention related to layout diagram 400A of FIG. 4A are noted. 1) All the switches in one row of the multi-stage network 100B are implemented in a single block. 2) The blocks are placed in such a way that all the inter-block links are either horizontal tracks, vertical tracks or diagonal tracks; 3) Length of the different wires between the same two middle stages is not the same. However it gives an opportunity to implement the most connected circuits to place and route through the blocks which have shorter wires.

Layout diagram 400C1 of FIG. 4C1 is another embodiment for the generalized folded multi-link multi-stage network $V_{fold-mlink}(N_1, N_2, d, s)$ diagram 100B in FIG. 1B. Referring to layout 400C1 of FIG. 4C1, there are sixteen blocks namely Block 1_2, Block 3_4, Block 5_6, Block 7_8, Block 9_10, Block 11_12, Block 13_14, Block 15_16, Block 17_18, Block 19_20, Block 21_22, Block 23_24, Block 25_26, Block 27_28, Block 29_30, and Block 31_32. Each block implements all the switches in one row of the network 100B of FIG. 1B, one of the key aspects of the current invention. For example Block 1_2 implements the input switch IS1, output Switch OS1, middle switch MS(1,1), middle switch MS(7,1), middle switch MS(2,1), middle switch MS(6,1), middle switch MS(3,1), middle switch MS(5,1), and middle switch MS(4,1). For the simplification of illustration, Input switch IS1 and output switch OS1 together are denoted as switch 1; Middle switch MS(1,1) and middle switch MS(7,1) together are denoted by switch 2; Middle switch MS(2,1) and middle switch MS(6,1) together are denoted by switch 3; Middle

switch MS(3,1) and middle switch MS(5,1) together are denoted by switch 4; And middle switch MS(4,1) is denoted by switch 5.

All the straight middle links are illustrated in layout 400C1 of FIG. 4C1. For example in Block 1_2, inlet links IL1-IL2, outlet links OL1-OL2, middle link ML(1,1), middle link ML(1,2), middle link ML(8,1), middle link ML(8,2), middle link ML(2,1), middle link ML(2,2), middle link ML(7,1), middle link ML(7,2), middle link ML(3,1), middle link ML(3,2), middle link ML(6,1), middle link ML(6,2), middle link ML(4,1), middle link ML(4,2), middle link ML(5,1) and middle link ML(5,2) are illustrated in layout 400C1 of FIG. 4C1.

Even though it is not illustrated in layout 400C1 of FIG. 4C1, in each block, in addition to the switches there may be Configurable Logic Blocks (CLB) or any arbitrary digital circuit or sub-integrated circuit block depending on the applications in different embodiments. The topology of the layout 400C1 in FIG. 4C1 is another embodiment of ring layout topology. For each of the neighboring rows in diagram 100B of FIG. 1B the corresponding blocks are also physically neighbors in layout diagram 400C of FIG. 4C. In addition the topmost row is also logically considered as neighbor to the bottommost row. For example Block 1_2 (implementing the switches belonging to a row in diagram 100B of FIG. 1B) has Block 3_4 as neighbor since Block 3_4 implements the switches in its neighboring row. Similarly Block 1_2 also has Block 31_32 as neighbor since Block 1_2 implements topmost row of switches and Block 31_32 implements bottommost row of switches in diagram 100B of FIG. 1B. The ring layout scheme illustrated in 400C of FIG. 4C can be generalized for a large multi-stage network $V_{fold-mlink}(N_1, N_2, d, s)$ where $N_1=N_2>32$, in accordance with the current invention.

All the layout embodiments disclosed in the current invention are applicable to generalized multi-stage networks $V(N_1, N_2, d, s)$, generalized folded multi-stage networks $V_{fold}(N_1, N_2, d, s)$, generalized butterfly fat tree networks $V_{bft}(N_1, N_2, d, s)$, generalized multi-link multi-stage networks $V_{mlink}(N_1, N_2, d, s)$, generalized folded multi-link multi-stage networks $V_{fold-mlink}(N_1, N_2, d, s)$, generalized multi-link butterfly fat tree networks $V_{mlink-bft}(N_1, N_2, d, s)$, and generalized hypercube networks $V_{hcube}(N_1, N_2, d, s)$ for s=1, 2, 3 or any number in general, and for both $N_1=N=N$ and $N_1 \neq N_2$, and d is any integer.

Conversely applicant makes another important observation that generalized hypercube networks $V_{hcube}(N_1, N_2, d, s)$ are implemented with the layout topology being the hypercube topology shown in layout 100C of FIG. 1C with large scale cross point reduction as any one of the networks described in the current invention namely: generalized multi-stage networks $V(N_1, N_2, d, s)$, generalized folded multi-stage networks $V_{fold}(N_1, N_2, d, s)$, generalized butterfly fat tree networks $V_{bft}(N_1, N_2, d, s)$, generalized multi-link multi-stage networks $V_{mlink}(N_1, N_2, d, s)$, generalized folded multi-link multi-stage networks $V_{fold-mlink}(N_1, N_2, d, s)$, generalized multi-link butterfly fat tree networks $V_{mlink-bft}(N_1, N_2, d, s)$ for s=1, 2, 3 or any number in general, and for both $N_1=N_2=N$ and $N_1 \neq N_2$, and d is any integer.

Applications Embodiments

All the embodiments disclosed in the current invention are useful in many varieties of applications. FIG. 5A1 illustrates the diagram of 500A1 which is a typical two by two switch with two inlet links namely IL1 and IL2, and two outlet links namely OL1 and OL2. The two by two switch also implements four crosspoints namely CP(1,1), CP(1,2), CP(2,1) and

US 8,269,523 B2

33   34

CP(2,2) as illustrated in FIG. 5A1. For example the diagram of 500A1 may the implementation of middle switch MS(1,1) of the diagram 100K of FIG. 1K where inlet link IL1 of diagram 500A1 corresponds to middle link ML(1,1) of diagram 100K, inlet link IL2 of diagram 500A1 corresponds to middle link ML(1,7) of diagram 100K, outlet link OL1 of diagram 500A1 corresponds to middle link ML(2,1) of diagram 500A1, outlet link OL2 of diagram 500A1 corresponds to middle link ML(2,3) of diagram 100K.

1) Programmable Integrated Circuit Embodiments

All the embodiments disclosed in the current invention are useful in programmable integrated circuit applications. FIG. 5A2 illustrates the detailed diagram 500A2 for the implementation of the diagram 500A1 in programmable integrated circuit embodiments. Each crosspoint is implemented by a transistor coupled between the corresponding inlet link and outlet link, and a programmable cell in programmable integrated circuit embodiments. Specifically crosspoint CP(1,1) is implemented by transistor C(1,1) coupled between inlet link IL1 and outlet link OL1, and programmable cell P(1,1); crosspoint CP(1,2) is implemented by transistor C(1,2) coupled between inlet link IL1 and outlet link OL2, and programmable cell P(1,2); crosspoint CP(2,1) is implemented by transistor C(2,1) coupled between inlet link IL2 and outlet link OL1, and programmable cell P(2,1); and crosspoint CP(2,2) is implemented by transistor C(2,2) coupled between inlet link IL2 and outlet link OL2, and programmable cell P(2,2).

If the programmable cell is programmed ON, the corresponding transistor couples the corresponding inlet link and outlet link. If the programmable cell is programmed OFF, the corresponding inlet link and outlet link are not connected. For example if the programmable cell P(1,1) is programmed ON, the corresponding transistor C(1,1) couples the corresponding inlet link IL1 and outlet link OL1. If the programmable cell P(1,1) is programmed OFF, the corresponding inlet link IL1 and outlet link OL1 are not connected. In volatile programmable integrated circuit embodiments the programmable cell may be an SRAM (Static Random Address Memory) cell. In non-volatile programmable integrated circuit embodiments the programmable cell may be a Flash memory cell. Also the programmable integrated circuit embodiments may implement field programmable logic arrays (FPGA) devices, or programmable Logic devices (PLD), or Application Specific Integrated Circuits (ASIC) embedded with programmable logic circuits or 3D-FPGAs.

FIG. 5A2 also illustrates a buffer B1 on inlet link IL2. The signals driven along inlet link IL2 are amplified by buffer B1. Buffer B1 can be inverting or non-inverting buffer. Buffers such as B1 are used to amplify the signal in links which are usually long.

2) One-Time Programmable Integrated Circuit Embodiments

All the embodiments disclosed in the current invention are useful in one-time programmable integrated circuit applications. FIG. 5A3 illustrates the detailed diagram 500A3 for the implementation of the diagram 500A1 in one-time programmable integrated circuit embodiments. Each crosspoint is implemented by a via coupled between the corresponding inlet link and outlet link in one-time programmable integrated circuit embodiments. Specifically crosspoint CP(1,1) is implemented by via V(1,1) coupled between inlet link IL1 and outlet link OL1; crosspoint CP(1,2) is implemented by

via V(1,2) coupled between inlet link IL1 and outlet link OL2; crosspoint CP(2,1) is implemented by via V(2,1) coupled between inlet link IL2 and outlet link OL1; and crosspoint CP(2,2) is implemented by via V(2,2) coupled between inlet link IL2 and outlet link OL2.

If the via is programmed ON, the corresponding inlet link and outlet link are permanently connected which is denoted by thick circle at the intersection of inlet link and outlet link. If the via is programmed OFF, the corresponding inlet link and outlet link are not connected which is denoted by the absence of thick circle at the intersection of inlet link and outlet link For example in the diagram 500A3 the via V(1,1) is programmed ON, and the corresponding inlet link IL1 and outlet link OL1 are connected as denoted by thick circle at the intersection of inlet link IL1 and outlet link OL1; the via V(2,2) is programmed ON, and the corresponding inlet link IL2 and outlet link OL2 are connected as denoted by thick circle at the intersection of inlet link IL2 and outlet link OL2; the via V(1,2) is programmed OFF, and the corresponding inlet link IL1 and outlet link OL2 are not connected as denoted by the absence of thick circle at the intersection of inlet link IL1 and outlet link OL2; the via V(2,1) is programmed OFF, and the corresponding inlet link IL2 and outlet link OL1 are not connected as denoted by the absence of thick circle at the intersection of inlet link IL2 and outlet link OL1. One-time programmable integrated circuit embodiments may be anti-fuse based programmable integrated circuit devices or mask programmable structured ASIC devices.

3) Integrated Circuit Placement and Route Embodiments

All the embodiments disclosed in the current invention are useful in Integrated Circuit Placement and Route applications, for example in ASIC backend Placement and Route tools. FIG. 5A4 illustrates the detailed diagram 500A4 for the implementation of the diagram 500A1 in Integrated Circuit Placement and Route embodiments. In an integrated circuit since the connections are known a-priori, the switch and crosspoints are actually virtual. However the concept of virtual switch and virtual crosspoint using the embodiments disclosed in the current invention reduces the number of required wires, wire length needed to connect the inputs and outputs of different netlists and the time required by the tool for placement and route of netlists in the integrated circuit.

Each virtual crosspoint is used to either to hardwire or provide no connectivity between the corresponding inlet link and outlet link. Specifically crosspoint CP(1,1) is implemented by direct connect point DCP(1,1) to hardwire (i.e., to permanently connect) inlet link IL1 and outlet link OL1 which is denoted by the thick circle at the intersection of inlet link IL1 and outlet link OL1; crosspoint CP(2,2) is implemented by direct connect point DCP(2,2) to hardwire inlet link IL2 and outlet link OL2 which is denoted by the thick circle at the intersection of inlet link IL2 and outlet link OL2. The diagram 500A4 does not show direct connect point DCP (1,2) and direct connect point DCP(1,3) since they are not needed and in the hardware implementation they are eliminated. Alternatively inlet link IL1 needs to be connected to outlet link OL1 and inlet link IL1 does not need to be connected to outlet link OL2. Also inlet link IL2 needs to be connected to outlet link OL2 and inlet link IL2 does not need to be connected to outlet link OL1. Furthermore in the example of the diagram 500A4, there is no need to drive the signal of inlet link IL1 horizontally beyond outlet link OL1 and hence the inlet link IL1 is not even extended horizontally

US 8,269,523 B2

35   36

until the outlet link OL2. Also the absence of direct connect point DCP(2,1) illustrates there is no need to connect inlet link IL2 and outlet link OL1.

In summary in integrated circuit placement and route tools, the concept of virtual switches and virtual cross points is used during the implementation of the placement & routing algorithmically in software, however during the hardware implementation cross points in the cross state are implemented as hardwired connections between the corresponding inlet link and outlet link, and in the bar state are implemented as no connection between inlet link and outlet link.

3) More Application Embodiments

All the embodiments disclosed in the current invention are also useful in the design of SoC interconnects, Field programmable interconnect chips, parallel computer systems and in time-space-time switches.

Numerous modifications and adaptations of the embodiments, implementations, and examples described herein will be apparent to the skilled artisan in view of the disclosure.

What is claimed is:

1. An integrated circuit device comprising a plurality of sub-integrated circuit blocks and a routing network, and

Said each plurality of sub-integrated circuit blocks comprising a plurality of inlet links and a plurality of outlet links; and

Said routing network comprising of a plurality of stages y, in each said sub-integrated circuit block, starting from the lowest stage of 1 to the highest stage of y, where $y \geqq 1$; and

Said routing network comprising a plurality of switches of size $d \times d$, where $d \geqq 2$, in each said stage and each said switch of size $d \times d$ having d inlet links and d outlet links; and

Said plurality of outlet links of said each sub-integrated circuit block are directly connected to said inlet links of said switches of its corresponding said lowest stage of 1, and said plurality of inlet links of said each sub-integrated circuit block are directly connected from said outlet links of said switches of its corresponding said lowest stage of 1; and

Said each sub-integrated circuit block comprising a plurality of forward connecting links connecting from switches in a lower stage to switches in its immediate succeeding higher stage, and also comprising a plurality of backward connecting links connecting from switches in a higher stage to switches in its immediate preceding lower stage; and

Said each sub-integrated circuit block comprising a plurality straight links in said forward connecting links from switches in said each lower stage to switches in its immediate succeeding higher stage and a plurality cross links in said forward connecting links from switches in said each lower stage to switches in its immediate succeeding higher stage, and further comprising a plurality of straight links in said backward connecting links from switches in said each higher stage to switches in its immediate preceding lower stage and a plurality of cross links in said backward connecting links from switches in said each higher stage to switches in its immediate preceding lower stage,

said plurality of sub-integrated circuit blocks arranged in a two-dimensional grid of rows and columns, and

said all straight links are connecting from switches in each said sub-integrated circuit block are connecting to switches in the same said sub-integrated circuit block;

and said all cross links are connecting as either vertical or horizontal links between switches in two different said sub-integrated circuit blocks which are either placed vertically above or below, or placed horizontally to the left or to the right,

each said plurality of sub-integrated circuit blocks comprising same number of said stages and said switches in each said stage, regardless of the size of said two-dimensional grid so that each said plurality of sub-integrated circuit block with its corresponding said stages and said switches in each stage is replicable in both vertical direction or horizontal direction of said two-dimensional grid.

2. The integrated circuit device of claim 1,

said two-dimensional grid of said sub-integrated circuit blocks with their corresponding said stages and said switches in each stage is scalable by any power of 2, and,

for each multiplication of 2 of the size of total said sub-integrated circuit blocks, by adding one more stage of switches and the layout is placed in hypercube format and also the cross links between said one more stage of switches are connected in hypercube format.

3. The integrated circuit device of claim 2, wherein said cross links in succeeding stages are connecting as alternative vertical and horizontal links between switches in said sub-integrated circuit blocks.

4. The integrated circuit device of claim 3, wherein said cross links from switches in a stage in one of said sub-integrated circuit blocks are connecting to switches in the succeeding stage in another of said sub-integrated circuit blocks so that said cross links are either vertical links or horizontal and vice versa, and hereinafter such cross links are "shuffle exchange links").

5. The integrated circuit device of claim 4, wherein said all horizontal shuffle exchange links between switches in any two corresponding said succeeding stages are substantially of equal length and said vertical shuffle exchange links between switches in any two corresponding said succeeding stages are substantially of equal length in the entire said integrated circuit device.

6. The integrated circuit device of claim 5, wherein the shortest horizontal shuffle exchange links are connecting at the lowest stage and between switches in two nearest neighboring said sub-integrated circuit blocks, and length of the horizontal shuffle exchange links is doubled in each succeeding stage; and the shortest vertical shuffle exchange links are connecting at the lowest stage and between switches in two nearest neighboring said sub-integrated circuit blocks, and length of the vertical shuffle exchange links is doubled in each succeeding stage.

7. The integrated circuit device of claim 6, wherein $y \geqq (\log_2 N)$, where N>1, so that the length of the horizontal shuffle exchange links in the highest stage is equal to half the size of the horizontal size of said two dimensional grid of sub-integrated circuit blocks and the length of the vertical shuffle exchange links in the highest stage is equal to half the size of the vertical size of said two dimensional grid of sub-integrated circuit blocks.

8. The integrated circuit device of claim 7, wherein d=2 and there is only one switch in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there is only one switch in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is rearrangeably nonblocking for unicast Benes network with full bandwidth.

US 8,269,523 B2

37   38

9. The integrated circuit device of claim **7**, wherein d=2 and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is strictly nonblocking for unicast Benes network and rearrangeably nonblocking for arbitrary fan-out multicast Benes network with full bandwidth.

10. The integrated circuit device of claim **7**, wherein d=2 and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is strictly nonblocking for arbitrary fan-out multicast Benes network with full bandwidth.

11. The integrated circuit device of claim **6**, wherein $y \geq (\log_2 N)$, where N>1, so that the length of the horizontal shuffle exchange links in the highest stage is equal to half the size of the horizontal size of said two dimensional grid of sub-integrated circuit blocks and the length of the vertical shuffle exchange links in the highest stage is equal to half the size of the vertical size of said two dimensional grid of sub-integrated circuit blocks, and

said each sub-integrated circuit block further comprising a plurality of U-turn links within switches in each of said stages in each of said sub-integrated circuit blocks.

12. The integrated circuit device of claim **11**, wherein d=2 and there is only one switch in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there is only one switch in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is rearrangeably nonblocking for unicast butterfly fat tree network with full bandwidth.

13. The integrated circuit device of claim **11**, wherein d=2 and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is strictly nonblocking for unicast butterfly fat tree network and rearrangeably nonblocking for arbitrary fan-out multicast butterfly fat tree network with full bandwidth.

14. The integrated circuit device of claim **11**, wherein d=2 and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is strictly nonblocking for arbitrary fan-out multicast butterfly fat tree network with full bandwidth.

15. The integrated circuit device of claim **1**, wherein said horizontal and vertical links are implemented on two or more metal layers.

16. The integrated circuit device of claim **1**, wherein switches comprising active and reprogrammable cross points and said each cross point is programmable by an SRAM cell or a Flash Cell.

17. The integrated circuit device of claim **1**, wherein said sub-integrated circuit blocks are of equal die size.

18. The integrated circuit device of claim **15**, wherein said sub-integrated circuit blocks are Lookup Tables (hereinafter "LUTs") and said integrated circuit device is a field program-

mable gate array (FPGA) device or field programmable gate array (FPGA) block embedded in another integrated circuit device.

19. The integrated circuit device of claim **15**, wherein said sub-integrated circuit blocks are AND or OR gates and said integrated circuit device is a programmable logic device (PLD).

20. The integrated circuit device of claim **1**, wherein said sub-integrated circuit blocks comprising any arbitrary hardware logic or memory circuits.

21. The integrated circuit device of claim **1**, wherein said switches comprising active one-time programmable cross points and said integrated circuit device is a mask programmable gate array (MPGA) device or a structured ASIC device.

22. The integrated circuit device of claim **1**, wherein said switches comprising passive cross points or just connection of two links or not and said integrated circuit device is a Application Specific Integrated Circuit (ASIC) device.

23. The integrated circuit device of claim **1**, wherein said sub-integrated circuit blocks further recursively comprise one or more super-sub-integrated circuit blocks and a sub-routing network.

24. The integrated circuit device of claim **4**, wherein said all horizontal shuffle exchange links between switches in any two corresponding said succeeding stages are of different length and said vertical shuffle exchange links between switches in any two corresponding said succeeding stages are of different length and $y \geq (\log_2 N)$ where N>1.

25. The integrated circuit device of claim **24**, wherein d=2 and there is only one switch in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there is only one switch in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is rearrangeably nonblocking for unicast generalized multi-stage network with full bandwidth.

26. The integrated circuit device of claim **24**, wherein d=2 and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is strictly nonblocking for unicast generalized multi-stage network and rearrangeably nonblocking for arbitrary fan-out multicast generalized multi-stage network with full bandwidth.

27. The integrated circuit device of claim **24**, wherein d=2 and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is strictly nonblocking for arbitrary fan-out multicast generalized multi-stage network with full bandwidth.

28. The integrated circuit device of claim **4**, wherein said all horizontal shuffle exchange links between switches in any two corresponding said succeeding stages are of different length and said vertical shuffle exchange links between switches in any two corresponding said succeeding stages are of different length and $y \geq (\log_2 N)$, where N>1, and

said each sub-integrated circuit block further comprising a plurality of U-turn links within switches in each of said stages in each of said sub-integrated circuit blocks.

29. The integrated circuit device of claim **28**, wherein d=2 and there is only one switch in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there is only one switch in each said stage in each

US 8,269,523 B2

39   40

said sub-integrated circuit block connecting said backward connecting links and said routing network is rearrangeably nonblocking for unicast generalized butterfly fat tree network with full bandwidth.

30. The integrated circuit device of claim 28, wherein d=2 and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is strictly nonblocking for unicast generalized butterfly fat tree Network and rearrangeably nonblocking for arbitrary fan-out multicast generalized butterfly fat tree network with full bandwidth.

31. The integrated circuit device of claim 28, wherein d=2 and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is strictly nonblocking for arbitrary fan-out multicast generalized butterfly fat tree network with full bandwidth.

32. The integrated circuit device of claim 1, wherein said straight links connecting from switches in each said sub-integrated circuit block are connecting to switches in the same said sub-integrated circuit block; and

said cross links are connecting as vertical or horizontal or diagonal links between two different said sub-integrated circuit blocks.

33. The integrated circuit device of claim 7, wherein d=4 and there is only one switch in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there is only one switch in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is rearrangeably nonblocking for unicast multi-link Benes network with full bandwidth.

34. The integrated circuit device of claim 7, wherein d=4 and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is strictly nonblocking for unicast multi-link Benes network and rearrangeably nonblocking for arbitrary fan-out multicast multi-link Benes network with full bandwidth.

35. The integrated circuit device of claim 7, wherein d=4 and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is strictly nonblocking for arbitrary fan-out multicast multi-link Benes network with full bandwidth.

36. The integrated circuit device of claim 11, wherein d=4 and there is only one switch in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there is only one switch in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is rearrangeably nonblocking for unicast multi-link butterfly fat tree network with full bandwidth.

37. The integrated circuit device of claim 11, wherein d=4 and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said forward

connecting links and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is strictly nonblocking for unicast multi-link butterfly fat tree network and rearrangeably nonblocking for arbitrary fan-out multicast multi-link butterfly fat tree network with full bandwidth.

38. The integrated circuit device of claim 11, wherein d=4 and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is strictly nonblocking for arbitrary fan-out multicast multi-link butterfly fat tree network with full bandwidth.

39. The integrated circuit device of claim 4, wherein said all horizontal shuffle exchange links between switches in any two corresponding said succeeding stages are of different length and said vertical shuffle exchange links between switches in any two corresponding said succeeding stages are of different length and $y \geqq (\log_2 N)$, where N>1.

40. The integrated circuit device of claim 39, wherein d=4 and there is only one switch in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there is only one switch in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is rearrangeably nonblocking for unicast generalized multi-link multi-stage network with full bandwidth.

41. The integrated circuit device of claim 39, wherein d=4 and there are at least two switches in each said stage in each said sub-integrated circuit block) connecting said forward connecting links and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is strictly nonblocking for unicast generalized multi-link multi-stage network and rearrangeably nonblocking for arbitrary fan-out multicast generalized multi-link multi-stage network with full bandwidth.

42. The integrated circuit device of claim 39, wherein d=4 and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is strictly nonblocking for arbitrary fan-out multicast generalized multi-link multi-stage network with full bandwidth.

43. The integrated circuit device of claim 4, wherein said all horizontal shuffle exchange links between switches in any two corresponding said succeeding stages are of different length and said vertical shuffle exchange links between switches in any two corresponding said succeeding stages are of different length and $y \geqq (\log_2 N)$, where N>1, and

said each sub-integrated circuit block further comprising a plurality of U-turn links within switches in each of said stages in each of said sub-integrated circuit blocks.

44. The integrated circuit device of claim 43, wherein d=4 and there is only one switch in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there is only one switch in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is rearrangeably nonblocking for unicast generalized multi-link butterfly fat tree network with full bandwidth.

US 8,269,523 B2

<table>
<tr><td>41</td><td>42</td></tr>
</table>

45. The integrated circuit device of claim **43**, wherein d=4 and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is strictly nonblocking for unicast generalized multi-link butterfly fat tree Network and rearrangeably nonblocking for arbitrary fan-out multicast generalized multi-link butterfly fat tree network with full bandwidth.

46. The integrated circuit device of claim **43**, wherein d=4 and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is strictly nonblocking for arbitrary fan-out multicast generalized multi-link butterfly fat tree network with full bandwidth.

47. The integrated circuit device of claim **1**, wherein said plurality of forward connecting links use a plurality of buffers to amplify signals driven through them and said plurality of backward connecting links use a plurality of buffers to amplify signals driven through them; and said buffers can be inverting or non-inverting buffers.

48. The integrated circuit device of claim **1**, wherein said all switches of size d×d are either fully populated or partially populated.

* * * * *

FIG. 1A
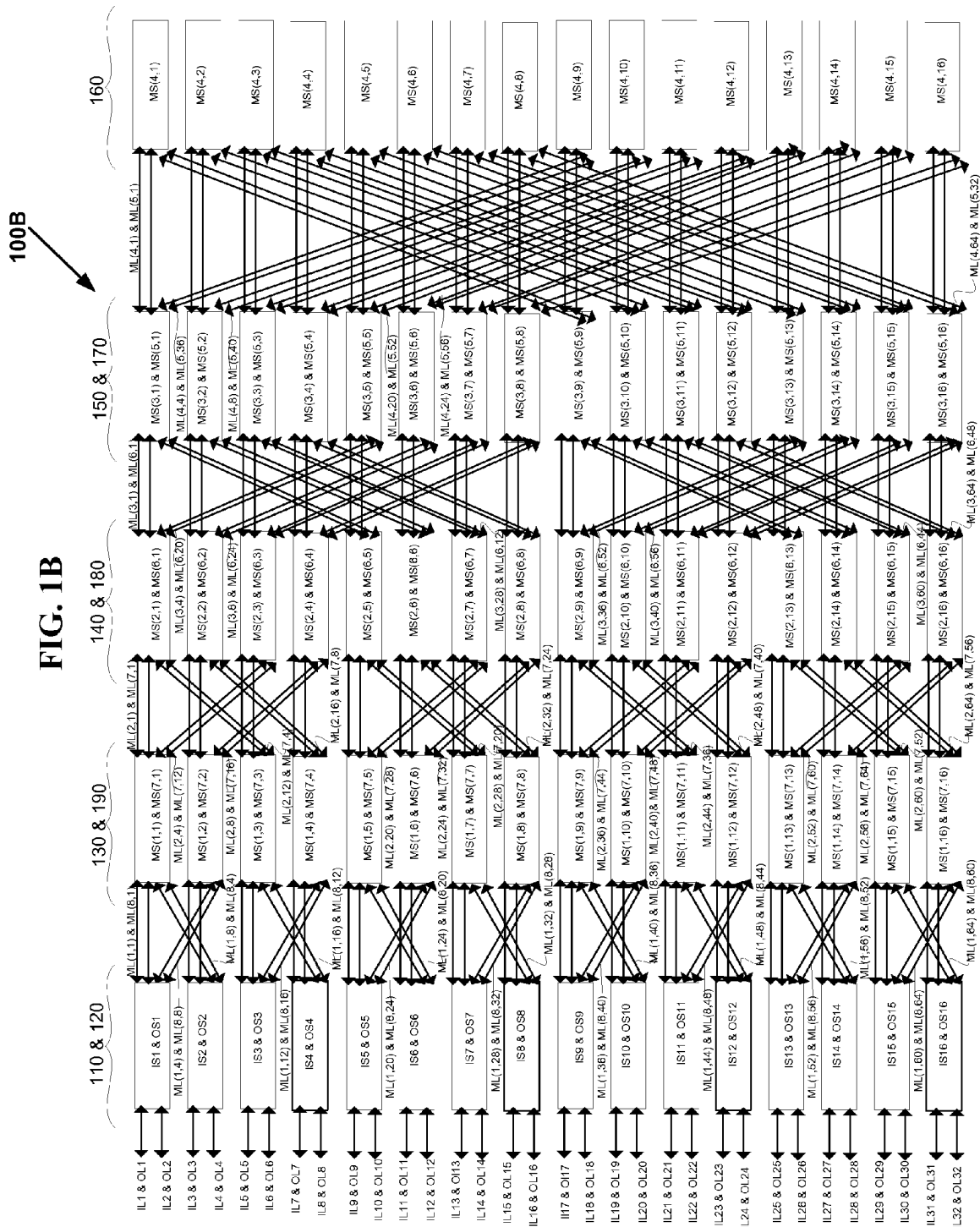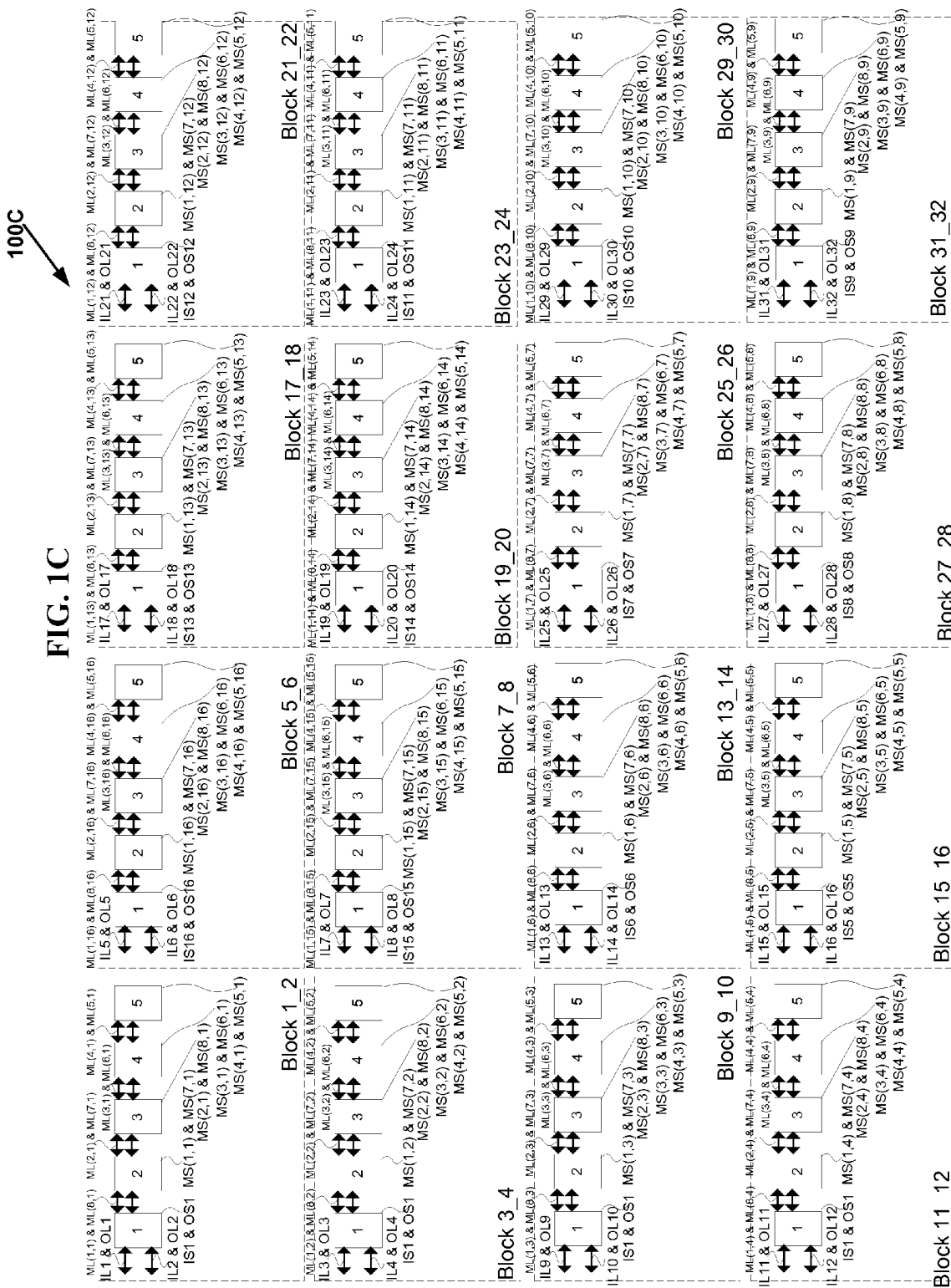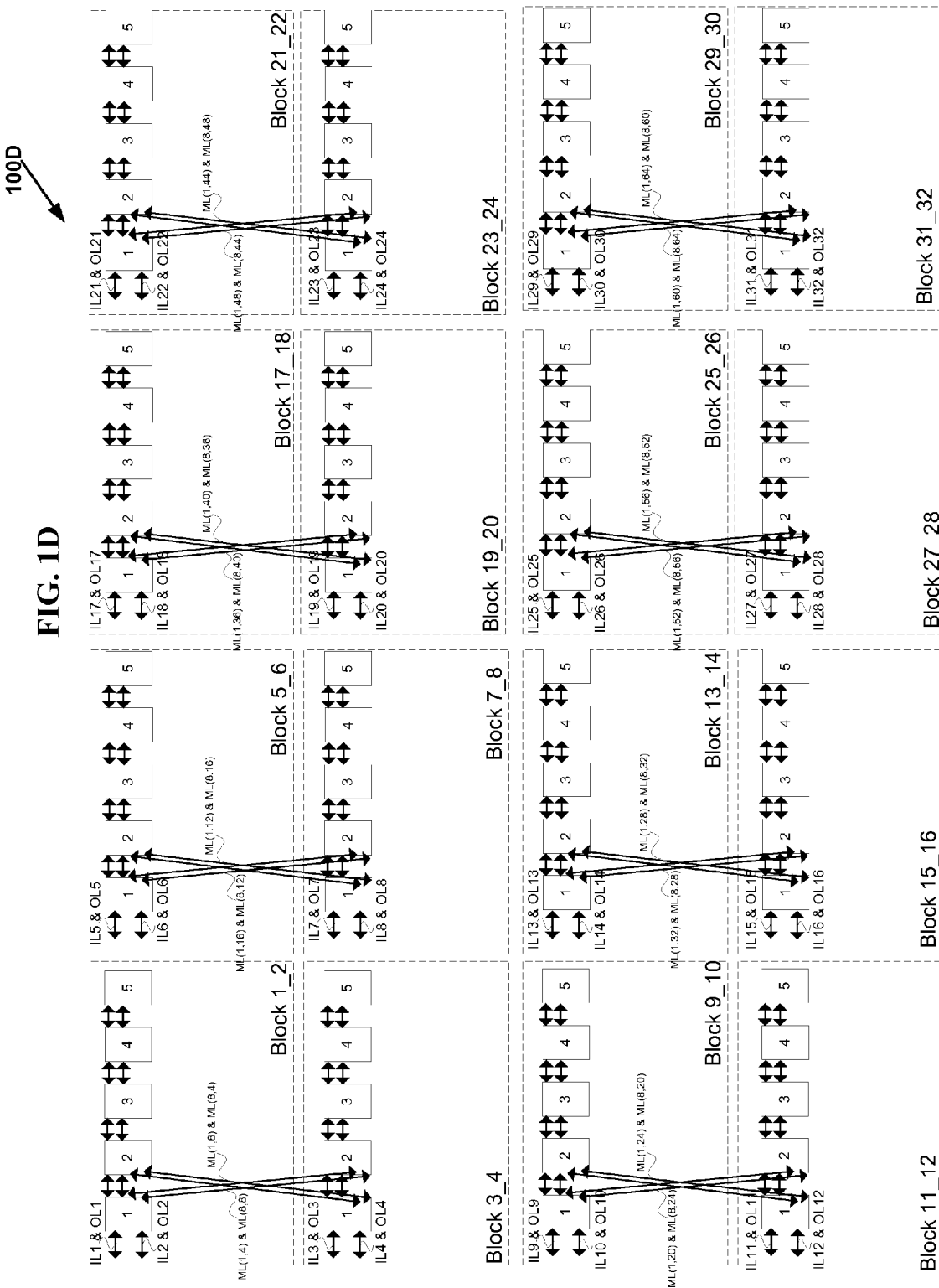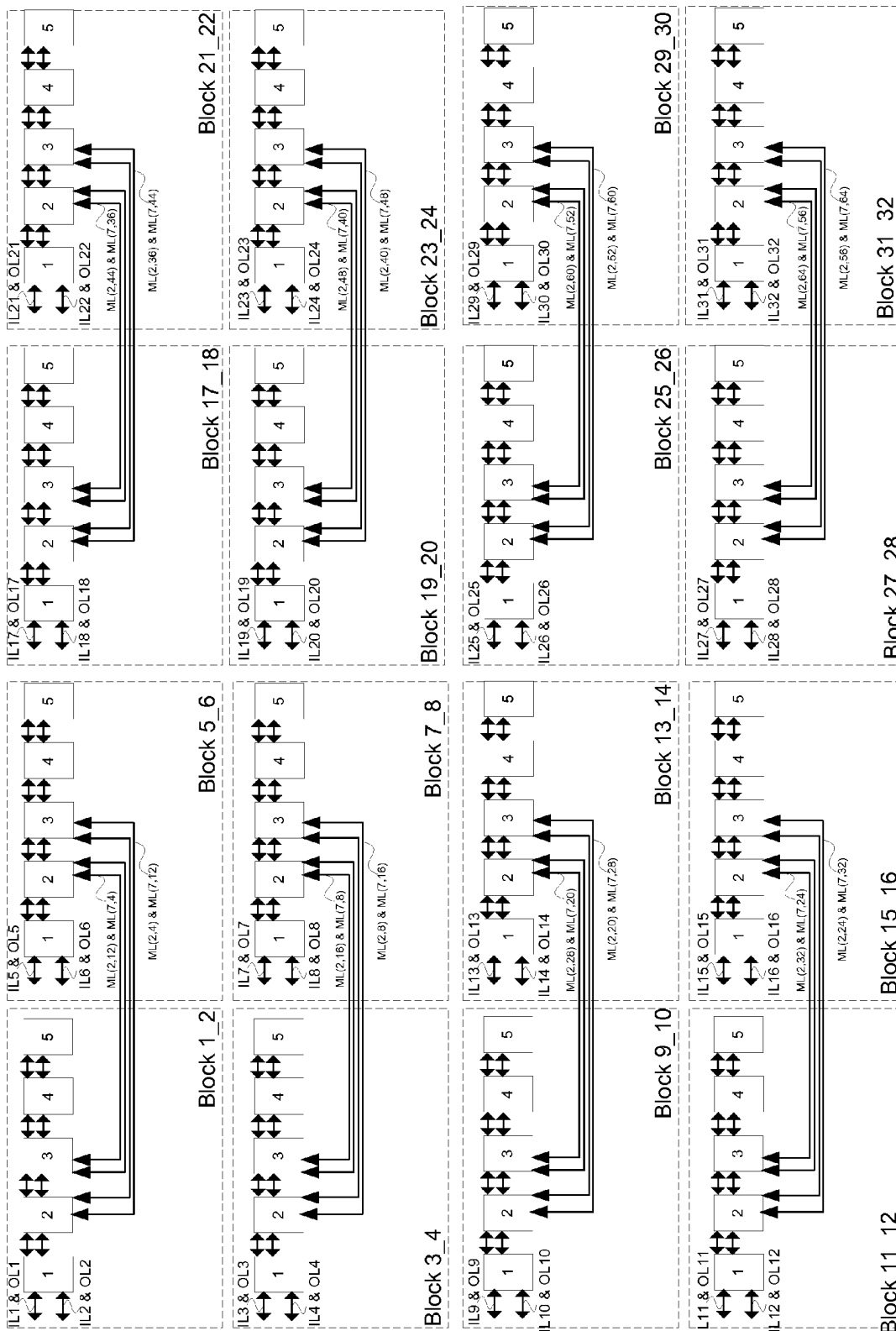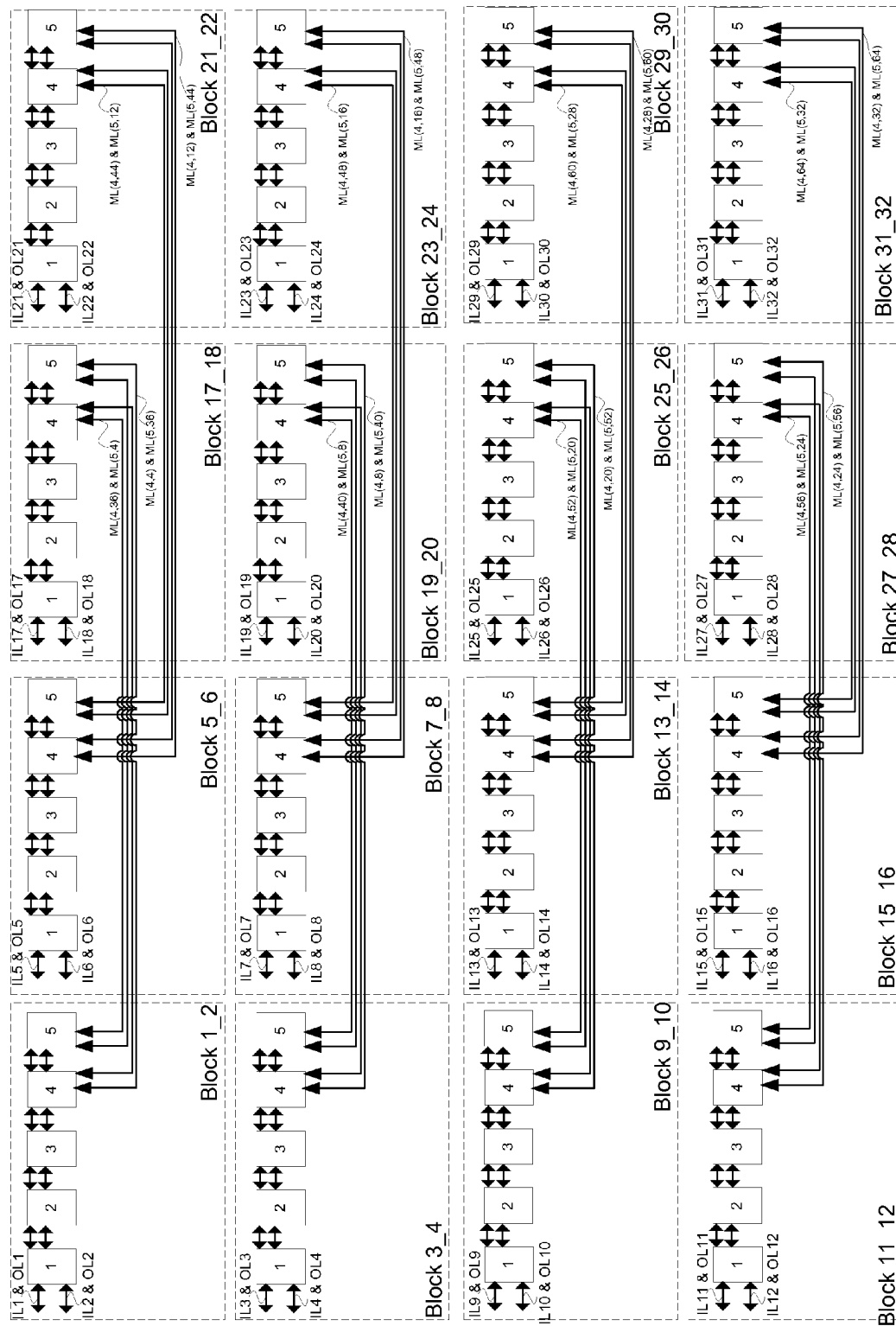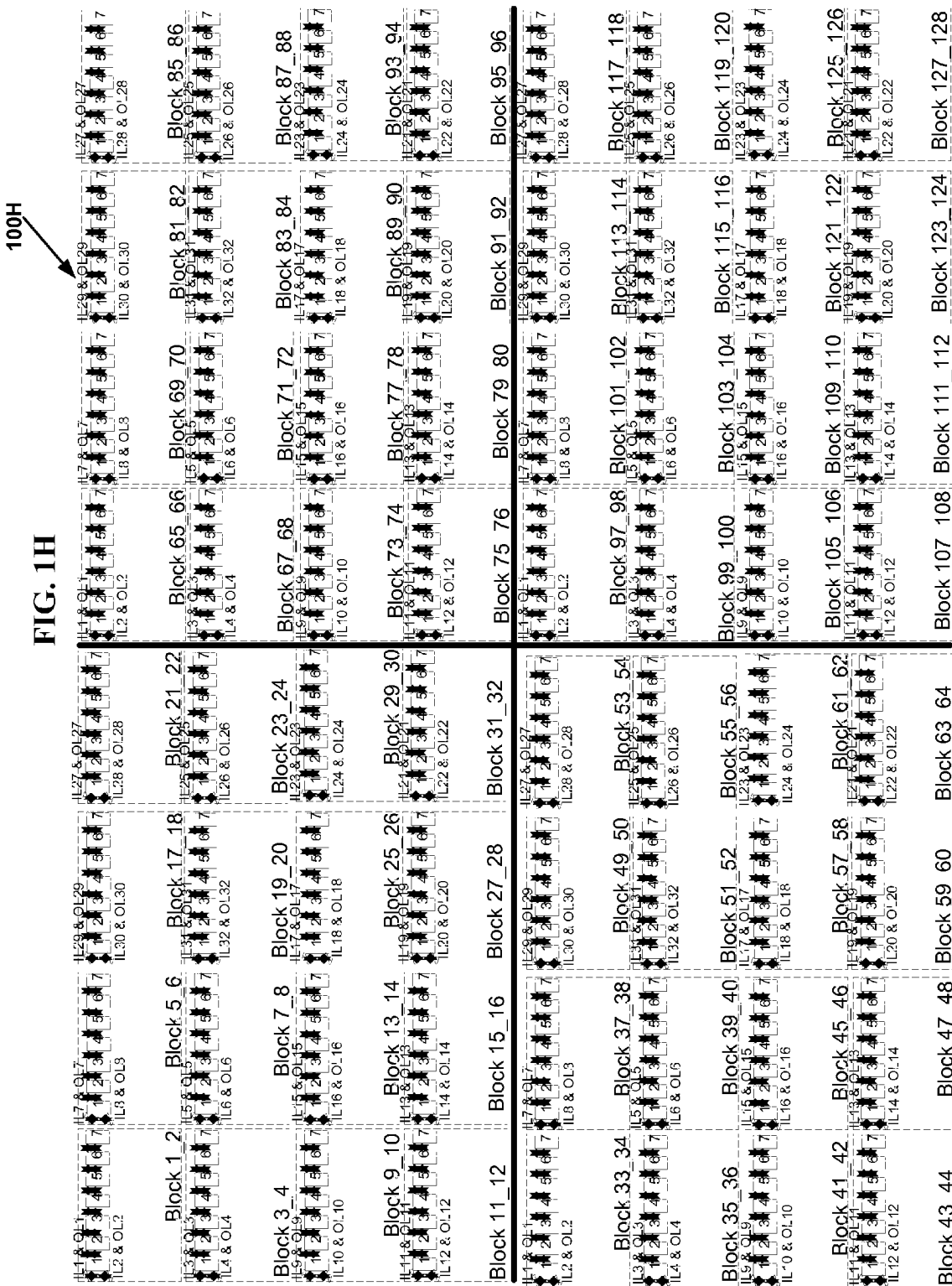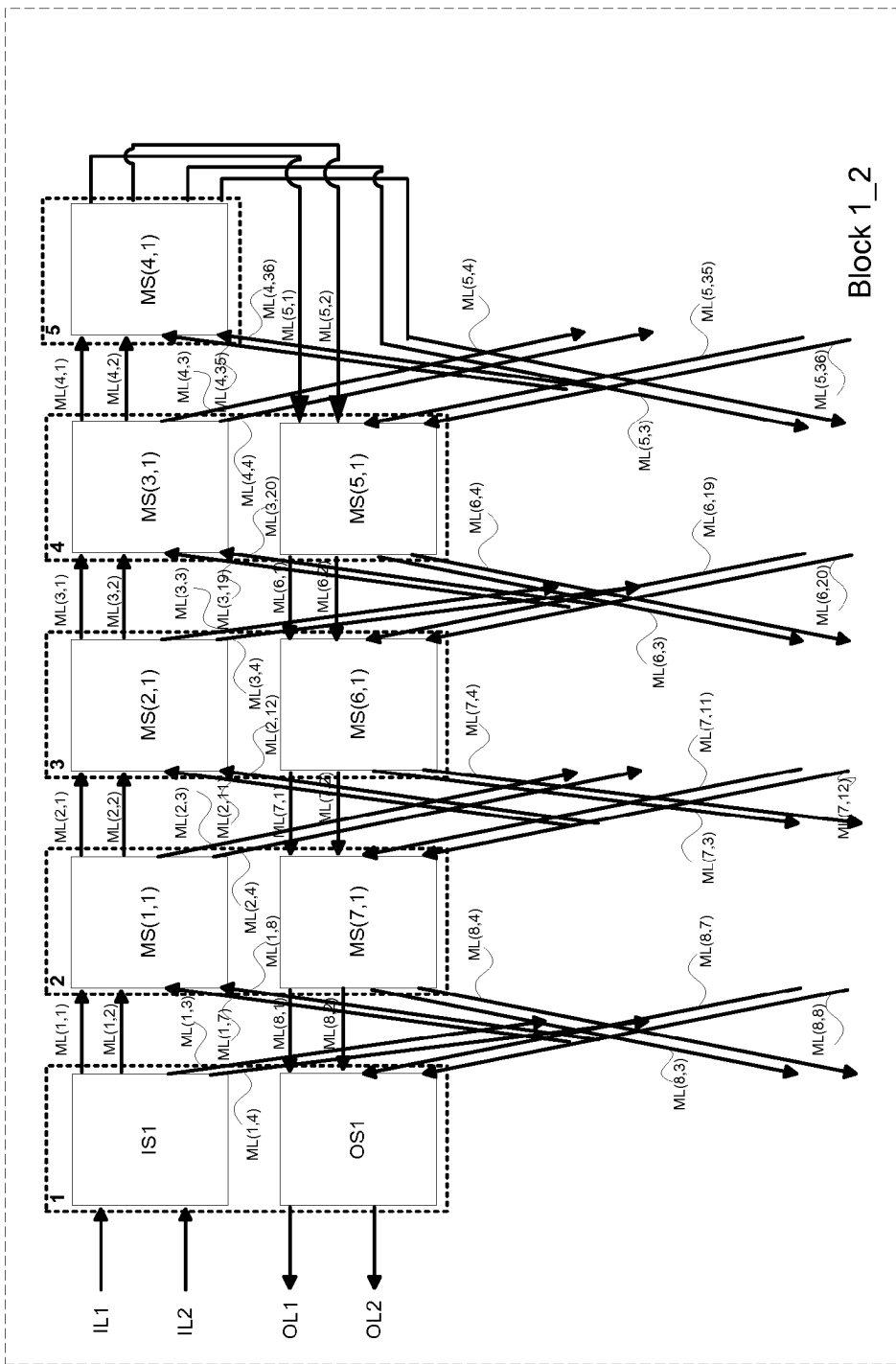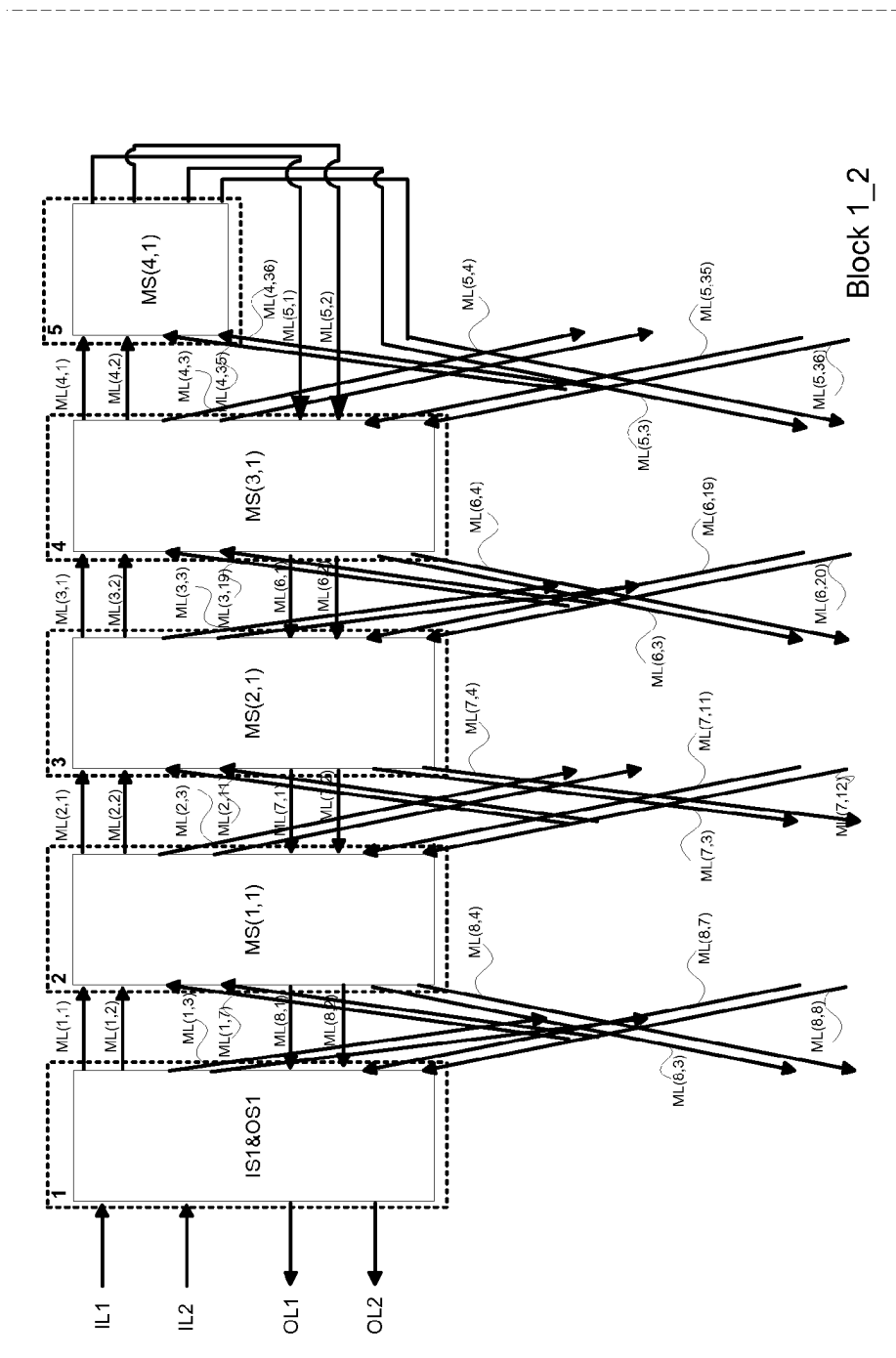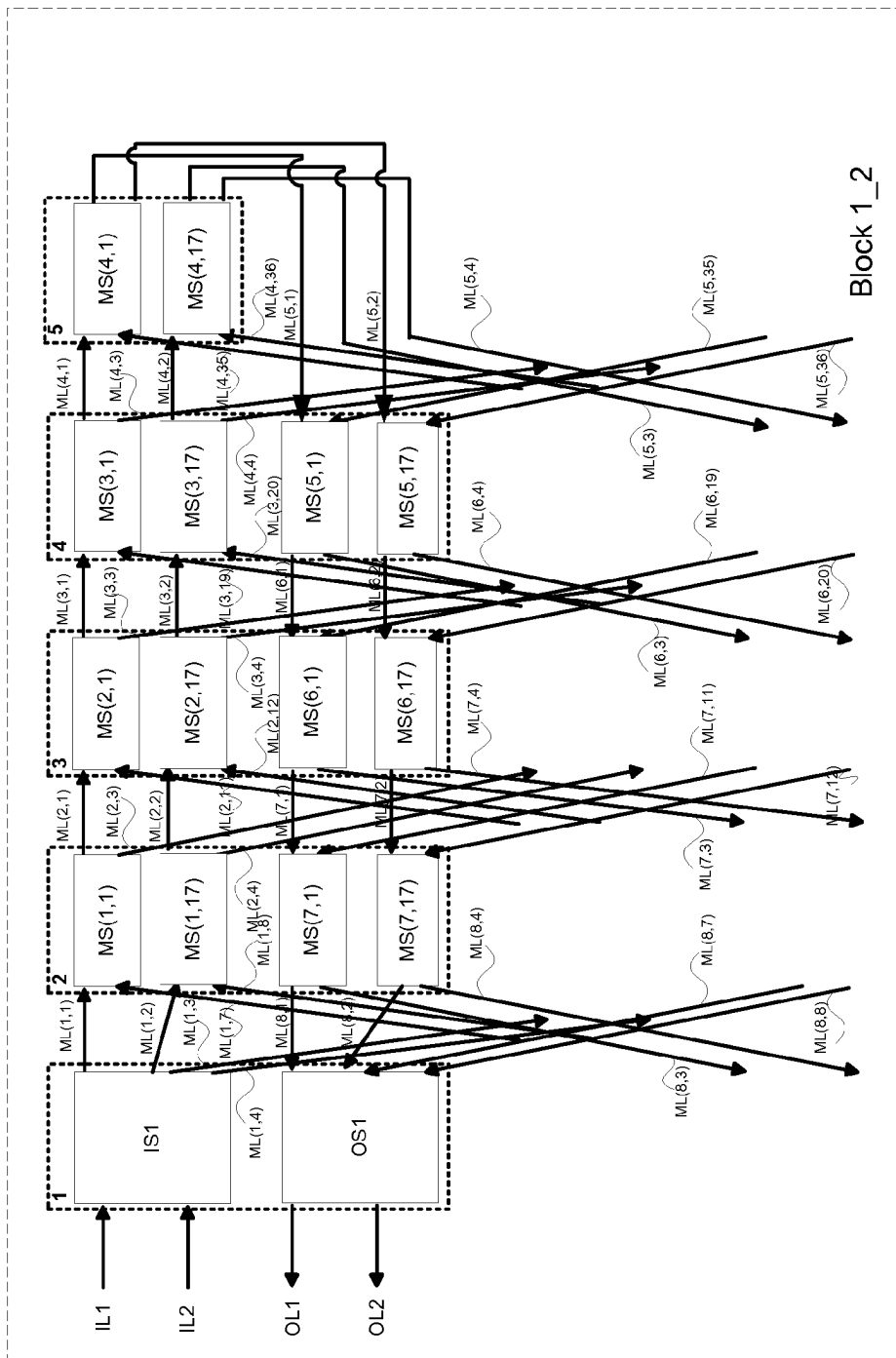
FIG. 1B

FIG.1C

FIG. 1D

FIG. 1E

FIG. 1F

FIG. 1G

FIG. 1H

100H

VENKAT KONDA EXHIBIT 2005

FIG. 1I

FIG. 1J

FIG. 1K

100K

Block 1_2

FIG. 1K1

100K1

Block 1_2

FIG. 1L

FIG. 1L1

FIG. 2A2

200A2

110 & 120

IS1 & OS1

IL1 & OL1

IL2 & OL2

FIG. 2A3

200A3

Block 1_2

IL1 & OL1

1

IL2 & OL2

FIG. 2A1

200A1

110 & 120

IS1 & OS1

OL1

OL2

IL1

IL2

FIG. 2B1

FIG. 2B2

FIG. 2B3

FIG. 2B4

FIG. 2C11

FIG. 2C12

FIG. 2C21

FIG. 2C22

FIG. 2C23

FIG. 2D1

FIG. 2D2

FIG. 2D3

200D3

FIG. 2D4

200D4

FIG. 2D5

FIG. 2D6

FIG. 3A

FIG. 3B

FIG. 3C

FIG. 3D

FIG. 3E

300E

FIG. 3F

FIG. 3G

FIG. 3H

300H

FIG. 4A

400A

FIG. 4B

FIG. 4C

FIG. 4D

FIG. 4E

FIG. 4C1

FIG. 5A

500A2

IL1

IL2

B1

OL1

OL2

P(1,1)

C(1,1)

P(2,1)

C(2,1)

P(1,2)

C(1,2)

P(2,2)

C(2,2)

FIG. 5A2
(Prior Art)

500A4

IL1

IL2

OL1

OL2

DCP(1,1)

DCP(2,2)

FIG. 5A4

500A1

IL1

IL2

OL1

OL2

CP(1,1)

CP(2,1)

CP(1,2)

CP(2,2)

FIG. 5A1
(Prior Art)

500A3

IL1

IL2

OL1

OL2

V(1,1)

V(2,1)

V(1,2)

V(2,2)
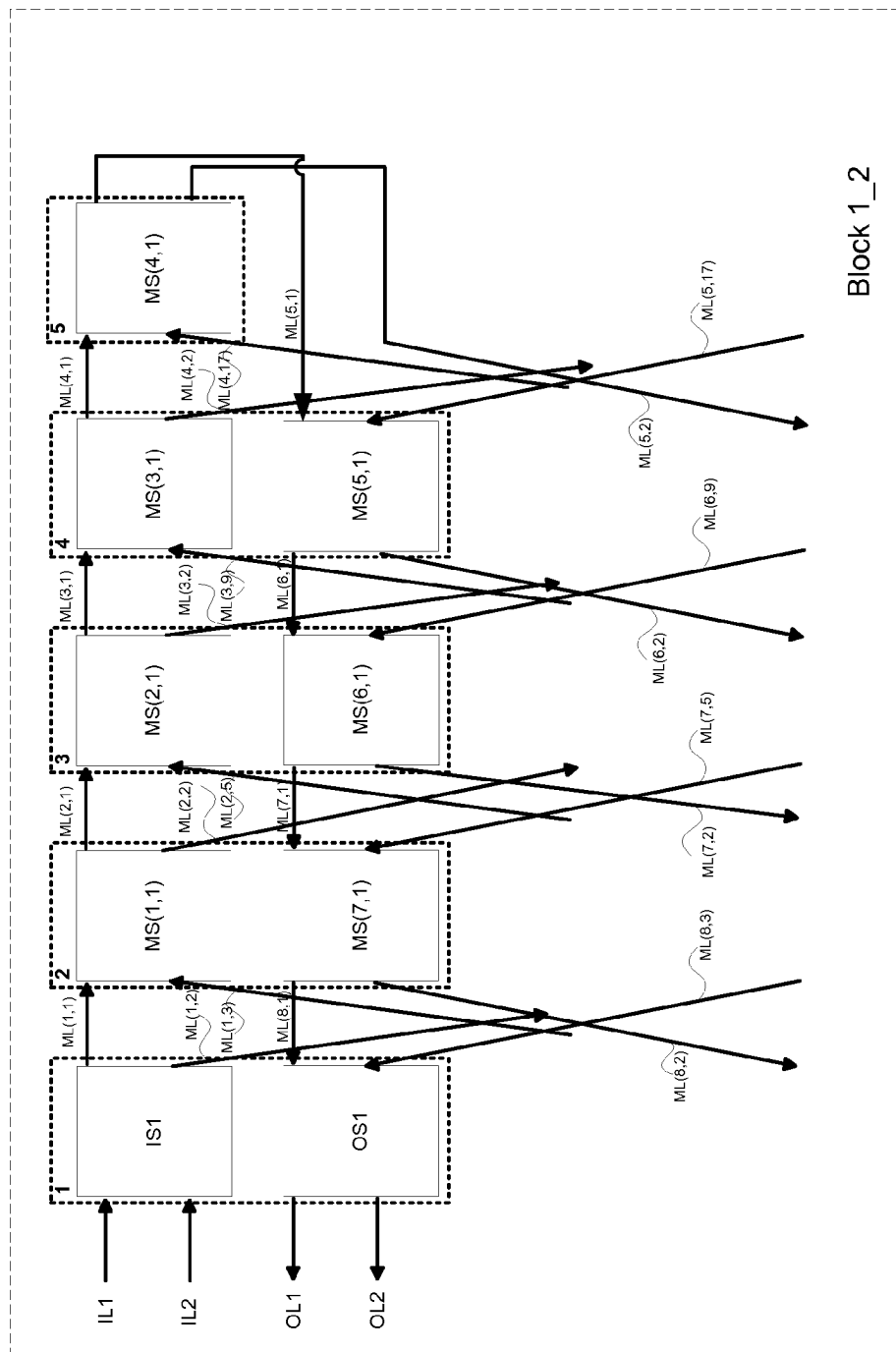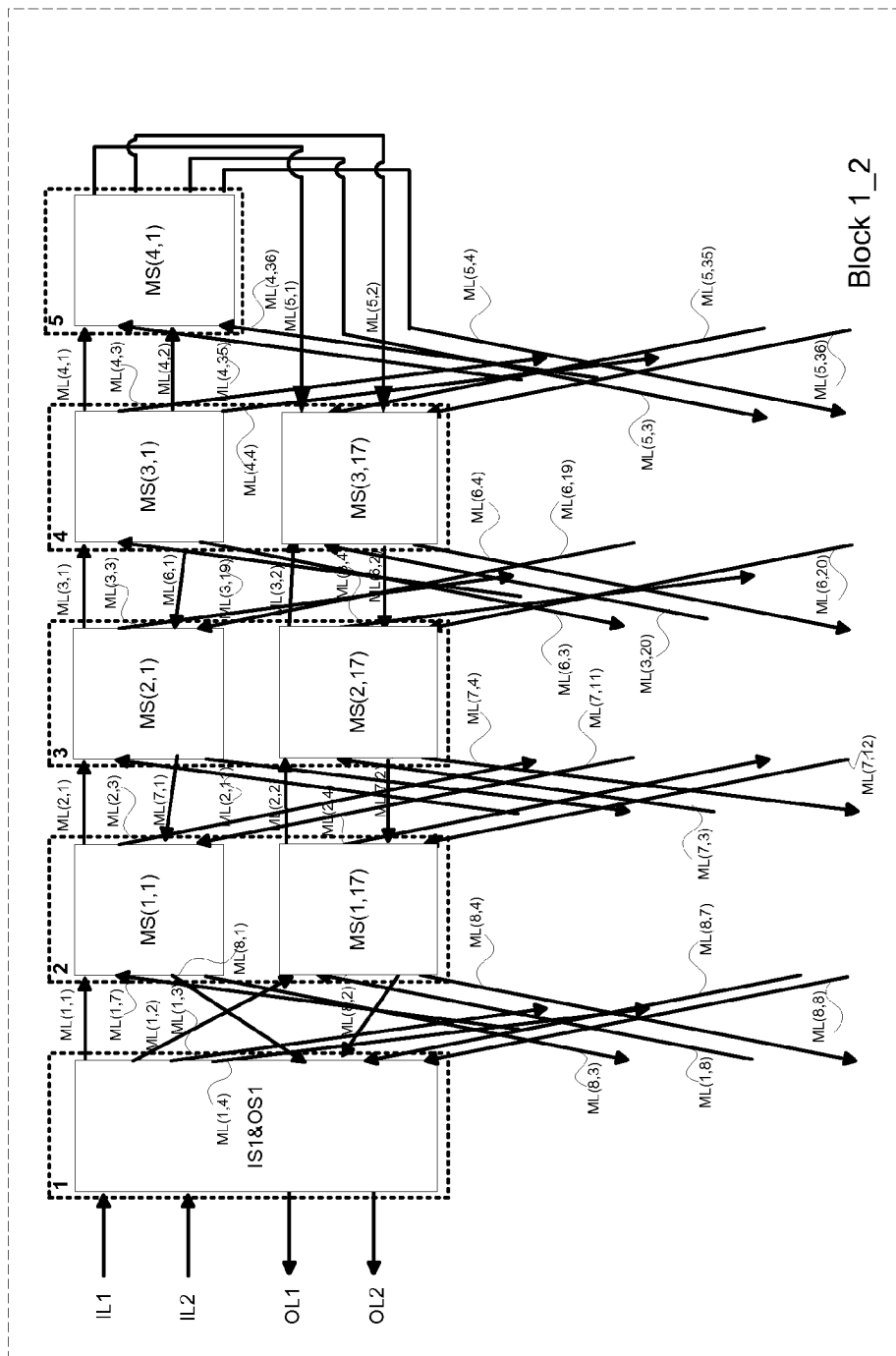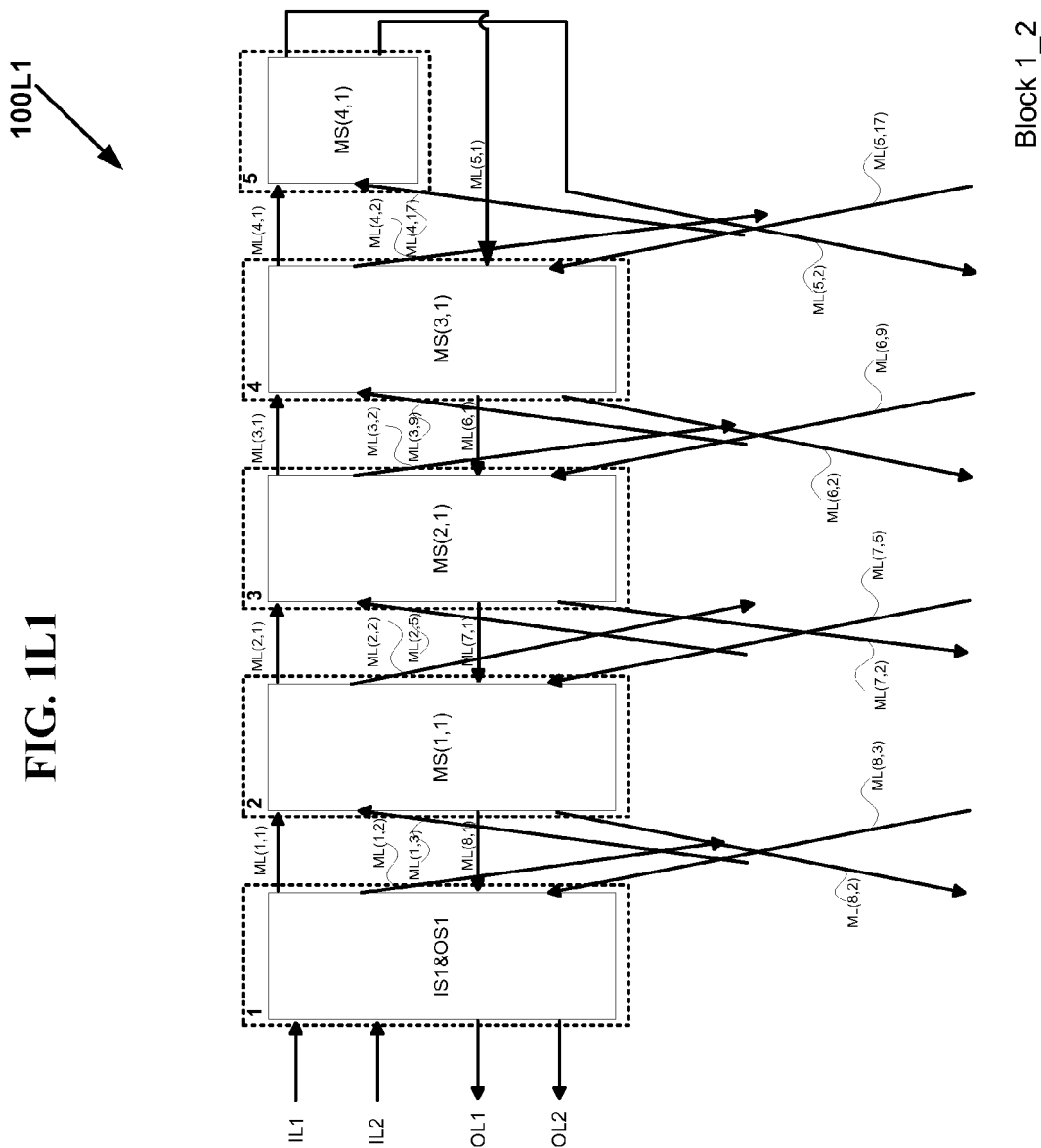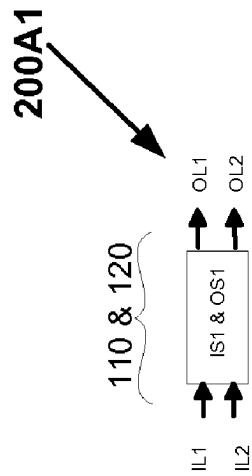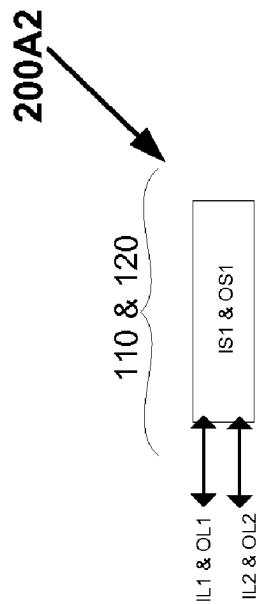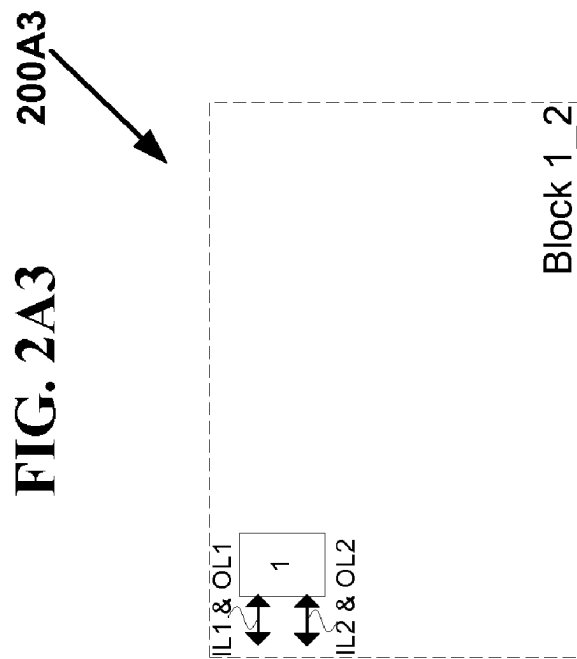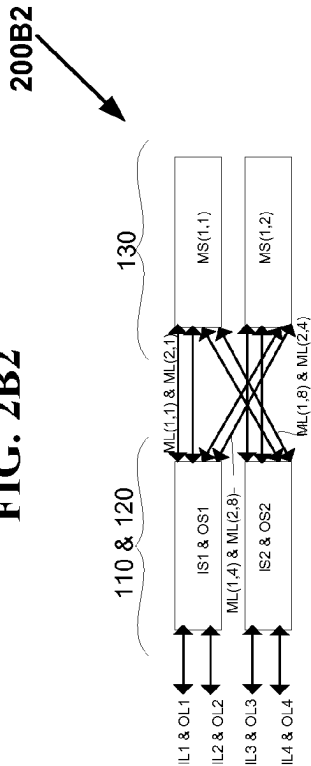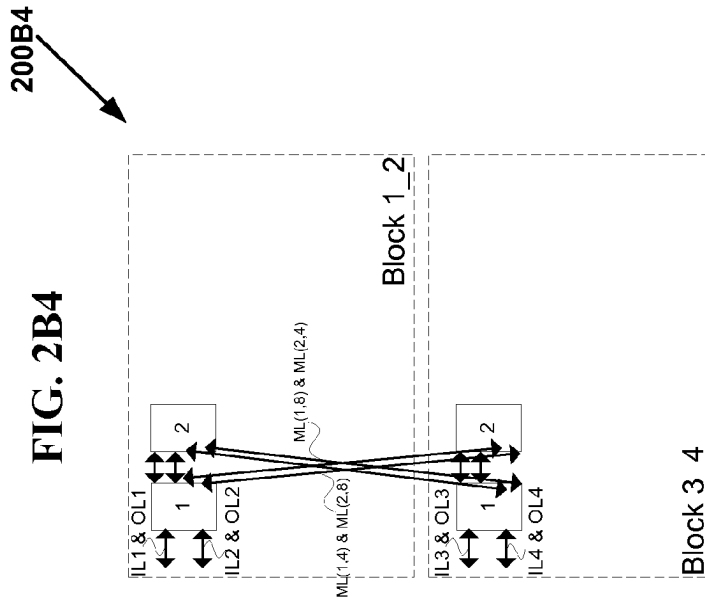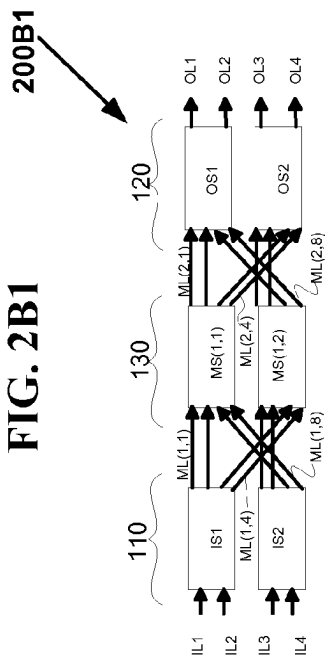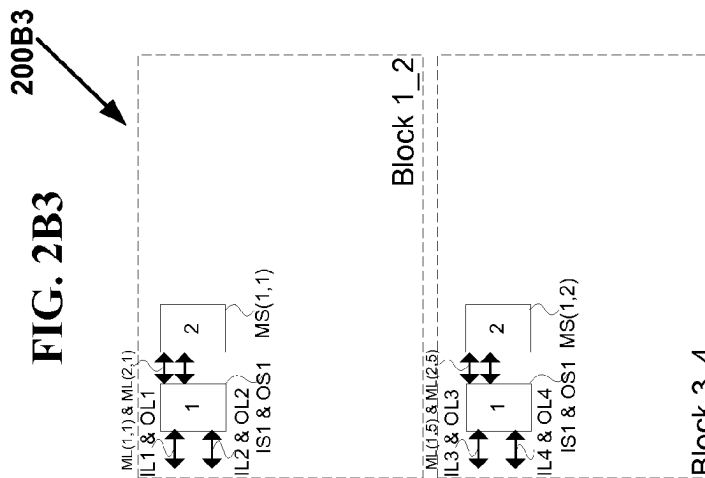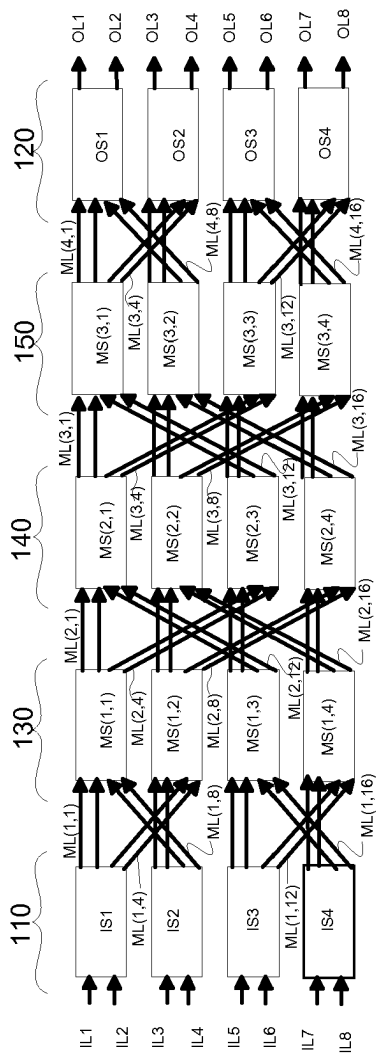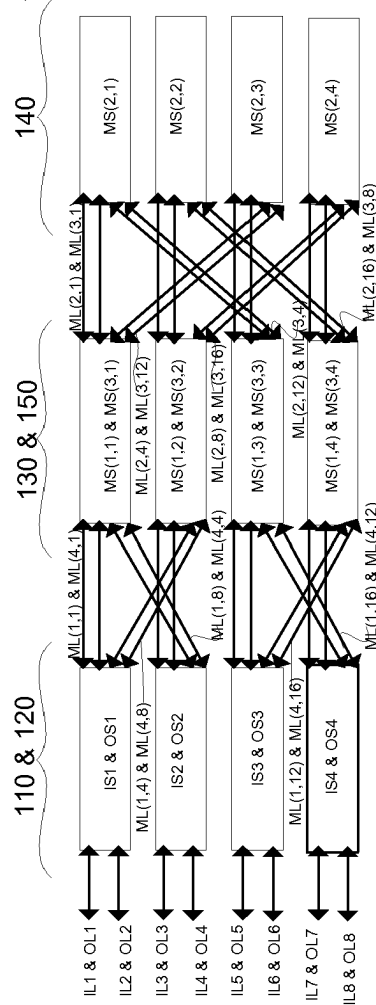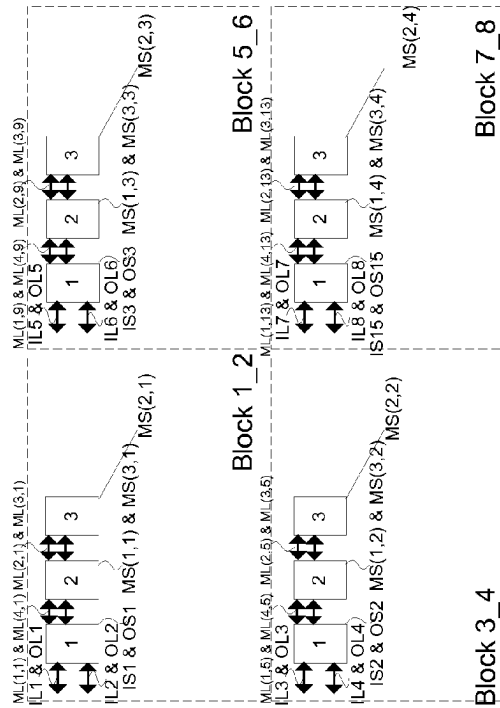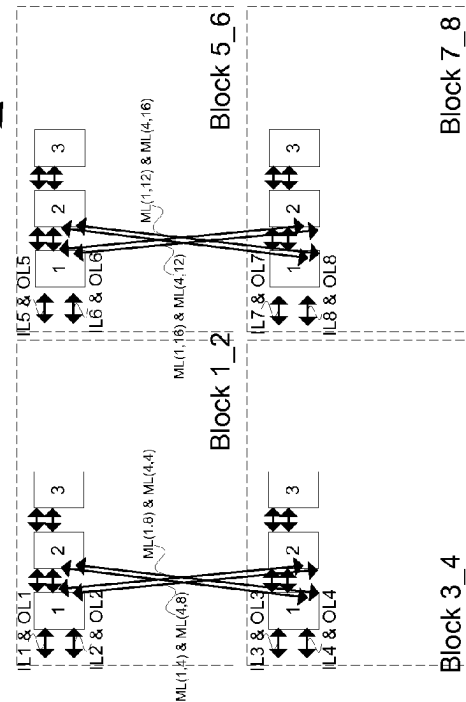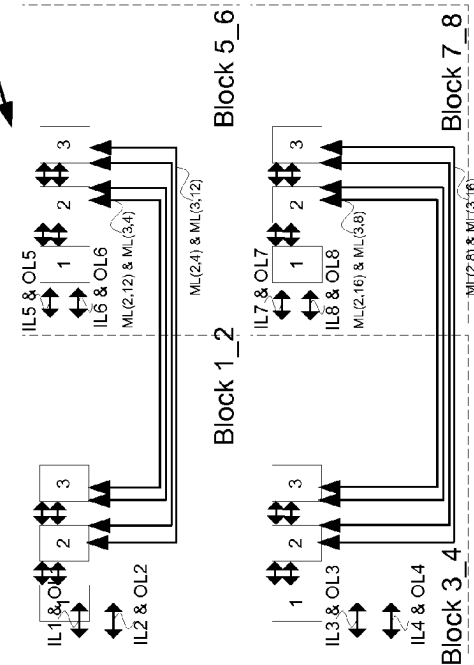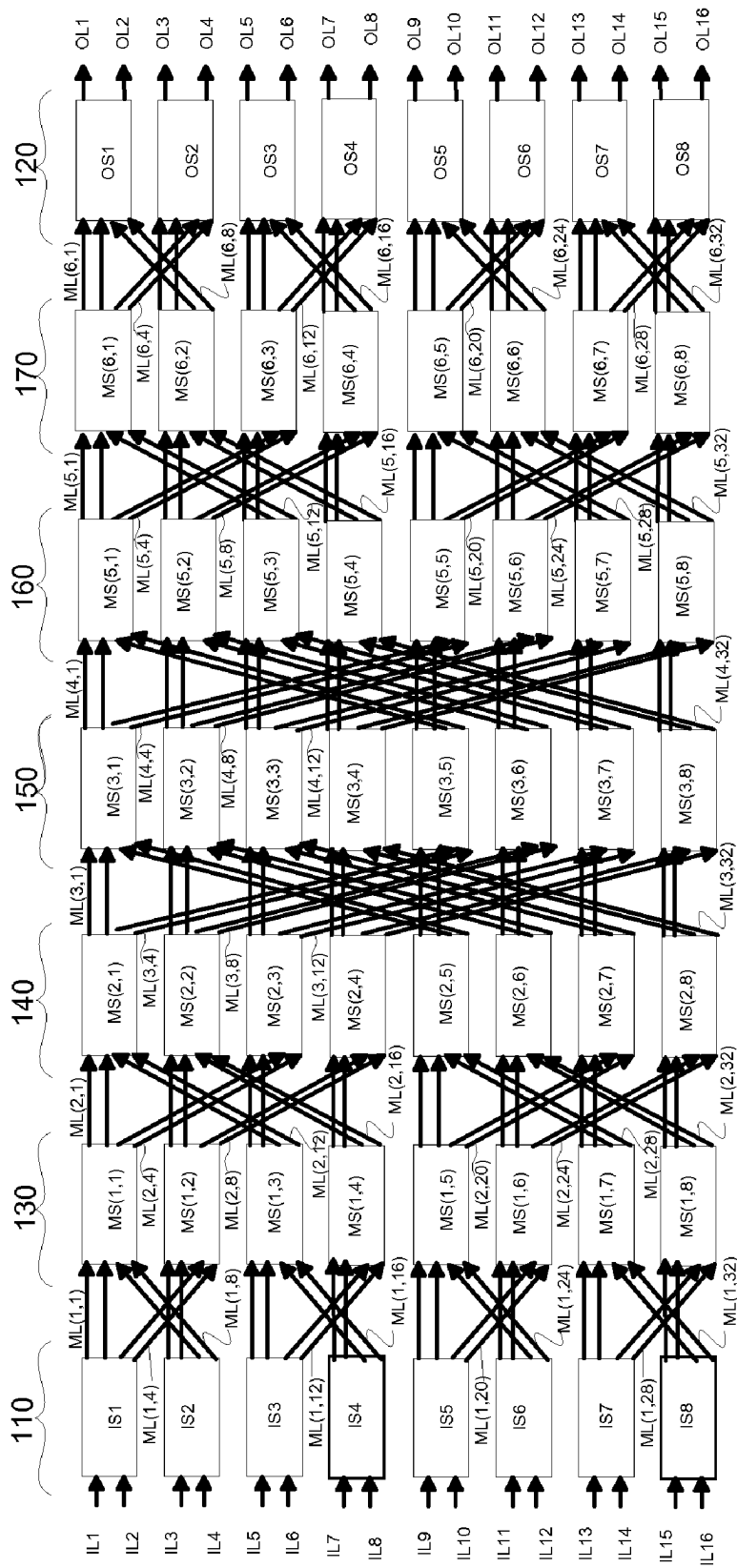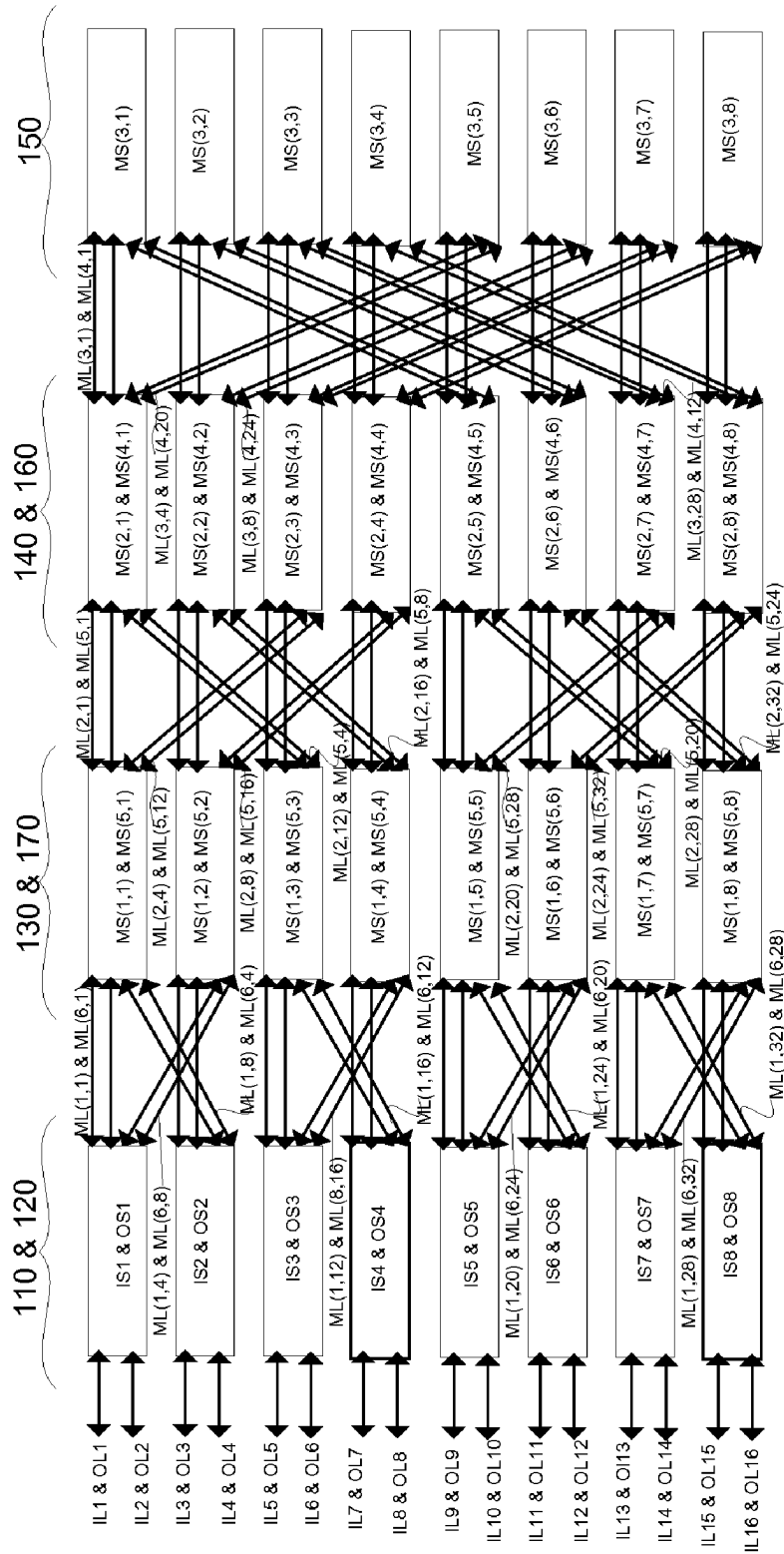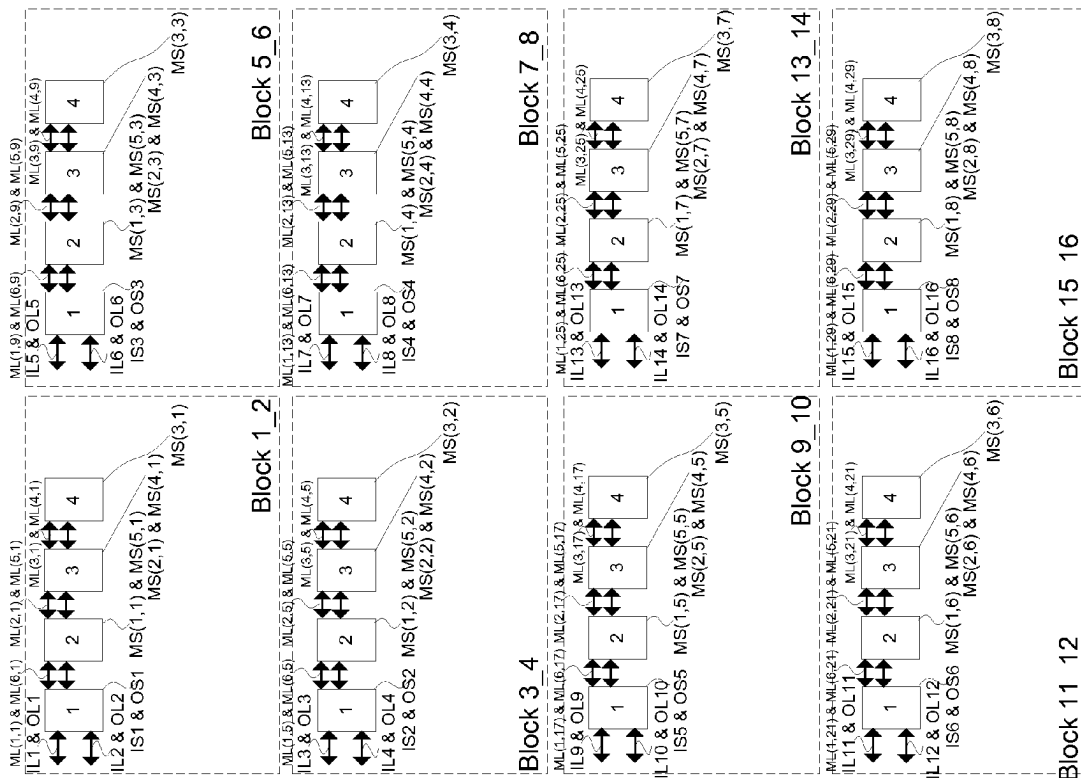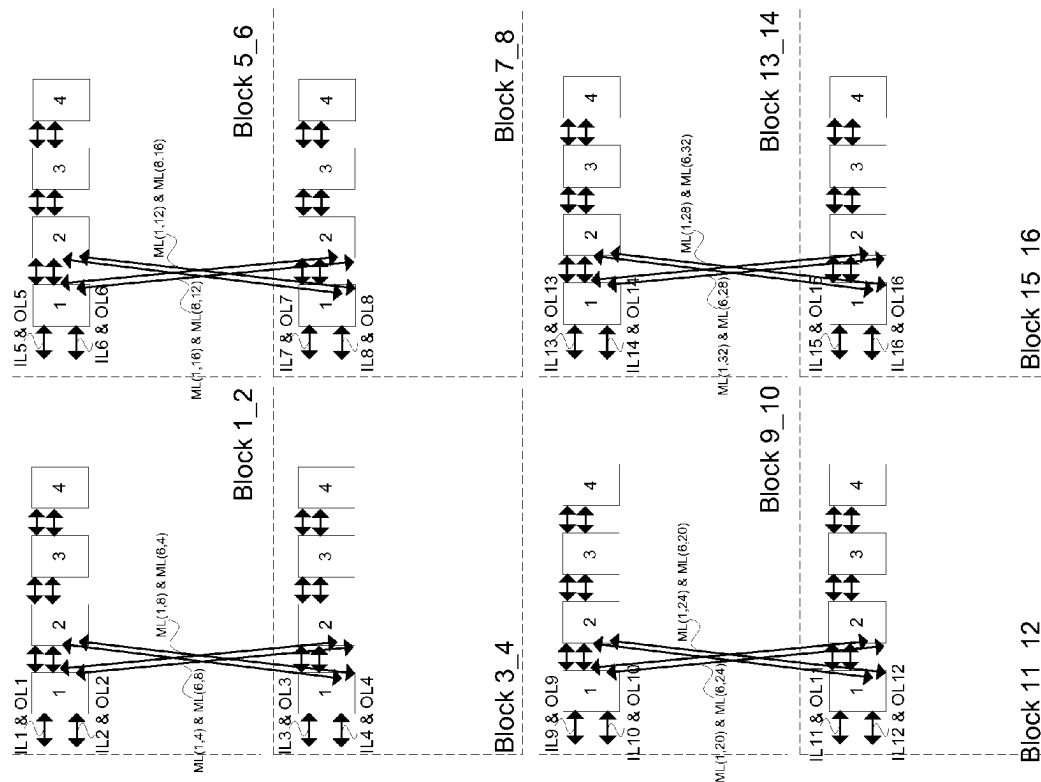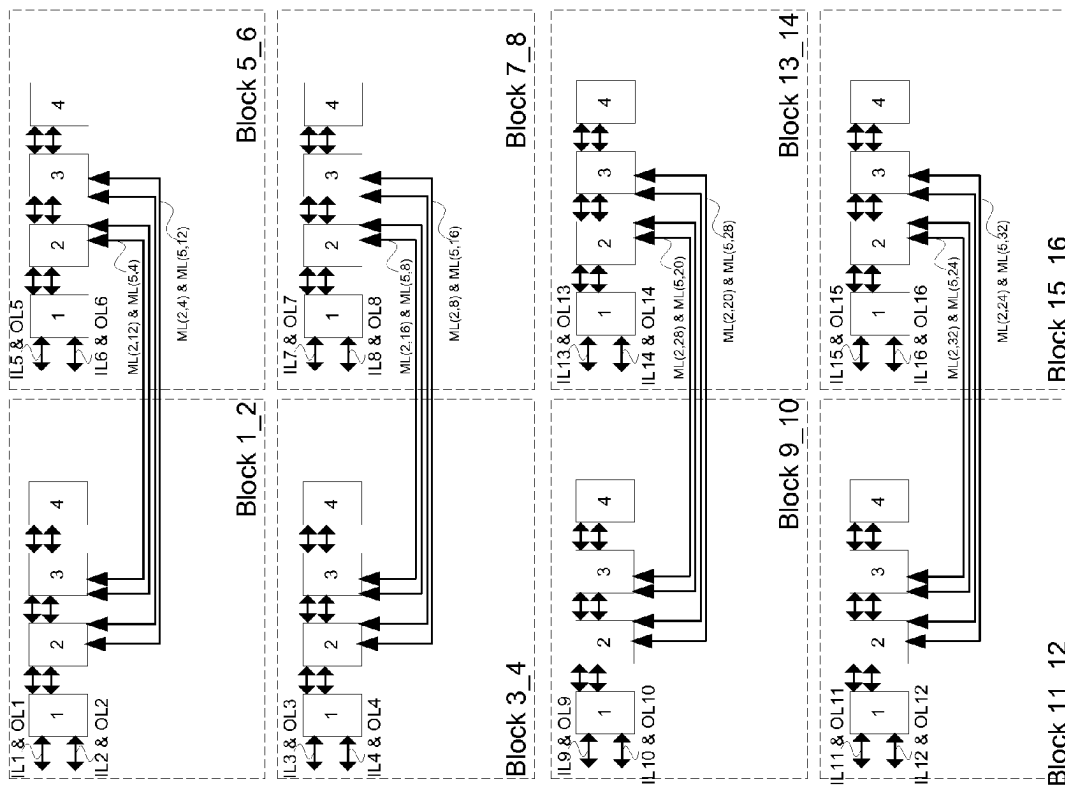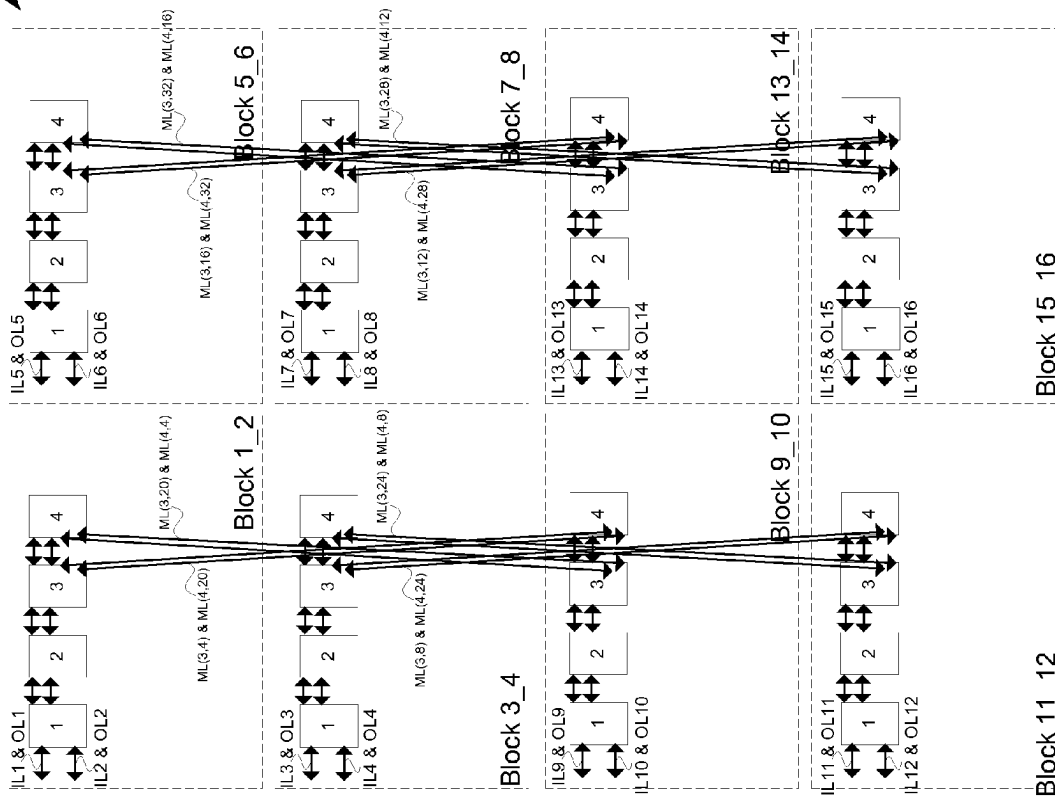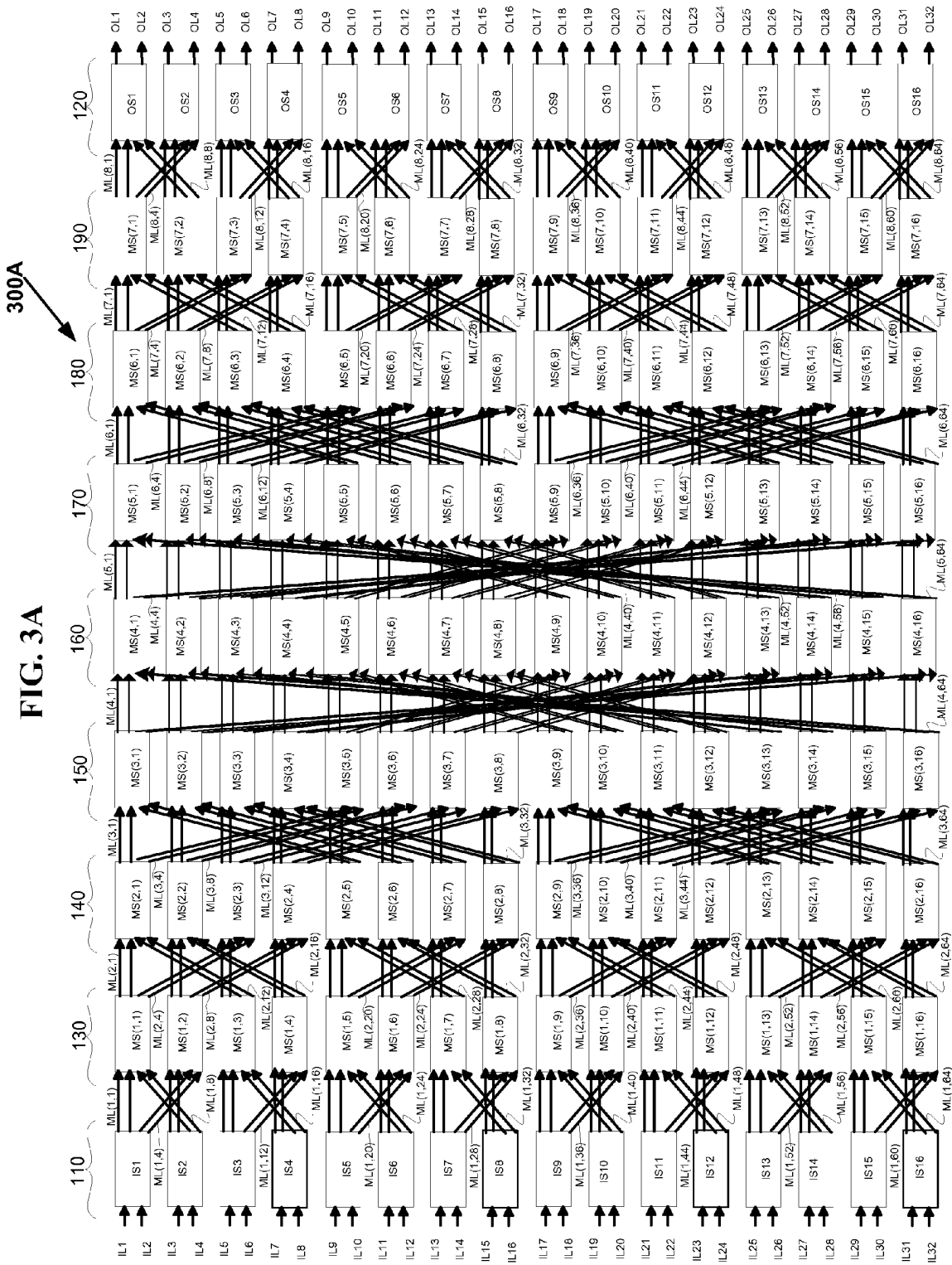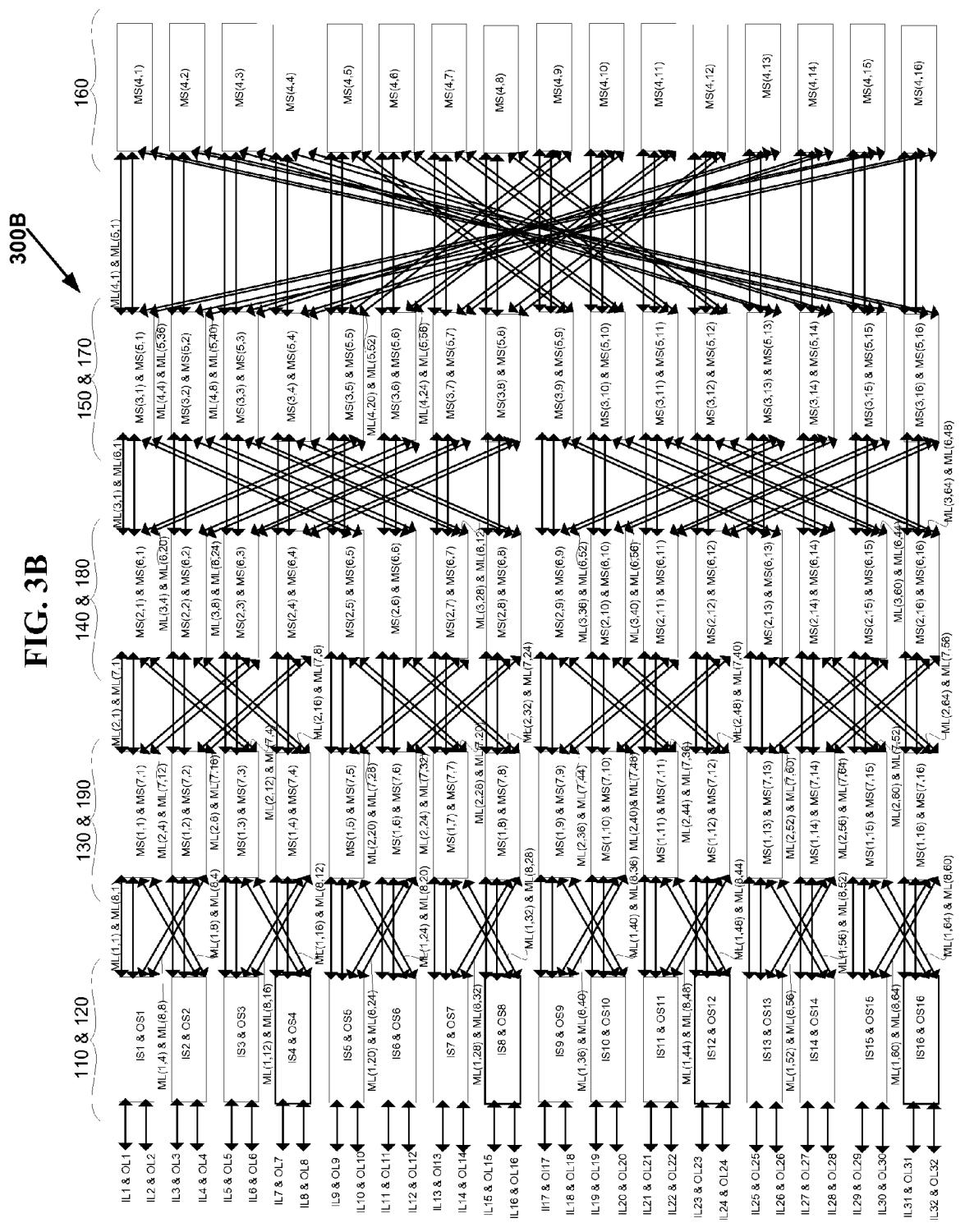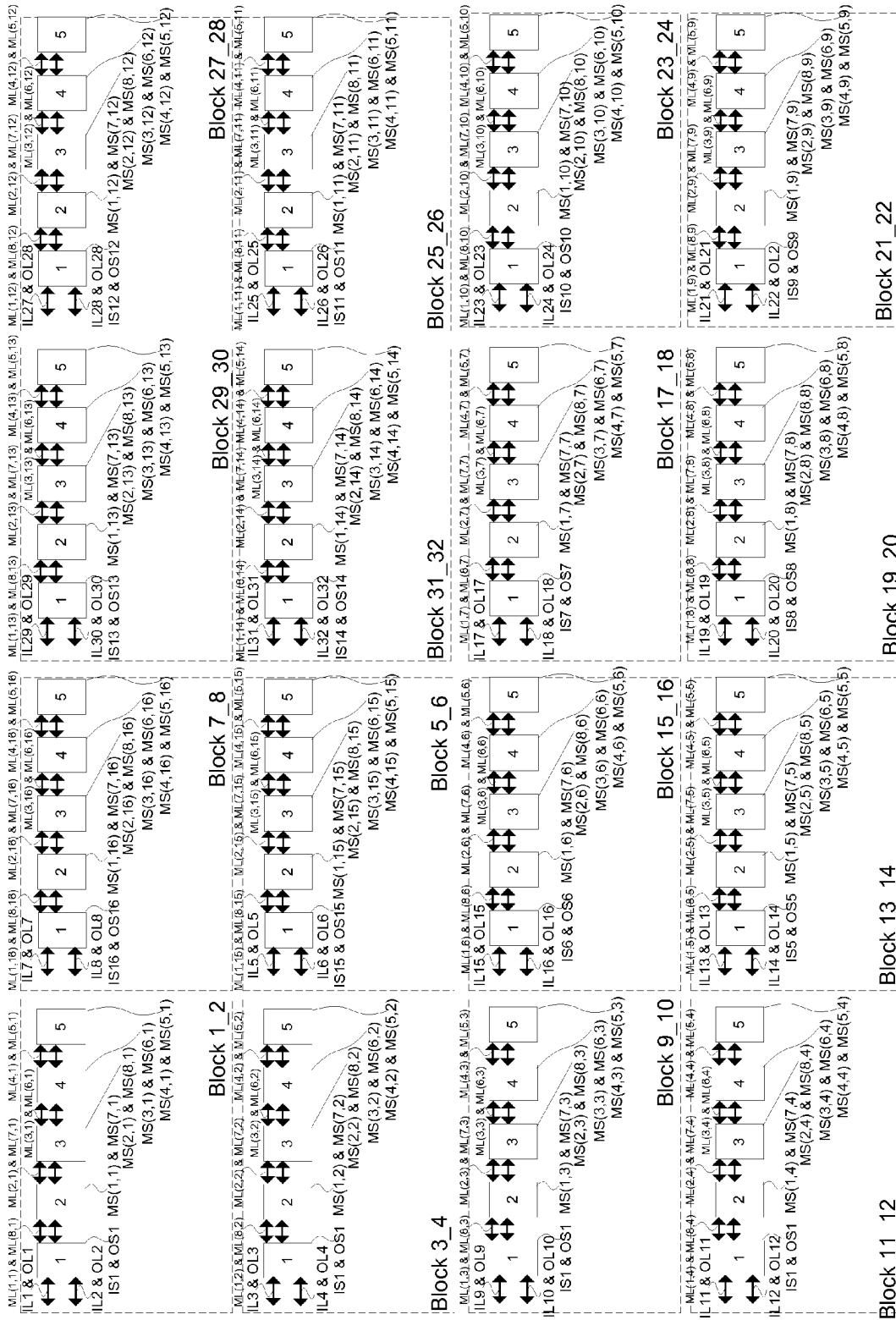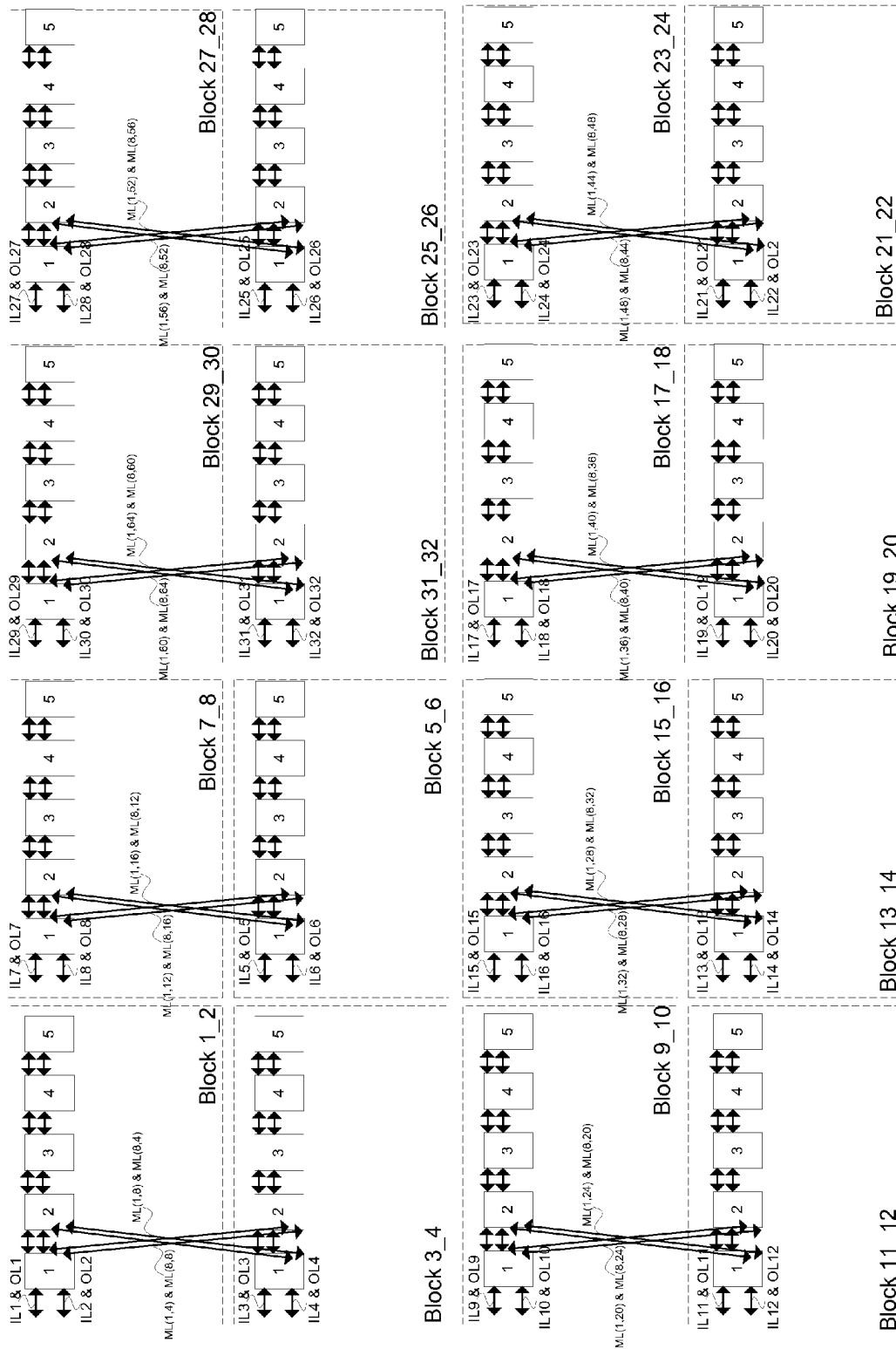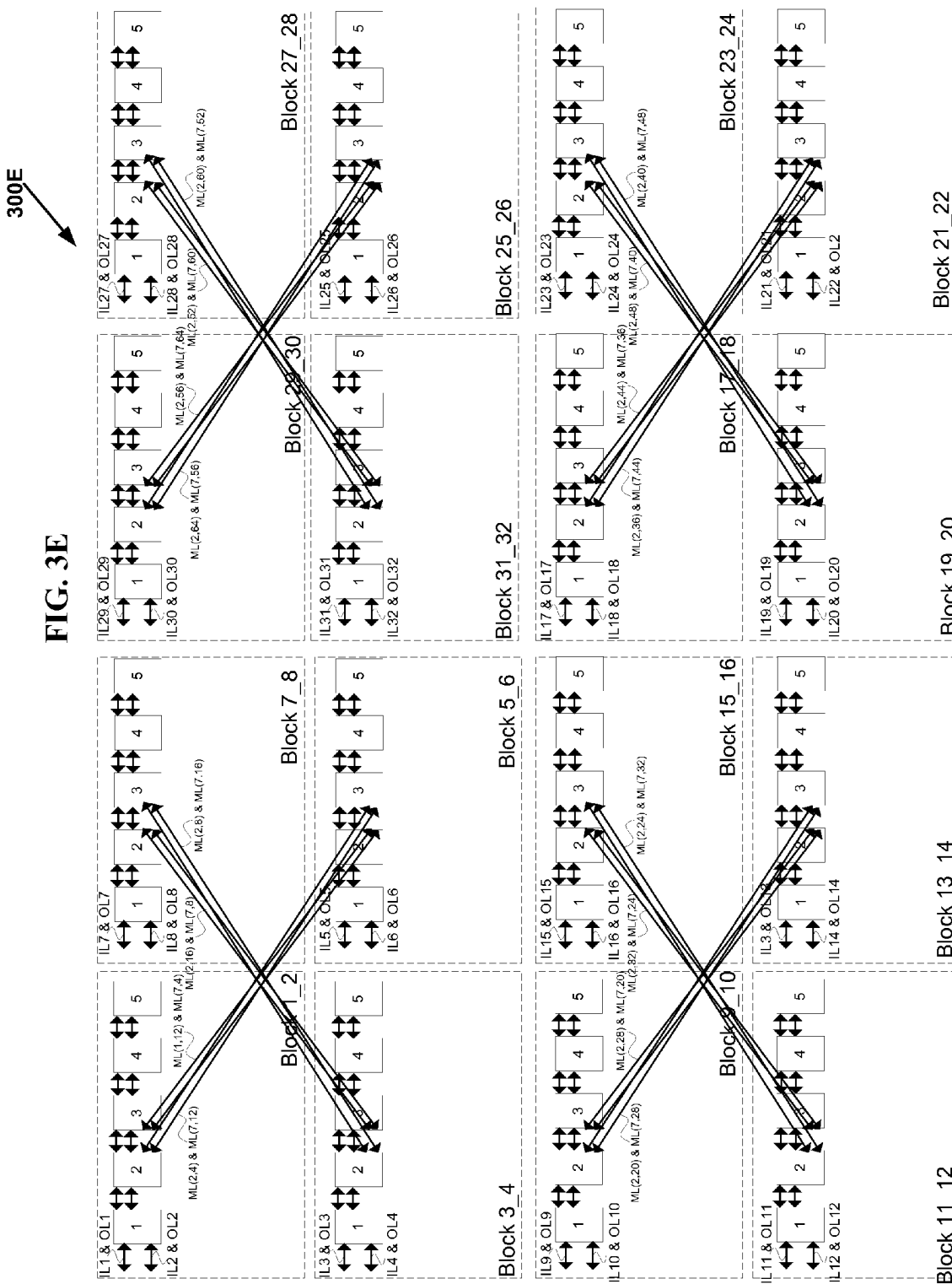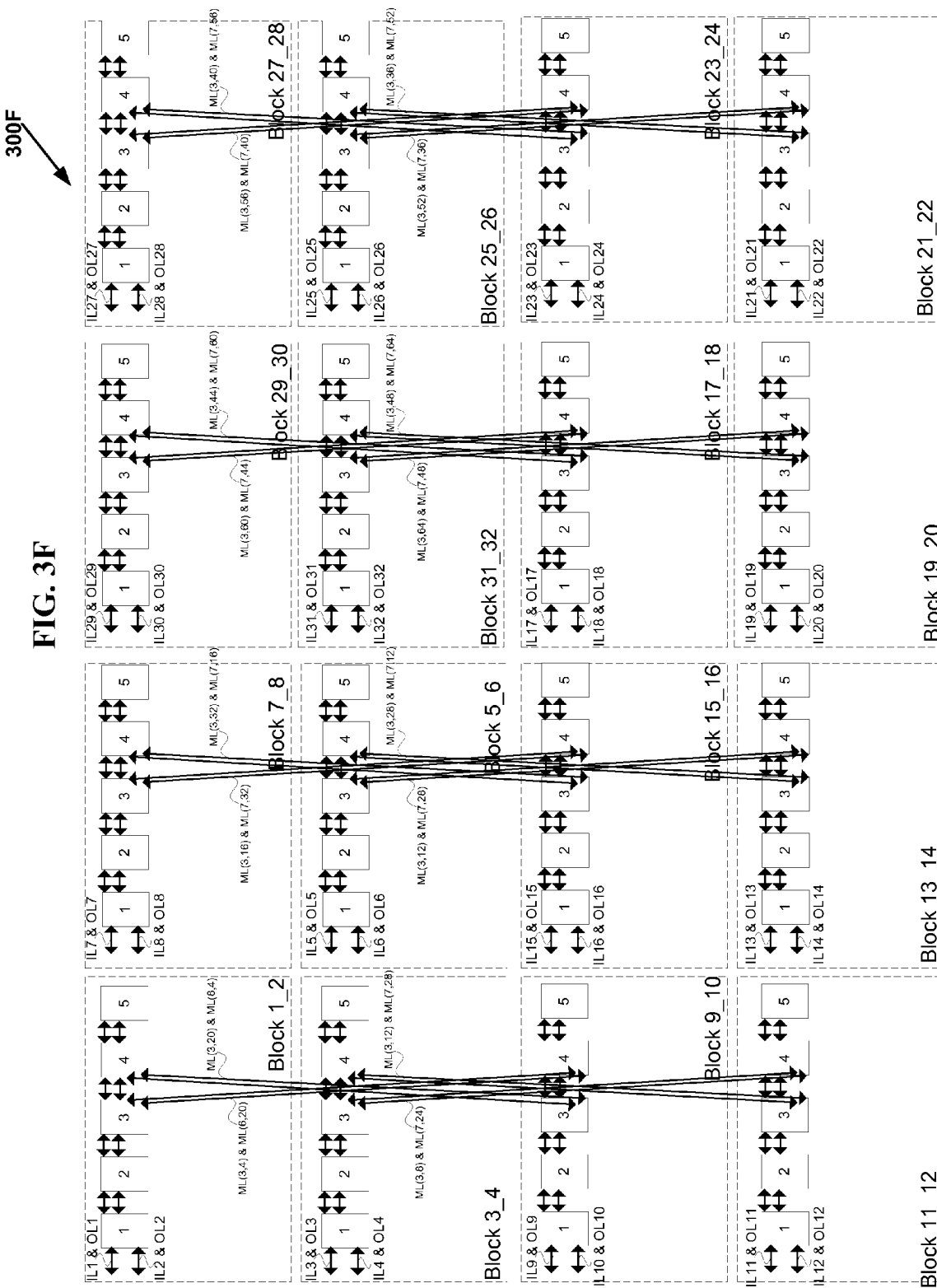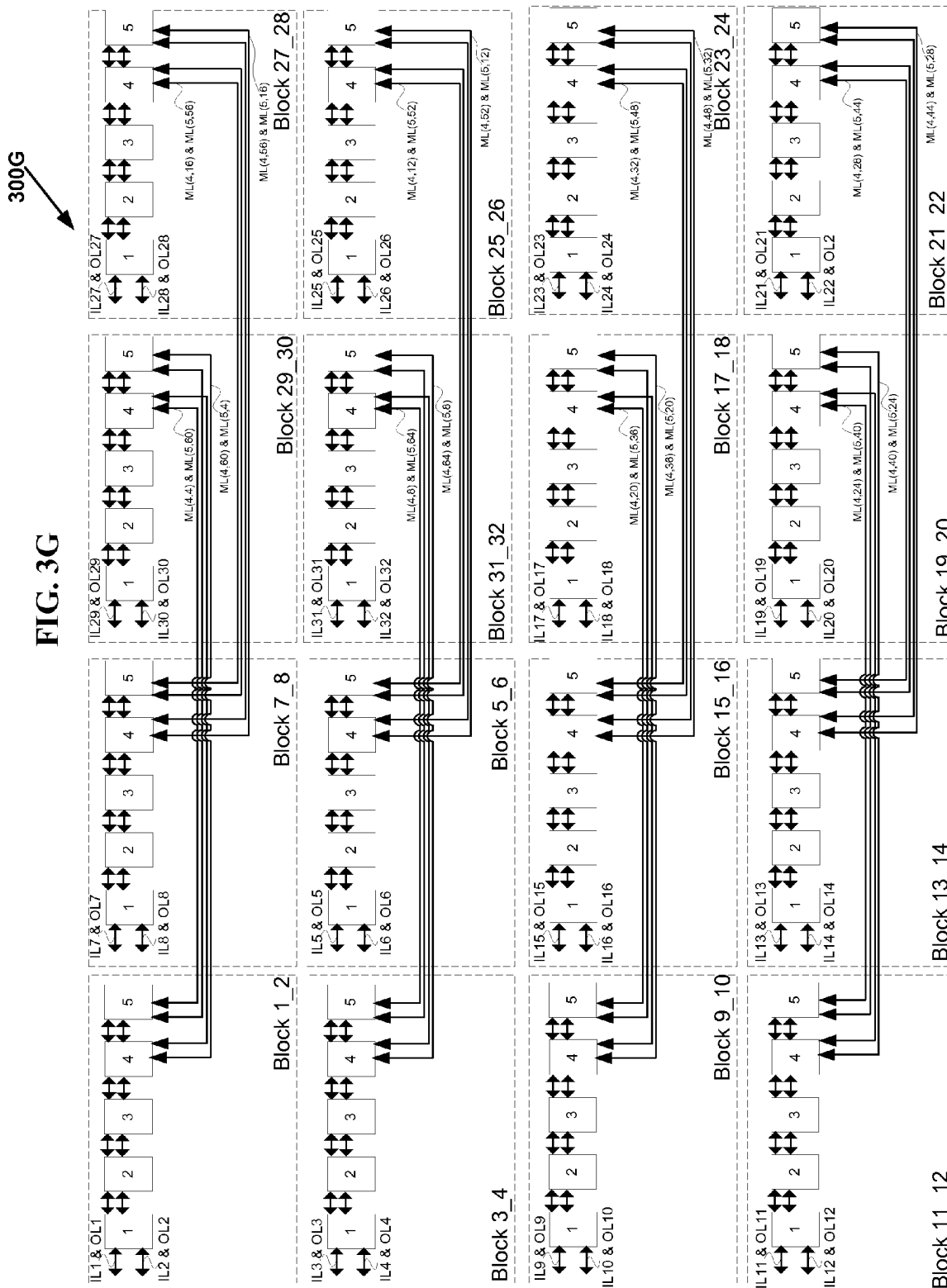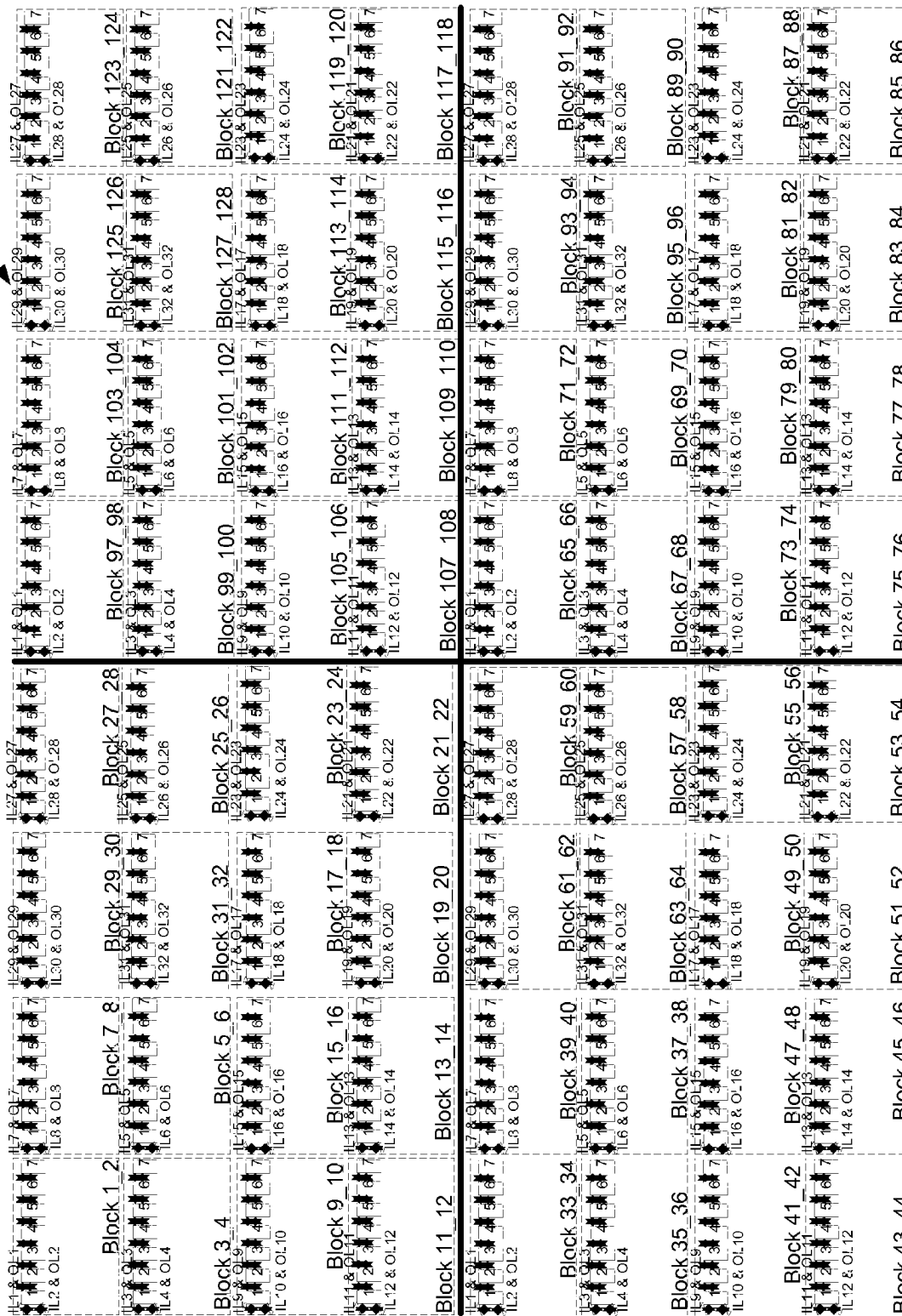
FIG. 5A3
(Prior Art)

Doc Code: REIS.DECL
Document Description: Reissue Declaration Filed In Accordance With MPEP 1414

REISSUE APPLICATION DECLARATION BY THE INVENTOR	Docket Number (Optional) **V-0070US**

I hereby declare that:

Each inventor's residence and mailing address are stated below next to their name.

I believe I am the original inventor or an original joint inventor of the subject matter which is described and claimed in patent number _8,269,523_ , granted _September 18, 2012_ and for which a reissue patent is sought on the invention titled _VLSI Layouts of Fully Connected Generalized Networks_,

_____,

the specification of which

[✔] is attached hereto.

[ ] was filed on _____ as reissue application number _____ .

The above-identified application was made or authorized to be made by me.

I hereby acknowledge that any willful false statement made in this declaration is punishable under 18 U.S.C. 1001 by fine or imprisonment of not more than five (5) years, or both.

I believe the original patent to be wholly or partly inoperative or invalid, for the reasons described below. (Check all boxes that apply.)

[ ] by reason of a defective specification or drawing.

[ ] by reason of the patentee claiming more or less than he had the right to claim in the patent.

[✔] by reason of other errors.

At least one error upon which reissue is based is described below. If the reissue is a broadening reissue, a claim that the application seeks to broaden must be identified:

Application No. 12/601,275 filed November 22, 2009 and issued as Patent No. 8,269,523 on September 18, 2012 was filed within 30 months of the priority date but the basic national stage fee was paid a month later than 30 months of the priority date. Under 37 CFR 1.495(g), with such conflicting instructions, 12/601,275 should have been prosecuted as 111(a) bypass Continuation application. But 12/601,275 was prosecuted as National Stage Entry 371(c) application and US 8,269,523 is granted. Applicant respectfully files for a reissue of patent 8,269,523 to prosecute as 111(a) bypass continuation application.

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

(REISSUE APPLICATION DECLARATION BY THE INVENTOR, page 2)	Docket Number (Optional) **V-0070US**

Note: To appoint a power of attorney, use form PTO/AIA/81.

Correspondence Address: Direct all communications about the application to:

[✔] The address associated with Customer Number: 38139

OR

[ ] Firm or Individual Name	
Address	
City	State	Zip
Country	
Telephone	Email

WARNING:

Petitioner/applicant is cautioned to avoid submitting personal information in documents filed in a patent application that may contribute to identity theft. Personal information such as social security numbers, bank account numbers, or credit card numbers (other than a check or credit card authorization form PTO-2038 submitted for payment purposes) is never required by the USPTO to support a petition or an application. If this type of personal information is included in documents submitted to the USPTO, petitioners/applicants should consider redacting such personal information from the documents before submitting them to the USPTO. Petitioner/applicant is advised that the record of a patent application is available to the public after publication of the application (unless a non-publication request in compliance with 37 CFR 1.213(a) is made in the application) or issuance of a patent. Furthermore, the record from an abandoned application may also be available to the public if the application is referenced in a published application or an issued patent (see 37 CFR 1.14). Checks and credit card authorization forms PTO-2038 submitted for payment purposes are not retained in the application file and therefore are not publicly available.

Legal name of sole or first inventor (*E.g.*, Given Name (first and middle (if any) and Family Name or Surname)

Venkat Konda

Inventor's Signature /Venkat Konda/	Date (Optional) November 27, 2018

Residence: City San Jose	State CA	Country USA

Mailing Address

6278 Grand Oak Way

City San Jose	State CA	Zip 95135	Country USA

[ ] Additional joint inventors are named on the _____ supplemental sheet(s) PTO/AIA/10 attached hereto.

Privacy Act Statement

The **Privacy Act of 1974 (P.L. 93-579)** requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (*i.e.*, GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

Doc Code: REIS.DECL
Document Description: Reissue Declaration Filed In Accordance With MPEP 1414

REISSUE APPLICATION DECLARATION BY THE ASSIGNEE	Docket Number (optional) V-0070US

I hereby declare that:

The residence and mailing address of the inventor or joint inventors are stated below.

Name of the Assignee:

Konda Technologies Inc.

I am authorized to act on behalf of the assignee (if the assignee is a juristic entity).

The entire title to the patent identified below is vested in said assignee, or if there are multiple assignees/owners, all assignees/owners have executed a Reissue Application Declaration to account for the entire title of the patent identified below.

Inventor Venkat Konda

Residence: City San Jose	State CA	Country USA

Mailing Address

6278 Grand Oak Way

City San Jose	State CA	Zip 95135	Country USA

☐   Additional Inventors are named on separately numbered sheets attached hereto.

Patent Number 8,269,523	Patent Issue Date September 18, 2018

I believe said inventor(s) to be the original inventor or original joint inventors of the subject matter which is described and claimed in said patent, for which a reissue patent is sought on the invention titled:

VLSI Layouts of Fully Connected Generalized Networks

the specification of which

☑ is attached hereto.

☐ was filed on _____ as reissue application number _____ .

The above-identified application was made or authorized to be made by me.

I hereby acknowledge that any willful false statement made in this declaration is punishable under 18 U.S.C. 1001 by fine or imprisonment of not more than five (5) years, or both.

I believe the original patent to be wholly or partly inoperative or invalid, for the reasons described below.
(Check all boxes that apply.)

☐   by reason of a defective specification or drawing.

☐   by reason of the patentee claiming more or less than he had the right to claim in the patent.

☑   by reason of other errors.

REISSUE APPLICATION DECLARATION BY THE ASSIGNEE	Docket Number (Optional) V-0070US

At least one error upon which reissue is based is described below. If the reissue is a broadening reissue, a claim that the application seeks to broaden must be identified and the box below must be checked:

Under 37 CFR 1.495(g), with conflicting instructions for filing documents and fees with in 30 month period of priority date, 12/601,275 should be prosecuted as 111(a) bypass Continuation and not as 371(c) application.

[Attach additional sheets, if needed.]

[✔] **The application for the original patent was filed under 37 CFR 1.46 by the assignee of the entire interest.**

I hereby appoint:

[✔] Practitioners associated with Customer Number: 38139

OR

[ ] Practitioner(s) named below:

Name	Registration Number

as my/our attorney(s) or agent(s) to prosecute the application identified above, and to transact all business in the United States Patent and Trademark Office connected therewith.

Correspondence Address: Direct all communications about the application to:

[✔] The address associated with Customer Number: 38139

OR

[ ] Firm or Individual Name	
Address	

City		State		Zip	

Country	

Telephone		Email	

WARNING:

Petitioner/applicant is cautioned to avoid submitting personal information in documents filed in a patent application that may contribute to identity theft. Personal information such as social security numbers, bank account numbers, or credit card numbers (other than a check or credit card authorization form PTO-2038 submitted for payment purposes) is never required by the USPTO to support a petition or an application. If this type of personal information is included in documents submitted to the USPTO, petitioners/applicants should consider redacting such personal information from the documents before submitting them to the USPTO. Petitioner/applicant is advised that the record of a patent application is available to the public after publication of the application (unless a non-publication request in compliance with 37 CFR 1.213(a) is made in the application) or issuance of a patent. Furthermore, the record from an abandoned application may also be available to the public if the application is referenced in a published application or an issued patent (see 37 CFR 1.14). Checks and credit card authorization forms PTO-2038 submitted for payment purposes are not retained in the application file and therefore are not publicly available.

Signature /Venkat Konda/	Date (Optional) 11-27-2018

Legal name of person signing	Venkat Konda

Address of Assignee   6278 Grand Oak Way, San Jose, CA 95135, USA

Privacy Act Statement

The **Privacy Act of 1974 (P.L. 93-579)** requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (*i.e.*, GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

PTO/AIA/14 (11-15)
Approved for use through 04/30/2017. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

| Application Data Sheet 37 CFR 1.76 | Attorney Docket Number | V-0070US |
| | Application Number | |

| Title of Invention | VLSI Layouts of Fully Connected Generalized Networks |

The application data sheet is part of the provisional or nonprovisional application for which it is being submitted. The following form contains the bibliographic data arranged in a format specified by the United States Patent and Trademark Office as outlined in 37 CFR 1.76.
This document may be completed electronically and submitted to the Office in electronic format using the Electronic Filing System (EFS) or the document may be printed and included in a paper filed application.

Secrecy Order 37 CFR 5.2:

☐ Portions or all of the application associated with this Application Data Sheet may fall under a Secrecy Order pursuant to 37 CFR 5.2 (Paper filers only. Applications that fall under Secrecy Order may not be filed electronically.)

Inventor Information:

| Inventor | 1 | | Remove |

Legal Name

Prefix	Given Name	Middle Name	Family Name	Suffix
▾	Venkat		Konda	▾

Residence Information (Select One)   ● US Residency   Non US Residency   Active US Military Service

| City | San Jose | State/Province | CA | Country of Residence | US |

Mailing Address of Inventor:

Address 1	6278 Grand Oak Way
Address 2	
City	San Jose	State/Province	CA
Postal Code	95135	Country i	US

All Inventors Must Be Listed - Additional Inventor Information blocks may be generated within this form by selecting the **Add** button. | Add |

Correspondence Information:

Enter either Customer Number or complete the Correspondence Information section below.
For further information see 37 CFR 1.33(a).

☐ **An Address is being provided for the correspondence Information of this application.**

| Customer Number | 38139 |
| Email Address | venkat@kondatech.com | Add Email | Remove Email |

Application Information:

Title of the Invention	VLSI Layouts of Fully Connected Generalized Networks		
Attorney Docket Number	V-0070US	Small Entity Status Claimed	☒
Application Type	Nonprovisional		▾
Subject Matter	Utility		▾
Total Number of Drawing Sheets (if any)	39	Suggested Figure for Publication (if any)	1H

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	V-0070US
	Application Number	

Title of Invention	VLSI Layouts of Fully Connected Generalized Networks

Filing By Reference:

Only complete this section when filing an application by reference under 35 U.S.C. 111(c) and 37 CFR 1.57(a). Do not complete this section if application papers including a specification and any drawings are being filed. Any domestic benefit or foreign priority information must be provided in the appropriate section(s) below (i.e., "Domestic Benefit/National Stage Information" and "Foreign Priority Information").

For the purposes of a filing date under 37 CFR 1.53(b), the description and any drawings of the present application are replaced by this reference to the previously filed application, subject to conditions and requirements of 37 CFR 1.57(a).

Application number of the previously filed application	Filing date (YYYY-MM-DD)	Intellectual Property Authority or Country

Publication Information:

☐	Request Early Publication (Fee required at time of Request 37 CFR 1.219)
☐	**Request Not to Publish.** I hereby request that the attached application not be published under 35 U.S.C. 122(b) and certify that the invention disclosed in the attached application **has not and will not** be the subject of an application filed in another country, or under a multilateral international agreement, that requires publication at eighteen months after filing.

Representative Information:

Representative information should be provided for all practitioners having a power of attorney in the application. Providing this information in the Application Data Sheet does not constitute a power of attorney in the application (see 37 CFR 1.32).
Either enter Customer Number or complete the Representative Name section below. If both sections are completed the customer Number will be used for the Representative Information during processing.

Please Select One:	● Customer Number	US Patent Practitioner	○ Limited Recognition (37 CFR 11.9)
Customer Number	38139		

Domestic Benefit/National Stage Information:

This section allows for the applicant to either claim benefit under 35 U.S.C. 119(e), 120, 121, 365(c), or 386(c) or indicate National Stage entry from a PCT application. Providing benefit claim information in the Application Data Sheet constitutes the specific reference required by 35 U.S.C. 119(e) or 120, and 37 CFR 1.78.
When referring to the current application, please leave the "Application Number" field blank.

Prior Application Status	Patented ▼			Remove	
Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)	Patent Number	Issue Date (YYYY-MM-DD)
	reissued of ▼	12/601275	2008-05-22	8269523	2012-09-18

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	V-0070US
	Application Number	

Title of Invention	VLSI Layouts of Fully Connected Generalized Networks

Prior Application Status	Expired ▾		Remove
Application Number	**Continuity Type**	**Prior Application Number**	**Filing or 371(c) Date (YYYY-MM-DD)**
12/601275	Continuation of ▾	PCT/US08/64605	2007-05-22
Prior Application Status	Expired ▾		Remove
Application Number	**Continuity Type**	**Prior Application Number**	**Filing or 371(c) Date (YYYY-MM-DD)**
PCT/US08/64605	Claims benefit of provisional ▾	60/940394	2007-05-25

Additional Domestic Benefit/National Stage Data may be generated within this form by selecting the **Add** button.

| Add |

Foreign Priority Information:

This section allows for the applicant to claim priority to a foreign application. Providing this information in the application data sheet constitutes the claim for priority as required by 35 U.S.C. 119(b) and 37 CFR 1.55. When priority is claimed to a foreign application that is eligible for retrieval under the priority document exchange program (PDX)[i] the information will be used by the Office to automatically attempt retrieval pursuant to 37 CFR 1.55(i)(1) and (2). Under the PDX program, applicant bears the ultimate responsibility for ensuring that a copy of the foreign application is received by the Office from the participating foreign intellectual property office, or a certified copy of the foreign priority application is filed, within the time period specified in 37 CFR 1.55(g)(1).

			Remove
Application Number	**Country[i]**	**Filing Date (YYYY-MM-DD)**	**Access Code[i] (if applicable)**

Additional Foreign Priority Data may be generated within this form by selecting the **Add** button.

| Add |

Statement under 37 CFR 1.55 or 1.78 for AIA (First Inventor to File) Transition Applications

☐	This application (1) claims priority to or the benefit of an application filed before March 16, 2013 and (2) also contains, or contained at any time, a claim to a claimed invention that has an effective filing date on or after March 16, 2013.
	NOTE: By providing this statement under 37 CFR 1.55 or 1.78, this application, with a filing date on or after March 16, 2013, will be examined under the first inventor to file provisions of the AIA.

VENKAT KONDA EXHIBIT 2005

| **Application Data Sheet 37 CFR 1.76** | Attorney Docket Number | V-0070US |
| | Application Number | |

| Title of Invention | VLSI Layouts of Fully Connected Generalized Networks |

Authorization or Opt-Out of Authorization to Permit Access:

When this Application Data Sheet is properly signed and filed with the application, applicant has provided written authority to permit a participating foreign intellectual property (IP) office access to the instant application-as-filed (see paragraph A in subsection 1 below) and the European Patent Office (EPO) access to any search results from the instant application (see paragraph B in subsection 1 below).

Should applicant choose not to provide an authorization identified in subsection 1 below, applicant **must opt-out** of the authorization by checking the corresponding box A or B or both in subsection 2 below.

NOTE: This section of the Application Data Sheet is **ONLY** reviewed and processed with the **INITIAL** filing of an application. After the initial filing of an application, an Application Data Sheet cannot be used to provide or rescind authorization for access by a foreign IP office(s). Instead, Form PTO/SB/39 or PTO/SB/69 must be used as appropriate.

1. Authorization to Permit Access by a Foreign Intellectual Property Office(s)

A. Priority Document Exchange (PDX) - Unless box A in subsection 2 (opt-out of authorization) is checked, the undersigned hereby **grants the USPTO authority** to provide the European Patent Office (EPO), the Japan Patent Office (JPO), the Korean Intellectual Property Office (KIPO), the State Intellectual Property Office of the People's Republic of China (SIPO), the World Intellectual Property Organization (WIPO), and any other foreign intellectual property office participating with the USPTO in a bilateral or multilateral priority document exchange agreement in which a foreign application claiming priority to the instant patent application is filed, access to: (1) the instant patent application-as-filed and its related bibliographic data, (2) any foreign or domestic application to which priority or benefit is claimed by the instant application and its related bibliographic data, and (3) the date of filing of this Authorization. See 37 CFR 1.14(h)(1).

B. Search Results from U.S. Application to EPO - Unless box B in subsection 2 (opt-out of authorization) is checked, the undersigned hereby **grants the USPTO authority** to provide the EPO access to the bibliographic data and search results from the instant patent application when a European patent application claiming priority to the instant patent application is filed. See 37 CFR 1.14(h)(2).

The applicant is reminded that the EPO's Rule 141(1) EPC (European Patent Convention) requires applicants to submit a copy of search results from the instant application without delay in a European patent application that claims priority to the instant application.

2. Opt-Out of Authorizations to Permit Access by a Foreign Intellectual Property Office(s)

☐ A. Applicant **DOES NOT** authorize the USPTO to permit a participating foreign IP office access to the instant application-as-filed. If this box is checked, the USPTO will not be providing a participating foreign IP office with any documents and information identified in subsection 1A above.

☐ B. Applicant **DOES NOT** authorize the USPTO to transmit to the EPO any search results from the instant patent application. If this box is checked, the USPTO will not be providing the EPO with search results from the instant application.

NOTE: Once the application has published or is otherwise publicly available, the USPTO may provide access to the application in accordance with 37 CFR 1.14.

VENKAT KONDA EXHIBIT 2005

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	V-0070US
	Application Number	

Title of Invention	VLSI Layouts of Fully Connected Generalized Networks

Applicant Information:

Providing assignment information in this section does not substitute for compliance with any requirement of part 3 of Title 37 of CFR to have an assignment recorded by the Office.

Applicant	1	Remove

If the applicant is the inventor (or the remaining joint inventor or inventors under 37 CFR 1.45), this section should not be completed. The information to be provided in this section is the name and address of the legal representative who is the applicant under 37 CFR 1.43; or the name and address of the assignee, person to whom the inventor is under an obligation to assign the invention, or person who otherwise shows sufficient proprietary interest in the matter who is the applicant under 37 CFR 1.46. If the applicant is an applicant under 37 CFR 1.46 (assignee, person to whom the inventor is obligated to assign, or person who otherwise shows sufficient proprietary interest) together with one or more joint inventors, then the joint inventor or inventors who are also the applicant should be identified in this section.

Clear

● Assignee	Legal Representative under 35 U.S.C. 117	Joint Inventor
Person to whom the inventor is obligated to assign.		Person who shows sufficient proprietary interest

If applicant is the legal representative, indicate the authority to file the patent application, the inventor is:

▼

Name of the Deceased or Legally Incapacitated Inventor:	

If the Applicant is an Organization check here.	☒

Organization Name	Konda Technologies Inc.

Mailing Address Information For Applicant:

Address 1	6278 Grand Oak Way		
Address 2			
City	San Jose	**State/Province**	CA
Country	US	Postal Code	95135
Phone Number	408-472-3273	Fax Number	408-238-2478
Email Address	venkat@kondatech.com		

Additional Applicant Data may be generated within this form by selecting the Add button.	Add

Assignee Information including Non-Applicant Assignee Information:

Providing assignment information in this section does not substitute for compliance with any requirement of part 3 of Title 37 of CFR to have an assignment recorded by the Office.

PTO/AIA/14 (11-15)
Approved for use through 04/30/2017. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

| **Application Data Sheet 37 CFR 1.76** | Attorney Docket Number | V-0070US |
| | Application Number | |

| Title of Invention | VLSI Layouts of Fully Connected Generalized Networks |

Assignee 1

Complete this section if assignee information, including non-applicant assignee information, is desired to be included on the patent application publication. An assignee-applicant identified in the "Applicant Information" section will appear on the patent application publication as an applicant. For an assignee-applicant, complete this section only if identification as an assignee is also desired on the patent application publication.

| | Remove |

If the Assignee or Non-Applicant Assignee is an Organization check here. ☐

| Prefix | **Given Name** | Middle Name | **Family Name** | Suffix |
| | | | | |

Mailing Address Information For Assignee including Non-Applicant Assignee:

Address 1	
Address 2	
City		**State/Province**	
Country i		Postal Code	
Phone Number		Fax Number	
Email Address	

Additional Assignee or Non-Applicant Assignee Data may be generated within this form by selecting the Add button. | Add |

Signature: | Remove |

NOTE: This Application Data Sheet must be signed in accordance with 37 CFR 1.33(b). **However, if this Application Data Sheet is submitted with the <u>INITIAL</u> filing of the application <u>and</u> either box A or B is <u>not</u> checked in subsection 2 of the "Authorization or Opt-Out of Authorization to Permit Access" section, then this form must also be signed in accordance with 37 CFR 1.14(c).**

This Application Data Sheet <u>**must**</u> be signed by a patent practitioner if one or more of the applicants is a **juristic entity** (e.g., corporation or association). If the applicant is two or more joint inventors, this form must be signed by a patent practitioner, <u>**all**</u> joint inventors who are the applicant, or one or more joint inventor-applicants who have been given power of attorney (e.g., see USPTO Form PTO/AIA/81) on behalf of <u>**all**</u> joint inventor-applicants.

See 37 CFR 1.4(d) for the manner of making signatures and certifications.

| **Signature** | /Venkat Konda/ | | | Date (YYYY-MM-DD) | 2018-11-27 |
| First Name | Venkat | Last Name | Konda | Registration Number | |

Additional Signature may be generated within this form by selecting the Add button. | Add |

EFS Web 2.2.12

| **Application Data Sheet 37 CFR 1.76** | Attorney Docket Number | V-0070US |
| | Application Number | |

| Title of Invention | VLSI Layouts of Fully Connected Generalized Networks |

This collection of information is required by 37 CFR 1.76. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 23 minutes to complete, including gathering, preparing, and submitting the completed application data sheet form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

Privacy Act Statement

The Privacy Act of 1974 (P.L. 93-579) requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C. 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether the Freedom of Information Act requires disclosure of these records.

2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.

3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.

4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).

5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent CooperationTreaty.

6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).

7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (i.e., GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.

8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspections or an issued patent.

9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

Doc Code: PET.OP
Document Description: Petition for Review by the Office of ~~Petitions~~ *PCT LEGAL*

Attn:- ERIN THOMPSON

PTO/SB/44S (05-18)
Approved for use through 11/30/2020. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PETITION TO ACCEPT AN UNINTENTIONALLY DELAYED CLAIM UNDER 35 U.S.C. 119(e)
FOR THE BENEFIT OF A PRIOR-FILED PROVISONAL APPLICATION (37 CFR 1.78(c))
AND FOR BENEFIT OF A PRIOR FILED INTERNATIONAL APPLICATION (37CFR 1.78(e))

Page 1 of 4

First named inventor:	Venkat Konda	
Application No.: ~~PCT~~	*(6/01)*	Art Unit: *UNASSIGNED*
Filed: November 24, 2018		Examiner: *UNASSIGNED*
Title:	VLSI LAYOUTS OF FULLY CONNECTED GENERALIZED NETWORKS	

Attention: Office of ~~Petitions~~ *PCT LEGAL*
Mall Stop ~~Petition~~ *PCT*
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

✓ *PROVISIONAL APPLICATION: 60/940,394*
PCT APPLICATION: PCT/US08/64605
REISSUE APP:- none

PCT LEGAL

NOTE: If information or assistance is needed in completing this form, please contact the Office of ~~Petitions~~

APPLICANT HEREBY PETITIONS FOR ACCEPTANCE OF AN UNINTENTIONALLY DELAYED BENEFIT CLAIM UNDER 37 CFR 1.78(c).
AND 37CFR 1.78(e)

NOTE: A grantable petition requires the following items:
(1) the reference required by 35 U.S.C. 119(e) and 37 CFR 1.78(a)(3) to the prior-filed provisional application, unless previously submitted;
(2) the petition fee set forth in 37 CFR 1.17(m); and
(3) a statement that the entire delay between the date the benefit claim was due under 37 CFR 1.78(a)(4) and the date the benefit claim was filed was unintentional. The Director may require additional information where there is a question as to whether the delay was unintentional.

1. The reference required by 35 U.S.C. 119(e) and 37 CFR 1.78(a)(3) to the prior-filed provisional application

[■] The application was filed on or after September 16, 2012, and the reference is either set forth in the attached corrected ADS, or in a corrected ADS that was previously submitted.

[ ] The application was filed prior to September 16, 2012, and the reference is either (1) set forth in a supplemental ADS, or (2) set forth in the attached amendment to the first sentence(s) of the specification, or (3) a supplemental ADS or amendment to the first sentence(s) of the specification was previously submitted. See 37 CFR 1.78(h).

Reminders:
- Any ADS which corrects or updates the information of record must comply with 37 CFR 1.76(c)(2) (or for applications filed prior to September 16, 2012, pre-AIA 37 CFR 1.76(c))
- The nonprovisional application must have been filed within 1 year of the filing date of the prior-filed provisional application except as provided in 37 CFR 1.78(b). See 37 CFR 1.78(a).

2. Petition fee

[■] Small entity fee $ 1000.00 (37 CFR 1.17(m)). Applicant asserts small entity status. See 37 CFR 1.27.

[ ] Micro entity fee $ _____ (37 CFR 1.17(m)). Applicant has established or is establishing micro entity status. See 37 CFR 1.29. Form PTO/SB/15A or B or equivalent must either be enclosed or have been submitted previously.

[ ] Undiscounted fee $ _____ (37 CFR 1.17(m)).

This collection of information is required by 37 CFR 1.78(c). The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11, 1.14 and 41.6. This collection is estimated to take 1 hour to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Mail Stop Petition, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

APPLICATION NO. 12/601,275 FILED NOVEMBER 22, 2009 and ISSUED AS PATENT NO. 8,269,523 ON SEPTEMBER 18, 2012 WAS FILED WITHIN 30 MONTHS OF THE PRIORITY DATE BUT THE BASIC NATIONAL STAGE FEE WAS PAID A MONTH LATER THAN 30 MONTHS PRIORITY DATE. UNDER 37 CFR 1.495(G) WITH CONFLICTING INSTRUCTIONS, 12/601,275 SHOULD HAVE BEEN AS 111(a) BYPASS CONTINUATION APPLICATION. BUT IT WAS PROSECUTED AS 371(c) APPLICATION. APPLICANT FILES FOR REISSUE AS 111(a) APPLICATION. PLEASE GRANT PETITION.

Doc Code: PET.OP
Document Description: Petition for Review by the Office of Petitions

PETITION TO ACCEPT AN UNINTENTIONALLY DELAYED CLAIM UNDER 35 U.S.C. 119(e)
FOR THE BENEFIT OF A PRIOR-FILED PROVISIONAL APPLICATION (37 CFR 1.78(c))
Page 2 of 4

3. STATEMENT: The entire delay between the date the benefit claim was due under 37 CFR 1.78(a)(4) and the date the benefit claim was filed was unintentional.

NOTE: The Director may require additional information where there is a question whether the delay was unintentional.

WARNING:

Petitioner/applicant is cautioned to avoid submitting personal information in documents filed in a patent application that may contribute to identity theft. Personal information such as social security numbers, bank account numbers, or credit card numbers (other than a check or credit card authorization form PTO-2038 submitted for payment purposes) is never required by the USPTO to support a petition or an application. If this type of personal information is included in documents submitted to the USPTO, petitioners/applicants should consider redacting such personal information from the documents before submitting them to the USPTO. Petitioner/applicant is advised that the record of a patent application is available to the public after publication of the application (unless a non-publication request in compliance with 37 CFR 1.213(a) is made in the application) or issuance of a patent. Furthermore, the record from an abandoned application may also be available to the public if the application is referenced in a published application or an issued patent (see 37 CFR 1.14). Checks and credit card authorization forms PTO-2038 submitted for payment purposes are not retained in the application file and therefore are not publicly available.

/Venkat Konda/	**11-27-2018**
Signature*	Date
Venkat Konda	
Typed or Printed Name	Registration Number, if applicable
6278 Grand Oak Way	**408-472-3273**
Address	Telephone Number
San Jose, CA 95135	
Address	

* This petition must be signed in accordance with 37 CFR 1.33. Please see 37 CFR 1.4(d) for the signature requirements. Submit multiple forms if more than one signature is required.

Enclosures:

[■] Application Data Sheet (see instructions starting on page 3)

[■] Fee

[ ] Other: _____

CERTIFICATE OF MAILING OR TRANSMISSION [37 CFR 1.8(a)]

I hereby certify that this correspondence is being:

[ ] Deposited with the United States Postal Service on the date shown below with sufficient postage as first class mail in an envelope addressed to: Mail Stop Petition, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450.

[■] Transmitted by EFS-Web or by fax on the date shown below to the United States Patent and Trademark Office at (571) 273-8300.

11-24-2018	**/Venkat Konda/**
Date	Signature
	Venkat Konda
	Typed or printed name of person signing certificate

Instructions for Petition to Accept an Unintentionally Delayed Claim under 35 U.S.C. 119(e) for the Benefit of a Prior-Filed Provisional Application (37 CFR 1.78(c))

(Not to be Submitted to the USPTO)

Note: This form is for unintentionally delayed benefit claims to provisional application(s) only

1. The reference required by 35 U.S.C. 119(e) and 37 CFR 1.78(a)(3) to the provisional application

   a. For applications filed on or after September 16, 2012, the reference must be set forth in a corrected ADS.

Note: Any ADS filed after the filing of the application is considered a corrected (or updated) ADS even if an ADS was not previously submitted. A corrected ADS must identify the information that is being changed with underlining for insertions and strike-through or brackets for text removed. In general, the identification of the information being changed should be made relative to the most recent filing receipt. For example, where the most recent filing receipt for the application shows no benefit claim, the entire benefit claim must be shown with underlining in the corrected ADS. In addition, if the ADS identified an incorrect provisional application number and the most recent filing receipt included the incorrect provisional application number, the corrected ADS should identify the incorrect provisional application number being deleted with strike-through or brackets, and should identify the correct provisional application number being added with underlining. For more information regarding a corrected ADS in an application filed on or after September 16, 2012, see MPEP 601.05(a), subsection II.

A corrected ADS may include all of the section headings listed in 37 CFR 1.76(b) with all appropriate data for each heading or only those sections (including the section headings) containing changed or updated information. See 37 CFR 1.76(c)(2). A corrected ADS must identify the application by application number and be properly signed.

Use of the corrected Web-based ADS is recommended for registered users of EFS-Web because it will pre-populate with information of record, applicants can then type in the desired changes, and the system will create a PDF version with the appropriate strike-through and underlining. For more information, see the "Quick Start Guide for Corrected Web-based Application Data Sheet (Corrected Web ADS)" available at https://www.uspto.gov/patents-application-process/applying-online/efs-web-guidance-and-resources.

Applicants may also use Form PTO/AIA/14 which may be printed and marked up to comply with 37 CFR 1.76(c).

[Page 3 of 4]

Instructions for Petition to Accept an Unintentionally Delayed Claim under 35 U.S.C. 119(e) for the Benefit of a Prior-Filed Provisional Application (37 CFR 1.78(c))

(Not to be Submitted to the USPTO)

b. For applications filed before September 16, 2012, the reference to the prior-filed application may be made in a supplemental ADS in compliance with pre-AIA 37 CFR 1.76(c) or in an amendment in the first sentence(s) of the specification following the title. See 37 CFR 1.78(h).

Note: For applications filed before September 16, 2012, any ADS submitted after the filing date of the application is a supplemental ADS, regardless of whether an original ADS was submitted with the application papers on filing. Supplemental ADS papers must be labeled Supplemental ADS or Supplemental Application Data Sheet, include each of the seven section headings listed in pre-AIA 37 CFR 1.76(b) with all appropriate data for the section heading, and identify the information that is being changed. See pre-AIA 37 CFR 1.76(c). A supplemental ADS must also identify the application by application number and be properly signed. For more information regarding a supplemental ADS in an application filed before September 16, 2012, see MPEP 601.05(b), subsection II.

2. Petition fee

The petition fee is set forth in 37 CFR 1.17(m) and must be included with the petition. Petitioner is advised to refer to the current fee schedule at https://www.uspto.gov/learning-and-resources/fees-and-payment/uspto-fee-schedule.

3. Statement

37 CFR 1.78(c) requires a statement that the entire delay between the date the benefit claim was due under 37 CFR 1.78(a)(4) and the date the benefit claim was filed was unintentional. The required statement is included in this form.

Note: the Director may require additional information where there is a question whether the delay was unintentional.

[Page 4 of 4]

Privacy Act Statement

The **Privacy Act of 1974 (P.L. 93-579)** requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (*i.e.*, GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

Electronic Patent Application Fee Transmittal

Application Number:	16202067
Filing Date:	
Title of Invention:	VLSI Layouts of Fully Connected Generalized Networks
First Named Inventor/Applicant Name:	Venkat Konda
Filer:	Venkat Konda
Attorney Docket Number:	V-0070US

Filed as Small Entity

Filing Fees for   Utility under 35 USC 111(a)

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
PET. DELAY SUB OR RESTORE PRIORITY-CLAIM	2454	1	1000	1000
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Extension-of-Time:				
Miscellaneous:				
			Total in USD ($)	**1000**

Electronic Acknowledgement Receipt

EFS ID:	34414584
Application Number:	16202067
International Application Number:	
Confirmation Number:	8134
Title of Invention:	VLSI Layouts of Fully Connected Generalized Networks
First Named Inventor/Applicant Name:	Venkat Konda
Customer Number:	38139
Filer:	Venkat Konda
Filer Authorized By:	
Attorney Docket Number:	V-0070US
Receipt Date:	27-NOV-2018
Filing Date:	
Time Stamp:	22:56:04
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	yes
Payment Type	CARD
Payment was successfully received in RAM	$ 1000
RAM confirmation Number	112818INTEFSW22591500
Deposit Account	
Authorized User	
The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows:	

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Petition for review by the PCT legal office	V-0070US-sb0445-scan.pdf	3197482 d61d11d681971ce4c20ebed1078c3ac3169ce343	no	5

Warnings:

Information:

| 2 | Fee Worksheet (SB06) | fee-info.pdf | 30061

cc59e23fd8e6a860bffb41a40ce63b8c888fb125 | no | 2 |

Warnings:

Information:

| | Total Files Size (in bytes): | | 3227543 | | |

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.
National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.
New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

Reissue of US Patent No. 8,269,523   Attorney Docket No. V-0070US
Reissue Application Filed: 11/27/18

In The United States Patent And Trademark Office

Reissue Application for US Patent No: 8,269,523   Group Art Unit: Unassigned

Issued: September 18, 2012   Examiner: Unassigned

Applicant(s): Venkat Konda   Filed: November 27, 2018

5   Title: VLSI Layouts of Fully Connected Generalized Networks

San Jose, 2018 Nov 27, Tue

AMENDMENT ACCOMPANYING REISSUE APPLICATION
SUBMITTED PURSUANT TO 37 C.F.R. §1.173(B)

10

Mail Stop Reissue

Commissioner for Patents

P.O. Box 1450

Alexandria, VA 22313-1450

15

Dear Sir/Madam:

This Amendment accompanies the reissue patent application filed herewith, and sets forth
applicant's requested amendments to U.S. Patent No. 8,269,523.

20   Amendment to Specification is set forth on page 2.
Amendments to the claims are set forth in the *Listing of the Claims,* which begins on
Page 3. No new claims are added. Originally issued claims 5 – 14, 17, 22, 23, 32 – 46 are
now cancelled. Therefore, originally issued claims 1 - 4, 15, 16, 18 – 21, 24 – 31, 47, 48
are pending in this reissue application following entry of this amendment.

25   *Remarks* are set forth on page 15

Reissue of US Patent No. 8,269,523   Attorney Docket No. V-0070US
Reissue Application Filed: 11/27/18

Amendments to the Specification

Amendments to the specification are indicated below wherein matter to added is indicated
by <u>underlining</u> and matter to be deleted is indicated in [brackets].

5

Please replace the paragraph at column 1, lines 7-15, following the heading
"CROSS REFERENCE TO RELATED APPLICATIONS" with the following:

This application [is related to] <u>is an application for reissue of U.S. Patent No.</u>
10   <u>8,269,523 and this application is bypass continuation</u> and claims priority of the PCT
Application Serial No. PCT/US08/64605 entitled "VLSI LAYOUTS OF FULLY
CONNECTED GENERALIZED NETWORKS" by Venkat Konda assigned to the same
assignee as the current application, filed May 22, 2008, [and] <u>which in turn claims</u>
<u>priority of</u> the U.S. Provisional Patent Application Serial No. 60/940, 394 entitled "VLSI
15   LAYOUTS OF FULLY CONNECTED GENERALIZED NETWORKS" by Venkat
Konda assigned to the same assignee as the current application, filed May 25, 2007.

Reissue of US Patent No. 8,269,523   Attorney Docket No. V-0070US
Reissue Application Filed: 11/27/18

LISTING OF THE CLAIMS

This Claim listing sets forth all the claims that will be pending in this reissue application following entry of the present amendment.

5   1.   (Previously issued; now pending): An integrated circuit device comprising a plurality of sub-integrated circuit blocks and a routing network, and

Said each plurality of sub-integrated circuit blocks comprising a plurality of inlet links and a plurality of outlet links; and

Said routing network comprising of a plurality of stages y, in each said sub-

10   integrated circuit block, starting from the lowest stage of 1 to the highest stage of y, where $y \geq 1$; and

Said routing network comprising a plurality of switches of size $d \times d$, where $d \geq 2$, in each said stage and each said switch of size $d \times d$ having d inlet links and d outlet links; and

15   Said plurality of outlet links of said each sub-integrated circuit block are directly connected to said inlet links of said switches of its corresponding said lowest stage of 1, and said plurality of inlet links of said each sub-integrated circuit block are directly connected from said outlet links of said switches of its corresponding said lowest stage of 1; and

20   Said each sub-integrated circuit block comprising a plurality of forward connecting links connecting from switches in a lower stage to switches in its immediate succeeding higher stage, and also comprising a plurality of backward connecting links connecting from switches in a higher stage to switches in its immediate preceding lower stage; and

25   Said each sub-integrated circuit block comprising a plurality straight links in said forward connecting links from switches in said each lower stage to switches in its immediate succeeding higher stage and a plurality cross links in said forward connecting

links from switches in said each lower stage to switches in its immediate succeeding

higher stage, and further comprising a plurality of straight links in said backward

connecting links from switches in said each higher stage to switches in its immediate

preceding lower stage and a plurality of cross links in said backward connecting links

5   from switches in said each higher stage to switches in its immediate preceding lower

stage,

said plurality of sub-integrated circuit blocks arranged in a two-dimensional grid

of rows and columns, and

said all straight links are connecting from switches in each said sub-integrated

10   circuit block are connecting to switches in the same said sub-integrated circuit block; and

said all cross links are connecting as either vertical or horizontal links between switches

in two different said sub-integrated circuit blocks which are either placed vertically above

or below, or placed horizontally to the left or to the right,

each said plurality of sub-integrated circuit blocks comprising same number of

15   said stages and said switches in each said stage, regardless of the size of said two-

dimensional grid so that each said plurality of sub-integrated circuit block with its

corresponding said stages and said switches in each stage is replicable in both vertical

direction or horizontal direction of said two-dimensional grid.

20   2.   (Previously issued; now pending): The integrated circuit device of claim

1,

said two-dimensional grid of said sub-integrated circuit blocks with their

corresponding said stages and said switches in each stage is scalable by any power of 2,

and,

25   for each multiplication of 2 of the size of total said sub-integrated circuit blocks,

by adding one more stage of switches and the layout is placed in hypercube format and

also the cross links between said one more stage of switches are connected in hypercube

format.

3.   (Previously issued; now pending): The integrated circuit device of claim 2, wherein said cross links in succeeding stages are connecting as alternative vertical and horizontal links between switches in said sub-integrated circuit blocks.

4.   (Previously issued; now pending): The integrated circuit device of claim 3, wherein said cross links from switches in a stage in one of said sub-integrated circuit blocks are connecting to switches in the succeeding stage in another of said sub-integrated circuit blocks so that said cross links are either vertical links or horizontal and vice versa, and hereinafter such cross links are "shuffle exchange links").

5.   (Previously issued; now cancelled): The integrated circuit device of claim 4, wherein said all horizontal shuffle exchange links between switches in any two corresponding said succeeding stages are substantially of equal length and said vertical shuffle exchange links between switches in any two corresponding said succeeding stages are substantially of equal length in the entire said integrated circuit device.

6.   (Previously issued; now cancelled): The integrated circuit device of claim 5, wherein the shortest horizontal shuffle exchange links are connecting at the lowest stage and between switches in two nearest neighboring said sub-integrated circuit blocks, and length of the horizontal shuffle exchange links is doubled in each succeeding stage; and the shortest vertical shuffle exchange links are connecting at the lowest stage and between switches in two nearest neighboring said sub-integrated circuit blocks, and length of the vertical shuffle exchange links is doubled in each succeeding stage.

7.   (Previously issued; now cancelled): The integrated circuit device of claim 6, wherein $y \geq (\log_2 N)$, where $N > 1$, so that the length of the horizontal shuffle exchange links in the highest stage is equal to half the size of the horizontal size of said two dimensional grid of sub-integrated circuit blocks and the length of the vertical shuffle exchange links in the highest stage is equal to half the size of the vertical size of said two dimensional grid of sub-integrated circuit blocks.

8.   (Previously issued; now cancelled): The integrated circuit device of claim 7, wherein $d = 2$ and there is only one switch in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there is only one switch in each said stage in each said sub-integrated circuit block connecting said

5   backward connecting links and said routing network is rearrangeably nonblocking for unicast Benes network with full bandwidth.

9.   (Previously issued; now cancelled): The integrated circuit device of claim 7, wherein $d = 2$ and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least

10   two switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is strictly nonblocking for unicast Benes network and rearrangeably nonblocking for arbitrary fan-out multicast Benes network with full bandwidth.

10.   (Previously issued; now cancelled): The integrated circuit device of claim

15   7, wherein $d = 2$ and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is strictly nonblocking for arbitrary fan-out multicast Benes network with full bandwidth.

20   11.   (Previously issued; now cancelled): The integrated circuit device of claim 6, wherein $y \geq (\log_2 N)$, where $N > 1$, so that the length of the horizontal shuffle exchange links in the highest stage is equal to half the size of the horizontal size of said two dimensional grid of sub-integrated circuit blocks and the length of the vertical shuffle exchange links in the highest stage is equal to half the size of the vertical size of said two

25   dimensional grid of sub-integrated circuit blocks, and
        said each sub-integrated circuit block further comprising a plurality of U-turn links within switches in each of said stages in each of said sub-integrated circuit blocks.

12.   (Previously issued; now cancelled): The integrated circuit device of claim 11, wherein $d = 2$ and there is only one switch in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there is only one switch in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is rearrangeably nonblocking for unicast butterfly fat tree network with full bandwidth.

13.   (Previously issued; now cancelled): The integrated circuit device of claim 11, wherein $d = 2$ and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is strictly nonblocking for unicast butterfly fat tree network and rearrangeably nonblocking for arbitrary fan-out multicast butterfly fat tree network with full bandwidth.

14.   (Previously issued; now cancelled): The integrated circuit device of claim 11, wherein $d = 2$ and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is strictly nonblocking for arbitrary fan-out multicast butterfly fat tree network with full bandwidth.

15.   (Previously issued; now pending): The integrated circuit device of claim 1, wherein said horizontal and vertical links are implemented on two or more metal layers.

16.   (Previously issued; now pending): The integrated circuit device of claim 1, wherein said switches comprising active and reprogrammable cross points and said each cross point is programmable by an SRAM cell or a Flash Cell.

17.   (Previously issued; now cancelled): The integrated circuit device of claim 1, wherein said sub-integrated circuit blocks are of equal die size.

18.   (Previously issued; now pending): The integrated circuit device of claim 15, wherein said sub-integrated circuit blocks are Lookup Tables (hereinafter "LUTs") and said integrated circuit device is a field programmable gate array (FPGA) device or field programmable gate array (FPGA) block embedded in another integrated circuit device.

19.   (Previously issued; now pending): The integrated circuit device of claim 15, wherein said sub-integrated circuit blocks are AND or OR gates and said integrated circuit device is a programmable logic device (PLD).

20.   (Previously issued; now pending): The integrated circuit device of claim 1, wherein said sub-integrated circuit blocks comprising any arbitrary hardware logic or memory circuits.

21.   (Previously issued; now pending): The integrated circuit device of claim 1, wherein said switches comprising active one-time programmable cross points and said integrated circuit device is a mask programmable gate array (MPGA) device or a structured ASIC device.

22.   (Previously issued; now cancelled): The integrated circuit device of claim 1, wherein said switches comprising passive cross points or just connection of two links or not and said integrated circuit device is a Application Specific Integrated Circuit (ASIC) device.

23.   (Previously issued; now cancelled): The integrated circuit device of claim 1, wherein said sub-integrated circuit blocks further recursively comprise one or more super-sub-integrated circuit blocks and a sub-routing network.

24.   (Previously issued; now pending): The integrated circuit device of claim 4, wherein said all horizontal shuffle exchange links between switches in any two corresponding said succeeding stages are of different length and said vertical shuffle exchange links between switches in any two corresponding said succeeding stages are of

5   different length and $y \geq \left(\log_2 N\right)$, where $N > 1$.

25.   (Previously issued; now pending): The integrated circuit device of claim 24, wherein $d = 2$ and there is only one switch in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there is only one switch in each said stage in each said sub-integrated circuit block connecting said

10   backward connecting links and said routing network is rearrangeably nonblocking for unicast generalized multi-stage network with full bandwidth.

26.   (Previously issued; now pending): The integrated circuit device of claim 24, wherein $d = 2$ and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least

15   two switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is strictly nonblocking for unicast generalized multi-stage network and rearrangeably nonblocking for arbitrary fan-out multicast generalized multi-stage network with full bandwidth.

27.   (Previously issued; now pending): The integrated circuit device of claim

20   24, wherein $d = 2$ and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is strictly nonblocking for arbitrary fan-out multicast generalized multi-stage network with full bandwidth.

25   28.   (Previously issued; now pending): The integrated circuit device of claim 4, wherein said all horizontal shuffle exchange links between switches in any two corresponding said succeeding stages are of different length and said vertical shuffle

Reissue of US Patent No. 8,269,523   Attorney Docket No. V-0070US
Reissue Application Filed: 11/27/18

exchange links between switches in any two corresponding said succeeding stages are of different length and $y \geq (\log_2 N)$, where $N > 1$, and

said each sub-integrated circuit block further comprising a plurality of U-turn links within switches in each of said stages in each of said sub-integrated circuit blocks.

5   29.   (Previously issued; now pending): The integrated circuit device of claim 28, wherein $d = 2$ and there is only one switch in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there is only one switch in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is rearrangeably nonblocking for

10   unicast generalized butterfly fat tree network with full bandwidth.

30.   (Previously issued; now pending): The integrated circuit device of claim 28, wherein $d = 2$ and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said

15   backward connecting links and said routing network is strictly nonblocking for unicast generalized butterfly fat tree Network and rearrangeably nonblocking for arbitrary fan-out multicast generalized butterfly fat tree network with full bandwidth.

31.   (Previously issued; now pending): The integrated circuit device of claim 28, wherein $d = 2$ and there are at least three switches in each said stage in each said sub-

20   integrated circuit block connecting said forward connecting links and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is strictly nonblocking for arbitrary fan-out multicast generalized butterfly fat tree network with full bandwidth.

32.   (Previously issued; now cancelled): The integrated circuit device of claim

25   1, wherein said straight links connecting from switches in each said sub-integrated circuit block are connecting to switches in the same said sub-integrated circuit block; and

Page 10 of 15

said cross links are connecting as vertical or horizontal or diagonal links between two different said sub-integrated circuit blocks.

33.   (Previously issued; now cancelled): The integrated circuit device of claim 7, wherein $d = 4$ and there is only one switch in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there is only one switch in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is rearrangeably nonblocking for unicast multi-link Benes network with full bandwidth.

34.   (Previously issued; now cancelled): The integrated circuit device of claim 7, wherein $d = 4$ and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is strictly nonblocking for unicast multi-link Benes network and rearrangeably nonblocking for arbitrary fan-out multicast multi-link Benes network with full bandwidth.

35.   (Previously issued; now cancelled): The integrated circuit device of claim 7, wherein $d = 4$ and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is strictly nonblocking for arbitrary fan-out multicast multi-link Benes network with full bandwidth.

36.   (Previously issued; now cancelled): The integrated circuit device of claim 11, wherein $d = 4$ and there is only one switch in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there is only one switch in each said stage in each said sub-integrated circuit block connecting said

backward connecting links and said routing network is rearrangeably nonblocking for unicast multi-link butterfly fat tree network with full bandwidth.

37.   (Previously issued; now cancelled): The integrated circuit device of claim 11, wherein $d = 4$ and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is strictly nonblocking for unicast multi-link butterfly fat tree network and rearrangeably nonblocking for arbitrary fan-out multicast multi-link butterfly fat tree network with full bandwidth.

38.   (Previously issued; now cancelled): The integrated circuit device of claim 11, wherein $d = 4$ and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is strictly nonblocking for arbitrary fan-out multicast multi-link butterfly fat tree network with full bandwidth.

39.   (Previously issued; now cancelled): The integrated circuit device of claim 4, wherein said all horizontal shuffle exchange links between switches in any two corresponding said succeeding stages are of different length and said vertical shuffle exchange links between switches in any two corresponding said succeeding stages are of different length and $y \geq \left(\log_2 N \right)$, where $N > 1$.

40.   (Previously issued; now cancelled): The integrated circuit device of claim 39, wherein $d = 4$ and there is only one switch in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there is only one switch in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is rearrangeably nonblocking for unicast generalized multi-link multi-stage network with full bandwidth.

Reissue of US Patent No. 8,269,523   Attorney Docket No. V-0070US
Reissue Application Filed: 11/27/18

41.   (Previously issued; now cancelled): The integrated circuit device of claim 39, wherein $d = 4$ and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is strictly nonblocking for unicast generalized multi-link multi-stage network and rearrangeably nonblocking for arbitrary fan-out multicast generalized multi-link multi-stage network with full bandwidth.

42.   (Previously issued; now cancelled): The integrated circuit device of claim 39, wherein $d = 4$ and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is strictly nonblocking for arbitrary fan-out multicast generalized multi-link multi-stage network with full bandwidth.

43.   (Previously issued; now cancelled): The integrated circuit device of claim 4, wherein said all horizontal shuffle exchange links between switches in any two corresponding said succeeding stages are of different length and said vertical shuffle exchange links between switches in any two corresponding said succeeding stages are of different length and $y \geq \left(\log_2 N\right)$, where $N > 1$, and

said each sub-integrated circuit block further comprising a plurality of U-turn links within switches in each of said stages in each of said sub-integrated circuit blocks.

44.   (Previously issued; now cancelled): The integrated circuit device of claim 43, wherein $d = 4$ and there is only one switch in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there is only one switch in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is rearrangeably nonblocking for unicast generalized multi-link butterfly fat tree network with full bandwidth.

45.   (Previously issued; now cancelled): The integrated circuit device of claim 43, wherein $d = 4$ and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least two switches in each said stage in each said sub-integrated circuit block connecting said

5   backward connecting links and said routing network is strictly nonblocking for unicast generalized multi-link butterfly fat tree Network and rearrangeably nonblocking for arbitrary fan-out multicast generalized multi-link butterfly fat tree network with full bandwidth.

46.   (Previously issued; now cancelled): The integrated circuit device of claim

10   43, wherein $d = 4$ and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said forward connecting links and there are at least three switches in each said stage in each said sub-integrated circuit block connecting said backward connecting links and said routing network is strictly nonblocking for arbitrary fan-out multicast generalized multi-link butterfly fat tree network with full bandwidth.

15   47.   (Previously issued; now pending): The integrated circuit device of claim 1, wherein said plurality of forward connecting links use a plurality of buffers to amplify signals driven through them and said plurality of backward connecting links use a plurality of buffers to amplify signals driven through them; and said buffers can be inverting or non-inverting buffers.

20   48.   (Previously issued; now pending): The integrated circuit device of claim 1, wherein said all switches of size $d \times d$ are either fully populated or partially populated.

Reissue of US Patent No. 8,269,523   Attorney Docket No. V-0070US
Reissue Application Filed: 11/27/18

REMARKS

The present amendment is made to cancel some claims issued U.S. Patent No. 8,269,523 to Venkat Konda and is bypass continuation and claims priority of the PCT Application Serial No. PCT/US08/64605 filed May 22, 2008, which in turn claims priority of the

5   U.S. Provisional Patent Application Serial No. 60/940,394 filed May 25, 2007.

No new claims are added. Originally issued claims 5 – 14, 17, 22, 23, 32 – 46 are now cancelled. Therefore, originally issued claims 1 - 4, 15, 16, 18 – 21, 24 – 31, 47, 48 are pending in this reissue application following entry of this amendment.

Should the Office have any questions concerning this communication, please contact the

10   inventor Venkat Konda at (408) 472-3273.

Very respectfully,

/Venkat Konda/

Venkat Konda

15   Konda Technologies, Inc (USPTO Customer Number: 38139)

6278 Grand Oak Way

San Jose, CA 95135

Phone: 408-472-3273

Fax: 408-238-2478

US008269523B2

(12) **United States Patent**   (10) **Patent No.:**   **US 8,269,523 B2**

Konda   (45) **Date of Patent:**   **Sep. 18, 2012**

(54) **VLSI LAYOUTS OF FULLY CONNECTED GENERALIZED NETWORKS**

(75) Inventor: **Venkat Konda**, San Jose, CA (US)

(73) Assignee: **Konda Technologies Inc.**, San Jose, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 183 days.

(21) Appl. No.: **12/601,275**

(22) PCT Filed: **May 22, 2008**

(86) PCT No.: **PCT/US2008/064605**

§ 371 (c)(1),
(2), (4) Date: **May 31, 2010**

(87) PCT Pub. No.: **WO2008/147928**

PCT Pub. Date: **Dec. 4, 2008**

(65) **Prior Publication Data**

US 2011/0037498 A1   Feb. 17, 2011

Related U.S. Application Data

(60) Provisional application No. 60/940,394, filed on May 25, 2007.

(51) **Int. Cl.**
H03K 19/177 (2006.01)
(52) **U.S. Cl.** **326/41**; 326/38
(58) **Field of Classification Search** 326/38–41; 370/390, 312, 360, 388, 412
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,813,038	A	*	3/1989	Lee	370/390
5,406,556	A	*	4/1995	Widjaja et al.	370/381
5,856,977	A	*	1/1999	Yang et al.	370/395.72
6,049,542	A	*	4/2000	Prasad	370/386
6,157,643	A	*	12/2000	Ma	370/389
6,567,858	B1	*	5/2003	Yang et al.	709/238
6,940,308	B2	*	9/2005	Wong	326/41
7,130,920	B1	*	10/2006	Sailor	709/238
7,136,380	B2	*	11/2006	Li	370/388
7,154,887	B2	*	12/2006	Wu et al.	370/388
7,346,049	B2	*	3/2008	Towles	370/386
7,349,387	B2	*	3/2008	Wu	370/360

(Continued)

Primary Examiner — Vibol Tan

(57) **ABSTRACT**

In accordance with the invention, VLSI layouts of generalized multi-stage networks for broadcast, unicast and multicast connections are presented using only horizontal and vertical links. The VLSI layouts employ shuffle exchange links where outlet links of cross links from switches in a stage in one sub-integrated circuit block are connected to inlet links of switches in the succeeding stage in another sub-integrated circuit block so that said cross links are either vertical links or horizontal and vice versa. In one embodiment the sub-integrated circuit blocks are arranged in a hypercube arrangement in a two-dimensional plane. The VLSI layouts exploit the benefits of significantly lower cross points, lower signal latency, lower power and full connectivity with significantly fast compilation.
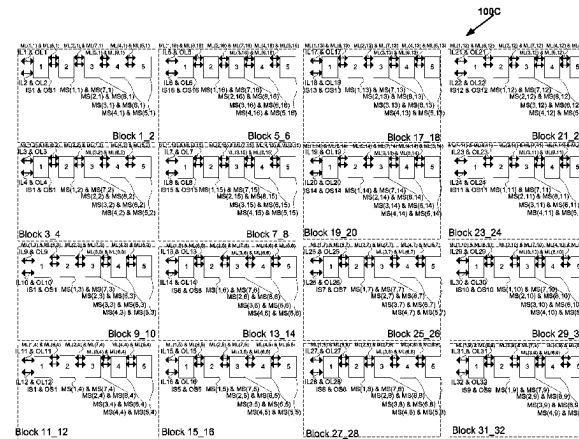
The VLSI layouts presented are applicable to generalized multi-stage networks $V(N_1, N_2, d, s)$, generalized folded multi-stage networks $V_{fold}(N_1, N_2, d, s)$, generalized butterfly fat tree networks $V_{bft}(N_1, N_2, d, s)$, generalized multi-link multi-stage networks $V_{mlink}(N_1, N_2, d, s)$, generalized folded multi-link multi-stage networks $V_{fold-mlink}(N_1, N_2, d, s)$, generalized multi-link butterfly fat tree networks $V_{mlink-bft}(N_1, N_2, d, s)$, and generalized hypercube networks $V_{hcube}(N_1, N_2, d, s)$ for s=1, 2, 3 or any number in general. The embodiments of VLSI layouts are useful in wide target applications such as FPGAs, CPLDs, pSoCs, ASIC placement and route tools, networking applications, parallel & distributed computing, and reconfigurable computing.

48 Claims, 39 Drawing Sheets

US 8,269,523 B2

Page 2

U.S. PATENT DOCUMENTS

7,397,796	B1 *	7/2008	Smiljani	370/390
7,424,010	B2 *	9/2008	Konda	370/388
7,424,011	B2 *	9/2008	Konda	370/388
7,468,974	B1 *	12/2008	Carr et al.	370/388
7,924,052	B1 *	4/2011	Feng et al.	326/41
8,098,081	B1 *	1/2012	Trimberger	326/41

* cited by examiner

PATENT APPLICATION FEE DETERMINATION RECORD Substitute for Form PTO-875	Application or Docket Number 16/202,067	Filing Date 11/27/2018	☐ To be Mailed

ENTITY: ☐ LARGE ☑ SMALL ☐ MICRO

APPLICATION AS FILED - PART I

FOR	(Column 1) NUMBER FILED	(Column 2) NUMBER EXTRA		RATE ($)	FEE ($)
☐ BASIC FEE (37 CFR 1.16(a), (b), or (c))	N/A	N/A		N/A	
☐ SEARCH FEE (37 CFR 1.16(k), (i), or (m))	N/A	N/A		N/A	
☐ EXAMINATION FEE (37 CFR 1.16(o), (p), or (q))	N/A	N/A		N/A	
TOTAL CLAIMS (37 CFR 1.16(i))	minus 20 =	*		x $50 =	
INDEPENDENT CLAIMS (37 CFR 1.16(h))	minus 3 =	*		x $230 =	
☐ APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is $310 ($155 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).				
☐ MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))					
* If the difference in column 1 is less than zero, enter "0" in column 2.				TOTAL	

APPLICATION AS AMENDED - PART II

AMENDMENT	11/27/2018	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA		RATE ($)	ADDITIONAL FEE ($)
	Total (37 CFR 1.16(i))	* 20	Minus	** 20	= 0		x $50 =	0
	Independent (37 CFR 1.16(h))	* 1	Minus	*** 3	= 0		x $230 =	0
	☐ Application Size Fee (37 CFR 1.16(s))							
	☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							
							TOTAL ADD'L FEE	0

AMENDMENT		(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA		RATE ($)	ADDITIONAL FEE ($)
	Total (37 CFR 1.16(i))	*	Minus	**	=		x $0 =	
	Independent (37 CFR 1.16(h))	*	Minus	***	=		x $0 =	
	☐ Application Size Fee (37 CFR 1.16(s))							
	☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							
							TOTAL ADD'L FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.	LDRC
** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".	/EVA V GILLIS/
*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".	
The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.	