

UNITED STATES PATENT AND TRADEMARK OFFICE

---

**BEFORE THE PATENT TRIAL AND APPEAL BOARD**

---

---

**DECLARATION OF JAMES L. MULLINS, PH.D.**

## TABLE OF CONTENTS

	<b>Page</b>
I. INTRODUCTION .....	1
II. BACKGROUND AND QUALIFICATIONS .....	1
III. PRELIMINARIES .....	3
IV. OPINION REGARDING INDIVIDUAL DOCUMENTS .....	19
CONCLUSION .....	39

I, James L. Mullins, hereby declare under penalty of perjury:

## **I. INTRODUCTION**

1. I have knowledge of the facts and opinions set forth in this declaration, I believe them to be true, and if called upon to do so, I would testify competently to them. I have been warned that willful false statements and the like are punishable by fine or imprisonment, or both.

2. I am a retired academic librarian working as the Founder and Owner of the firm Prior Art Documentation Librarian Services, LLC at 106 Berrow, Williamsburg, VA 23188. Attached as Appendix A is a true and correct copy of my Curriculum Vitae describing my background and experience. Further information about my firm, Prior Art Documentation Librarian Services, LLC (PADLS), is available at [www.priorartdoclib.com](http://www.priorartdoclib.com).

3. I have been retained by Sidley Austin LLP to authenticate and establish the dates of public accessibility of certain documents for use in one or more *inter partes* review proceedings. For this service, I am being paid my usual hourly fee of \$185/hour. My compensation in no way depends on the substance of my testimony or the outcome of the proceeding.

## **II. BACKGROUND AND QUALIFICATIONS**

4. Presently, I am the Dean of Libraries Emeritus and Esther Ellis Norton Professor Emeritus, Purdue University.

5. I was previously employed as follows:

- Dean of Libraries and Professor & Esther Ellis Norton Professor, Purdue University, West Lafayette, IN, 2004-2017.
- Assistant/Associate Director for Administration, Massachusetts Institute of Technology Libraries, Cambridge, MA, 2000-2004.
- University Librarian and Director, Falvey Memorial Library, Villanova University, Villanova, PA, 1996-2000.
- Director of Library Services, Indiana University South Bend, South Bend, IN, 1978-1996. Part-time instructor, School of Library and Information Science, Indiana University, Bloomington, IN, 1979-1996.
- Associate Law Librarian, and associated titles, Indiana University School of Law, Bloomington, IN, 1974-1978.
- Catalog Librarian, Assistant Professor, Georgia Southern College (now University), Statesboro, GA, 1973-1974.

6. Over the course of my career as a librarian, instructor of library science, author of scholarly publications, and presenter at national and international conferences, I have had experience with catalog records and online library management systems built around Machine-Readable Cataloging (MARC) standards.

7. In the course of more than forty-four years as an academic librarian and scholar, I have been an active researcher. In my years as a librarian I have facilitated the research of faculty colleagues either directly or through the provision of and access to the requisite print and/or digital materials and services at the universities I worked. I have kept current on the professional library science literature and served



on the editorial board of the most prominent library journal, *College and Research Libraries*. This followed service as the chair of the Research Committee of the Association of College and Research Libraries (ACRL), a division of the American Library Association (ALA). As an academic library administrator I have had responsibility to insure that students were educated to identify, locate, assess and integrate information garnered from library resources.

### **III. PRELIMINARIES**

8. *Scope of this declaration.* I am not a lawyer and I am not rendering an opinion on the legal question of whether any particular document is, or is not, a “printed publication” under the law.

9. I am, however, rendering my expert opinion on the authenticity of the documents referenced herein and on when and how each of these documents was disseminated or otherwise made available to the extent that persons interested and ordinarily skilled in the subject matter or art, exercising reasonable diligence, could have located the documents no later than September 11, 2007.

10. *Materials considered.* In forming the opinions expressed in this declaration, I have reviewed the documents and attachments referenced herein. These materials are records created in the ordinary course of business by publishers, libraries, indexing services, and others. From my years of experience, I am familiar with the process for creating many of these records, and I know these records are

created by people with knowledge of the information in the record. Further, these records are created with the expectation that researchers and other members of the public will use them. All materials cited in this declaration and its attachments are of a type that experts in my field would reasonably rely upon and refer to in forming their opinions.

11. *Persons of ordinary skill in the art.* I have been informed by counsel that a person of ordinary skill in the art in the field of U.S. Patent Nos. 7,225,324 (“324 Patent”) and 7,620,800 (“800 Patent”) in the 2002 time frame would have had an advanced degree in electrical or computer engineering, or computer science with substantial study in computer architecture, hardware design, and computer algorithms, and at least three years’ experience working in the field. Alternatively, that person would have had a bachelor’s degree covering those disciplines and at least four years working the field. Such a person would have been knowledgeable about the programming, design and operation of computer systems based on reconfigurable components such as FPGAs (field programmable gate arrays) and CPLDs (complex programmable logic devices), including computer systems for performing systolic and data driven calculations. That person would also have been familiar with hardware description languages such as VHDL that could be used to configure FPGAs and CPLDs that serve as components of reconfigurable computer systems. Finally, such a person would also have been familiar with various other

areas of technology that by 2002 had relied on high performance and parallel computing systems.

12. I am told that Patent Owner claims a priority date of October 31, 2002. By that date, a person of ordinary skill would have had access to a vast array of long-established print resources in the field of field programmable gate array based systems and to a rich and fast changing set of online resources providing indexing information, abstracts, and full text services for persons interested in field programmable gate array based systems.

### **Library Catalog Records**

13. Some background on MARC (Machine-Readable Cataloging) formatted records, OCLC, and WorldCat is helpful to understand the library catalog records discussed in this declaration. I am fully familiar with the library cataloging standard known as the MARC standard, which is an industry-wide standard method of storing and organizing library catalog information.<sup>1</sup> MARC practices have been consistent since the MARC format was developed by the Library of Congress in the 1960s, and by the early 1970s became the U.S. national standard for disseminating bibliographic data. By the mid-1970s, MARC format became the international

<sup>1</sup> The full text of the standard is available from the Library of Congress at <http://www.loc.gov/marc/bibliographic/>.

standard, and persists through the present. A MARC-compatible library is one that has a catalog consisting of individual MARC records for each of its items. Today, MARC is the primary communications protocol for the transfer and storage of bibliographic metadata in libraries.<sup>2</sup> The MARC practices discussed below were in place during the mid to late 1990s timeframe relevant to the documents referenced herein.

14. Similarly, OCLC practices have been consistent since the 1970s through the present, and the OCLC practices discussed below were in place during the mid to late 1990s timeframe relevant to the documents referenced herein. The OCLC was created “to establish, maintain and operate a computerized library

<sup>2</sup> Almost every major library in the world is MARC-compatible. *See, e.g., MARC Frequently Asked Questions (FAQ)*, LIBRARY OF CONGRESS, <https://www.loc.gov/marc/faq.html> (last visited January 24, 2018) (“MARC is the acronym for MACHine-Readable Cataloging. It defines a data format that emerged from a Library of Congress-led initiative that began nearly forty years ago. It provides the mechanism by which computers exchange, use, and interpret bibliographic information, and its data elements make up the foundation of most library catalogs used today.”). MARC is the ANSI/NISO Z39.2-1994 (reaffirmed 2009) standard for Information Interchange Format.

network and to promote the evolution of library use, of libraries themselves, and of librarianship, and to provide processes and products for the benefit of library users and libraries, including such objectives as increasing availability of library resources to individual library patrons and reducing the rate of rise of library per-unit costs, all for the fundamental public purpose of furthering ease of access to and use of the ever-expanding body of worldwide scientific, literary and educational knowledge and information.”<sup>3</sup> Among other services, OCLC and its members are responsible for maintaining the WorldCat database (<http://www.worldcat.org/>), used by independent and institutional libraries throughout the world.

15. A complementary library organization, Research Library Group (RLG) was formed in 1974 by several major research universities including Yale, New York Public, Columbia, and Harvard as an alternative to OCLC. RLG created a database of bibliographic records and holdings identified as RLIN that used the MARC format in creating its OPAC and holding records at member institutions. RLG merged with OCLC in 2006 to form one unified library bibliographic, member driven utility.

<sup>3</sup> Third Article, Amended Articles of Incorporation of OCLC Online Computer Library Center, Incorporated (available at [http://www.oclc.org/en-US/councils/documents/amended\\_articles.html](http://www.oclc.org/en-US/councils/documents/amended_articles.html)).

16. Libraries world-wide have used the machine-readable MARC format for catalog records. MARC formatted records have provided a variety of subject access points based on the content of the document being cataloged. A MARC record comprises several fields, each of which contains specific data about the work. Each field is identified by a standardized, unique, three-digit code corresponding to the type of data that follows. For example, a work's title is recorded in field 245, the primary author of the work is recorded in field 100, an item's International Standard Book Number ("ISBN") is recorded in field 020, an item's Library of Congress call number is recorded in field 050, and the publication date is recorded in field 260 under the subfield "c." If a work is a periodical, then its publication frequency is recorded in field 310, and the publication dates (*e.g.*, the first and last publication) are recorded in field 362, which is also referred to as the enumeration/chronology field.

17. The MARC Field 040, subfield a, identifies the library or other entity that created the original catalog record for a given document and transcribed it into machine readable form. The MARC Field 008 identifies the date when this first catalog record was entered on the file. This date persists in all subsequent uses of the first catalog record, although newly-created records for the same document, separate from the original record will show a new date. It is not unusual to find multiple

catalog records for the same document, typically this is the result of the merger of RLG and OCLC.

18. MARC records also include several fields that include subject matter classification information. An overview of MARC record fields is available through the Library of Congress at <http://www.loc.gov/marc/bibliographic/>. For example, 6XX fields are termed “Subject Access Fields.”<sup>4</sup> Among these, for example, is the 650 field; this is the “Subject Added Entry – Topical Term” field. *See* <http://www.loc.gov/marc/bibliographic/bd650.html>. The 650 field is a “[s]ubject added entry in which the entry element is a topical term.” *Id.* These entries “are assigned to a bibliographic record to provide access according to generally accepted thesaurus-building rules (e.g., *Library of Congress Subject Headings* (LCSH), *Medical Subject Headings* (MeSH)).” *Id.* Thus, a researcher might discover material relevant to his or her topic by a search using the terms employed in the MARC Fields 6XX.

19. The 9XX fields are not part of the standard MARC 21 format.<sup>5</sup> OCLC has defined these 9XX fields for use by the Library of Congress and for internal OCLC use: 936, 938, 956, 987, 989, and 994. 955 is used by the Library of Congress

<sup>4</sup> *See* <http://www.loc.gov/marc/bibliographic/bd6xx.html>.

<sup>5</sup> *See* <https://www.oclc.org/bibformats/en/9xx.html>.

to track the progress of a new acquisition from the time it is submitted for Cataloging in Publication (CIP) review until it is published and fully cataloged and available for use within the Library of Congress. Fields 901-907, 910, and 945-949 have been defined by OCLC for local use and will pass OCLC validation. Fields 905 or 910 are often used by an individual library for internal processing purposes, for example the date of cataloging and the initials of the cataloger.

20. Further, MARC records include call numbers, which themselves include a classification number. For example, the 050 field is the “Library of Congress Call Number.”<sup>6</sup> A defined portion of the Library of Congress Call Number is the classification number, and “source of the classification number is *Library of Congress Classification* and the *LC Classification-Additions and Changes*.” *Id.* Thus, included in the 050 field is a subject matter classification. Each item in a library has a single classification number. A library selects a classification scheme (e.g., the Library of Congress Classification scheme just described or a similar scheme such as the Dewey Decimal Classification scheme) and uses it consistently. When the Library of Congress assigns the classification number, it appears as part of the 050 field. If a local library assigns the classification number, it appears in a

<sup>6</sup> See <http://www.loc.gov/marc/bibliographic/bd050.html>.



090 field. In either scenario, the MARC record includes a classification number that represents a subject matter classification.

21. WorldCat is the world's largest public online catalog, maintained by the Online Computer Library Center, Inc., or OCLC, and built with the records created by the thousands of libraries that are members of OCLC. OCLC has provided bibliographic and abstract information to the public based on MARC records through its OCLC WorldCat database. WorldCat requires no knowledge of MARC tags and code and does not require a log-in or password. WorldCat is easily accessible through the World Wide Web to all who wish to search it; there are no restrictions to be a member of a particular community, etc. The date a given catalog record was created (corresponding to the MARC Field 008) appears in some detailed WorldCat records as the Date of Entry but not necessarily all. Whereas WorldCat records are widely available, the availability of MARC formatted records varies from library to library and when made available will be identified as MARC record or librarian/staff view.

22. When an OCLC member institution acquires a work, it creates a MARC record for this work in its computer catalog system in the ordinary course of its business. MARC records created at the Library of Congress were historically tape-loaded into the OCLC database through a subscription to MARC Distribution Services daily or weekly. Once the MARC record is created by a cataloger at an

OCLC member institution or is tape-loaded from the Library of Congress, the MARC record is then made available to any other OCLC members online, and therefore made available to the public. Accordingly, once the MARC record is created by a cataloger at an OCLC member institution or is tape-loaded from the Library of Congress or another library anywhere in the world, any publication corresponding to the MARC record has been cataloged and indexed according to its subject matter such that a person interested in that subject matter could, with reasonable diligence, locate and access the publication through any library with access to the OCLC WorldCat database or through the Library of Congress.

23. When an OCLC member institution creates a new MARC record, OCLC automatically supplies the date of creation for that record. The date of creation for the MARC record appears in the fixed field (008), characters 00 through 05. The MARC record creation date reflects the date on which the item was first acquired or cataloged. Initially, field 005 of the MARC record is automatically populated with the date the MARC record was created in year, month, day format (YYYYMMDD) (some of the newer library catalog systems also include hour, minute, second (HHMMSS)). Thereafter, the library's computer system may automatically update the date in field 005 every time the library updates the MARC record (*e.g.*, to reflect that an item has been moved to a different shelving location within the library).

24. Once one library has cataloged and indexed a publication by creating a MARC record for that publication, other libraries that receive the publication do not create additional MARC records—the other libraries instead rely on the original MARC record. They may update or revise the MARC record to ensure accuracy, but they do not replace or duplicate it. This practice does more than save libraries from duplicating labor. It also enhances the accuracy of MARC records. Further, it allows librarians around the world to know that a particular MARC record is authoritative (in contrast, a hypothetical system wherein duplicative records were created would result in confusion as to which record is authoritative).

25. The date of creation of the MARC record by a cataloger at an OCLC member institution reflects when the underlying item is accessible to the public. Upwards of two-thirds to three-quarters of book sales to libraries come from a jobber or wholesaler for online and print resources. These resellers make it their business to provide books to their customers as fast as possible, often providing turnaround times of only a single day after publication. Libraries purchase a significant portion of the balance of their books directly from publishers themselves, which provide delivery on a similarly expedited schedule. In general, libraries make these purchases throughout the year as the books are published and shelve the books as soon thereafter as possible in order to make the books available to their patrons.

Thus, books are generally available at libraries across the country within just a few weeks of publication.

### **Monograph Publications**

26. Monograph publications are written on a single topic, presented at length and distinguished from an article and include books, dissertations, and technical reports. A library typically creates a catalog record when the monograph is acquired by the library. First, it will search OCLC to determine if a record has already been created by the Library of Congress or a contributing member library. If the record is found in OCLC, the record is downloaded into the library's LMS (Library Management System) that includes typically the acquisitions, cataloging, and circulation integrated functions. Once the item is downloaded into the library's LMS, the library adds its identifier to the OCLC database so when a search is completed on WorldCat, the library will be indicated as an owner of the title. With the creation of the record in the LMS it is searchable and viewable through the library Online Public Access Catalog (OPAC), by author, title, and subject heading, at the library and from anywhere in the world through the internet. The OPAC also connects with the circulation function of the library which typically indicates whether the book, dissertation, tech report is available, in circulation, etc., with its call number and location in a specific departmental/disciplinary library. The OPAC not only provides immediate bibliographic access on site, it also facilitates

interlibrary loan of items, that is, the loan of an item from one library to another to meet a research need.

### **Ownership and Date Stamp**

27. Every library sets its own practice or policy on whether-or-not to date stamp, but all will have an ownership stamp somewhere in the publication — typically on the cover page, verso of the cover page, or a designated page within the publication, but sometimes even on the top, side, or bottom edge of the monograph. The ownership and date stamp can also vary from one library to another when the date stamp is entered on the monograph. It could occur when received in acquisitions after shipment to the library, or it could be at time of cataloging. Therefore, there could be instances when the date of receipt precedes the cataloging date or vice versa.

### **Periodicals, Indexes, and Citation Sources**

#### **a. Publications in Series: Conference Proceedings/Technical Report publications**

28. A library typically creates a MARC catalog record for a series of closely related publications, such as the proceedings of an annual conference or a technical report when the library receives its first issue and assumes there will be annual or succeeding issues/volumes/reports. When the institution receives subsequent issues/volumes/reports of the series, the issues/volumes/reports are checked in (sometimes using a date stamp), added to the institution's holdings

records, and made available very soon thereafter—normally within a few days of receipt or (at most) within a few weeks of receipt. The initial series record may not reflect all subsequent changes in publication details (including minor variations in title, etc.).

**b. Indexing**

29. A researcher may discover material relevant to his or her topic in a variety of ways. One common means of discovery is to search for relevant information in an index of periodical and other publications. Having found relevant material, the researcher will then normally obtain it online, look for it in libraries, or purchase it from the publisher, a bookstore, a document delivery service, or other provider. Sometimes, the date of a document's public accessibility will involve both indexing and library date information. Date information for indexing entries is, however, often unavailable. This is especially true for online indices. Indexing services use a wide variety of controlled vocabularies to provide subject access and other means of discovering the content of documents. The formats in which these access terms are presented vary from service to service.

30. Online indexing services commonly provide bibliographic information, abstracts, and full-text copies of the indexed publications, along with a list of the documents cited in the indexed publication. These services also often provide lists of publications that cite a given document. A citation of a document is evidence that

the document was publicly available and in use by researchers no later than the publication date of the citing document.

31. IEEE Xplore Digital Library. The Institute of Electrical and Electronics Engineers is the world's largest organization for the advancement of technology, with some 430,000 members in 160 countries. Known by its acronym IEEE, it has created IEEE Xplore Digital Library, which provides access to the contents of over 170 journals, more than 1,400 conference proceedings, some 5,100 technical standards, 2,000 eBooks, and 400 educational courses. More than 3 million documents, dating from 1872, are searchable and available either through subscription or individual purchase.

32. Wiley Online Library. Produced by John Wiley & Sons, Wiley Online Library is a collection of online resources covering life, health and physical sciences, the social sciences, and the humanities. It provides access to over 6 million articles from over 1,500 journals and over 19,000 online books.

**c. Citation Sources**

33. Web of Science. Like its print predecessors Science Citation Index, Social Science Citation Index, and Arts and Humanities Citation Index, Web of Science provides thorough coverage of a broad set of disciplines. A Thomson Reuters product, Web of Science indexes 1,700 arts and humanities journals from

1975 to the present, 8,500 scientific journals from 1900 to the present, and some 300 social science journals from 1900 to the present.

34. Science Direct. Science Direct, provided by the major publisher Elsevier, is a database of abstracts and articles in the physical sciences and engineering, the life and health sciences, and the social sciences and humanities. It has over 12 million items from 3,500 journals and 34,000 books.

35. Scopus. Produced by Elsevier, a major publisher, Scopus is the largest database of abstracts and citations of peer-reviewed literature. Its scope includes the social sciences, science, technology, medicine, and the arts. It includes 60 million records from more than 21,500 titles from some 5,000 international publishers. Coverage includes 360 trade publications, over 530 book series, more than 7.2 million conference papers, and 116,000 books. Records date from 1823.

36. Google Scholar. Google Scholar indexes the texts and metadata of scholarly publications across a wide range of disciplines. It includes most peer-reviewed online academic journals, conference papers, theses, technical reports, and other material. Google does not publish the size of the Google Scholar database, but researchers have estimated that it contained approximately 160 million items in 2014 (Enrique Oduña-Malea, et al., “About the size of Google Scholar: playing the numbers,” Granada: EC3 Working Papers, 1B: 23 July 2014, available at <https://arxiv.org/ftp/arxiv/papers/1407/1407.6239.pdf>).



#### IV. OPINION REGARDING INDIVIDUAL DOCUMENTS

**Document 1: Carl Ebeling, et al., “Mapping Applications to the RaPiD Configurable Architecture,” IEEE Symposium on FPGAs for Custom Computing Machines. April 16-18, 1997 Napa Valley, California: 106-115.**

##### **Authentication**

37. Document 1 is an article by Carl Ebeling, et al., titled “Mapping Applications to the RaPiD Configurable Architecture” presented at the IEEE Symposium on FPGAs for Custom Computing Machines, published in the Proceedings: IEEE Symposium on FPGAs for Custom Computer Machines, held on April 16-18, 1997 Napa Valley, California.

38. Attachment 1A was provided to me by Wisconsin TechSeach (WTS) from the Library of Congress. It includes the cover, title page, verso of the title page, table of contents and the article by Carl Ebeling, et al., “Mapping Applications to the RaPiD Configurable Architecture.” On the title page is the ownership stamp of the Library of Congress with the check-in date of October 9, 1997. On the verso of the title page is the call number hand written of: TK7895.G36 I35 1997.

39. Attachment 1B, downloaded by me, includes Document 1 in PDF format from IEEE Xplore through Purdue University Libraries: <https://ieeexplore-ieee-org.ezproxy.lib.purdue.edu/document/624610/>.

40. Attachment 1A is in a condition that creates no suspicion about its authenticity. Specifically, Attachment 1A is not missing any intermediate pages, the text on each page appears to flow seamlessly from one page to the next, and there

are no visible alterations to the document. Attachment 1A was found within the custody of a library, the Library of Congress – a place where, if authentic, it would likely be found.

41. Based on finding Attachment 1A in a library and Document 1 (Attachment 1B) online at IEEE Xplore, I conclude that Document 1 is an authentic document and that Attachment 1A is an authentic copy of Document 1.

### **Public Accessibility**

42. Attachment 1C is the WorldCat record. This record shows that Document 1 is held by 207 libraries world-wide as of July, 2018. Subject access is possible through subject headings: Field Programmable Data Arrays – Congresses; Computer Engineering – Congresses; and Computer Engineering. Library of Congress is listed as one of the holding libraries among the 207 as of July, 2018.

43. Attachment 1D is the Library of Congress MARC record for Document 1. In MARC field 955 is the record of the processing of Document 1. MARC field 955 is used by the Library of Congress to track the progress of a new acquisition from the time it is submitted for Cataloging in Publication (CIP) review until it is published and fully cataloged and available for use within the Library of Congress. In Attachment 1D, MARC field 955 contains the phrase “pb23 11-18-97 to cat,” which indicates that a Library of Congress librarian noted that Document 1 was sent to cataloging on November 18, 1997. In addition, MARC field 955 in Attachment

D contains the phrase “jg00 11-26-97.” That field further indicates that a librarian noted that Document 1 was sent to the Library of Congress stacks November 26, 1997.

44. Attachment 1E is the Library of Congress Online Catalog (OPAC) record. On page 2, this OPAC record indicates that there are two copies of Document 2 available, locatable by call numbers TK7895.G36 I35 1997 or TK7895.G36 I35 1997 FT MEADE. That record states that copy 1, call number TK7895.G36 I35 1997 (the same call number written on Attachment 1A), is available upon request in the Jefferson or Adams Building Reading Rooms. From my familiarity with the Library of Congress’ procedures, I would expect that based on this record, any person could request Document 1 be brought to one of the reading rooms for review. A request for photocopies or scans could be made from Document 1. Items do not circulate outside the Reading Rooms except to Members of Congress. I conclude from the information in Attachments 1D and 1E that Document 1 would have been publicly accessible by interested persons at the Library of Congress no later than the date it was sent to the stacks, on November 26, 1997.

45. Attachment 1F is a paper by Darren C. Cronquist et al., “Specifying and Compiling Applications for RaPiD” from the Proceedings, IEEE Symposium on FPGAs for Custom Computing Machines, April 17, 1998 pp. 116-125. Number 2 of the References on page 116 is Document 1. Downloaded by me on July 5, 2018,

from IEEE Xplore through Purdue University Libraries: <https://ieeexplore-ieee-org.ezproxy.lib.purdue.edu/stamp/stamp.jsp?tp=&arnumber=707889>.

## **Conclusion**

46. Based on the evidence presented here I conclude that Attachment 1A is an authentic copy of Document 1, which is an article published in the Proceedings: IEEE Symposium on FPGAs for Custom Computer Machines, held on April 16-18, 1997 Napa Valley, California. Based on the Library of Congress's records showing the receipt of this document on October 9, 1997, the cataloguing and availability of this document at the end of November of 1997, and the fact this this document was cited in a published article in 1998, it is my opinion that Document 1 was publicly accessible to and in actual use by ordinarily skilled researchers as early as the first part of December, 1997.

**Document 2: Michael Rencher and Brad L. Hutchins, "Automated Target Recognition on Splash 2," IEEE Symposium on FPGAs for Custom Computing Machines. April 16-18, 1997 Napa Valley, California: 192-200**

## **Authentication**

47. Document 2 is an article by Michael Rencher and Brad L. Hutchins titled "Automated Target Recognition on Splash 2" presented as a paper at the IEEE Symposium on FPGAs for Custom Computing Machines held on April 16-18, 1997 Napa Valley, California.

48. Attachment 2A a true and accurate copy of Document 2 including the cover page, title page, verso of the title page, and table of contents for the 1997 IEEE

Symposium, and the article comprising Document 2. On the cover is the ownership and date stamp of the British Library's Boston Spa facility, November 10, 1997 indicating the date that the British Library received Document 2. Attachment 2A was supplied to me by the Wisconsin TechSearch (WTS) on July 2, 2018.

49. Attachment 2B of Document 2 was downloaded by me through Purdue University Libraries online through IEEE Xplore <https://ieeexplore-ieee-org.ezproxy.lib.purdue.edu/stamp/stamp.jsp?tp=&arnumber=624619>.

50. Attachment 2A is in a condition that creates no suspicion about its authenticity. Specifically, Attachment 2A is not missing any intermediate pages, the text on each page appears to flow seamlessly from one page to the next, and there are no visible alterations to the document. Attachment 2A was found within the custody of a library (the British Library) – a place where, if authentic, it would likely be found.

51. Based on finding Attachment 2A in a library, the British Library, and locating Document 2 online at IEEE Xplore, I conclude that Document 2 is an authentic document and that Attachment 2A is an authentic copy of Document 2.

### **Public Accessibility**

52. Attachment 2C is the WorldCat record that shows British Library, Boston Spa as one of the libraries holding Document 2. The record shows that Document 2 is held by 207 libraries world-wide as of July 2018. Subject access is

possible through subject headings: Field Programmable Data Arrays – Congresses; Computer Engineering – Congresses; and Computer Engineering.

53. Attachment 2D is the British Library MARC and OPAC records for Document 2. Attachment 2D indicates through MARC field 260 that Document 2 has a copyright date of 1997, meaning that a librarian noted that the Document 2 claimed to have been copyrighted in the year 1997. Since there are no special notes in this record to indicate otherwise, I would expect that this document was catalogued and sent to the stacks for public access within a few weeks of receipt by the British Library. Thus, based on the date stamp of November 10, 1997 included on Attachment 2A and the records shown in Attachment 2D, I conclude that Attachment 2A would have been available in the stacks of the British Library no later than the end of November, 1997. As additional evidence, I further note that since Document 1 and Document 2 are from the same monograph, the processing identified by the Library of Congress MARC record in Attachment 1D should apply as well as Document 2. Since Document 1 was sent to cataloging on November 18, 1997 and went to the Library of Congress stacks November 26, 1997, I would expect the same of Document 2.

54. Attachment 2E is a true and accurate copy of a paper by John Villasenor and Brad Hutchings, “The Flexibility of Configurable Computing” IEEE Signal Processing Magazine, September 1998: 67-84. Number 35 of the References on page

94 is Document 2. Typically, due to the requirements of editing and processing, an article will have been completed approximately 1 year prior to its publication date. Thus, I would expect that Document 2 would have been available to the authors of Attachment 2E no later than the end of November, 1997, for it to appear as an article published in September of 1998.

### **Conclusion**

55. Based on the evidence presented here, I conclude that Attachment 2A is an authentic copy of Document 2, which is an article published in Proceedings: IEEE Symposium on FPGAs for Custom Computer Machines, held on April 16-18, 1997 Napa Valley, California. Based on the British Library's records showing the receipt of this document on November 10, 1997, the lack of special notes in the document's records, and the fact this this document was cited in a published article in September of 1998, it is my opinion that Document 2 was publicly accessible to and in actual use by ordinarily skilled researchers no later than the end of November 1997.

**Document 3: Jean-Luc Gaudiot, "Data Driven Multicomputers in Digital Signal Processing." Proceedings of the IEEE, 1987, vol. 75, no. 9: 1220-1234.**

### **Authentication**

56. Document 3 is an article by Jean-Luc Gaudiot titled "Data Driven Multicomputers in Digital Signal Processing," 1987, IEEE, Proceedings, volume 75, number 9, pages 1220-1234.

57. Attachment 3A includes the cover of the September 1987 issue of IEEE Proceedings: table of contents and Document 3. On the cover of the issue is the stamp that identifies it as the property of Linda Hall Library, and the date October 23, 1987, indicating when it was checked-in by the Linda Hall Library. Attachment 3A was provided to me by the Wisconsin TechSearch (WTS).

58. Document 3 is available on-line through IEEE Xplore. Attachment 3B is a download I completed of Document 3 from IEEE Xplore on June 30, 2018 through the Purdue University Libraries: <https://ieeexplore-ieee-org.ezproxy.lib.purdue.edu/stamp/stamp.jsp?tp=&arnumber=1458142>.

59. Attachment 3A is in a condition that creates no suspicion about its authenticity. Specifically, Attachment 3A is not missing any intermediate pages, the text on each page appears to flow seamlessly from one page to the next, and there are no visible alterations to the document. Attachment 3A was found within the custody of a library, the Linda Hall Library – a place where, if authentic, it would likely be found. The text of the article is also consistent in Attachment 3A and 3B. I see no reason to question the authenticity of the cover and its stamp or other preliminary pages in Attachment 3A.

60. Based on finding Attachment 3A in a library, on finding Document 3 online at IEEE Xplore, I conclude that Document 3 is an authentic document and that Attachment 3A is an authentic copy of Document 3.



## **Public Accessibility**

61. Attachment 3C is the WorldCat record for Document 3 it shows that it is held by 789 libraries world-wide as of July 2018 and one of those 789 is the Linda Hall Library. Subject access is possible through subject headings: Elektronik; Elektrotechnik; and Institute of Electrical and Electronics Engineers.

62. Attachment 3D is the Linda Hall Library OPAC and MARC records for Document 3. The OPAC record documents that the Linda Hall Library holds Volume 75 (1987). This is indicated toward the bottom on the record under heading: Holdings Information – Issues: v51(1963) – 85(1997) indicates all volumes and issues are held from volume 51 - 1963 through volume 85 - 1997, thereby indicating that volume 75 – 1987 is held by the Linda Hall Library. Since there are no special notes in this record to indicate otherwise, I would expect that Document 3D was catalogued and sent to the stacks for public access within a week to ten days after receipt by the Linda Hall Library. Thus, based on the date stamp of October 23, 1987 included on Attachment 3A and the records shown in Attachment 3D, I conclude that Attachment 3A would have been available in the stacks of the Linda Hall Library, no later than early November 1987.

63. Attachment 3E is a true and accurate copy of a paper by Edward Ashford Lee and Jeffery C. Bier, “Architecture for Statically Scheduled Dataflow” published in the Journal of Parallel and Distributed Computing, Volume 10, issue 4,

December 1990: 333-348. On page 347, in References, citation number 21 is Document 3.

## **Conclusion**

64. Based on the evidence presented here, I conclude that Attachment 3A is an authentic copy of Document 3, which is an article by Jean-Luc Gaudiot titled “Data Driven Multicomputers in Digital Signal Processing,” 1987, IEEE, Proceedings, volume 75, number 9:1220-1234. Based on the Linda Hall Library’s records showing the receipt of this document on October 23, 1987, the lack of special notes in the document’s records, and the fact this this document was cited in a published article in December of 1990, it is my opinion that Document 3 was publicly accessible to and in actual use by ordinarily skilled researchers no later than early November 1987.

**Document 4: D. Roccatano et al., “Development of a Parallel Molecular Dynamics Code on SIMD Computers: Algorithm for Use of Pair List Criterion” Journal of Computational Chemistry. Volume 19, number 7 (May 1998): 685 – 694**

## **Authentication**

65. Document 4 is an article by D. Roccatano, et al., titled “Development of a Parallel Molecular Dynamics Code on SIMD Computers Algorithm for Use of Pair List Criterion” published in the Journal of Computational Chemistry, volume 19, number 7 (May 1998): 685-694.

66. Attachment 4A includes the cover of the May 1998 issue of the Journal of Computational Chemistry, the title page, table of contents and Document 4. This information was provided to me by the Wisconsin TechSearch (WTS) from the Linda Hall Library. On the first page of Document 4 is the ownership and date stamp of the Linda Hall Library indicating that it processed this issue into its collection on April 28, 1998.

67. Document 4 is available on-line through Wiley Online Library <https://onlinelibrary-wiley-com.ezproxy.lib.purdue.edu/doi/10.1002/%28SICI%291096-987X%28199805%2919%3A7%3C685%3A%3AAID-JCC1%3E3.0.CO%3B2-MIIEE>. Attachment 4B is a download I made from Wiley Online Library through Purdue University Libraries on June 30, 2018.

68. Attachment 4A is in a condition that creates no suspicion about its authenticity. Specifically, Attachment 4A is not missing any intermediate pages, the text on each page appears to flow seamlessly from one page to the next, and there are no visible alterations to the document. Attachment 4A was found within the custody of a library (the Linda Hall Library) – a place where, if authentic, it would likely be found. The text of the article is also consistent in Attachment 4A and 4B. I see no reason to question the authenticity of the cover and its stamp or other

preliminary pages in Attachment 4A. There are no missing pages and the text of the article flows seamlessly from one page to the next.

69. Based on finding Attachment 4A in a library, on finding Document 4 online at Wiley Online Library, I conclude that Document 4 is an authentic document and that Attachment 4A is an authentic copy of Document 4.

### **Public Accessibility**

70. Attachment 4C is the WorldCat record that shows Linda Hall Library as holding the Journal of Computational Chemistry. The record shows that the Journal of Computational Chemistry is held by 812 libraries world-wide as of July, 2018 among those libraries is the Linda Hall Library. Subject access is possible through subject headings: Chemistry – Data Processing – Periodicals; Chemistry – Methods; and Computer Simulation.

71. Attachment 4D is the Linda Hall Library OPAC record for Journal of Computational Chemistry. The OPAC record documents that the Linda Hall Library holds the 1988 volume in the “Holdings Information”: Issues: v.1:no. 1 (1980-Spring), v.4 (1983) – v.34, issue 31/32 (2013) indicates that the Linda Hall Library has holdings that are inclusive from v.4 (1983) – v.34 (2013) with no missing volumes or gaps, hence, volume 19, 1998 is held by the Linda Hall Library. Since there are no special notes in this record to indicate otherwise, I would expect that this document was catalogued and sent to the stacks for public access within a few

weeks of receipt by the Linda Hall Library. Thus, based on the date stamp of April 28, 1998 included with Attachment 4A and the records shown in Attachment 4D, I conclude that Attachment 4A would have been available in the stacks of the Linda Hall Library, no later than the middle of May 1998.

72. Attachment 4E is a paper by G. Chillemi et al., “The Role of Computer Technology in Applied Computational Chemical-Physics.” *Computer Physics Communication*, Volume 139, Issue 1 (1 September 2001): 1-19. On page 18, References, citation number 9 is for Document 4.

## **Conclusion**

73. Based on the evidence presented here, I conclude that Attachment 4A is an authentic copy of Document 4, which is an article by D. Roccatano et al., titled “Development of a Parallel Molecular Dynamics Code on SIMD Computers Algorithm for Use of Pair List Criterion” published in the *Journal of Computational Chemistry*, volume 19, number 7 (May 1998): 685-694. Based on the Linda Hall Library record showing the receipt of this document was on April 28, 1998, the lack of special notes in the document’s records, and the fact this this document was cited in a published article in September of 2001, it is my opinion that Document 4 was publicly accessible to and in actual use by ordinarily skilled researchers no later than the middle of May 1998.

**Document 5. Buell, Duncan, et al., Splash 2: FPGAs in a Custom Computing Machine. Vol. 9. Wiley-IEEE Computer Society Press, 1996.**

**Authentication**

74. Document 5 is a monograph edited by Duncan Buell and others with the title of “Splash 2: FPGAs in a Custom Computing Machine.” Volume 9 published by Wiley-IEEE Computer Society Press in 1996. Attachment 2A is a scan of Document 2 provided to me by the Wisconsin TechSearch (WTS) of the book held by the Massachusetts Institute of Technology (MIT) Library.

75. Attachment 5A includes a scan of the cover; inside front cover (with ownership stamp of Massachusetts Institute of Technology Libraries & inventory label indicating MIT Libraries); two blank facing introductory pages; introductory title page; trademark statement page; full title page identifying the editors as Duncan A. Buell, Jeffrey M. Arnold, and Walter J. Kleinfelder; Published by IEEE Computer Society Press, Los Alamitos, California; verso of the title page which provides the Cataloging in Publication (CIP) information and copyright date of 1996. The verso of the title page also has the stamp of the MIT Libraries and the check-in date of August 8, 1996. Written in pencil by hand is the call number: QA76.8. S65 B84 1996. The Contents section follows and the last page is the back cover.

76. Attachment 5B is and true and accurate copy of Document 5 that was provided to me by Counsel. Attachment 5B is in a condition that creates no suspicion about its authenticity. Specifically, Attachment 5B is not missing any

intermediate pages, the text on each page appears to flow seamlessly from one page to the next, and there are no visible alterations to the document. I have been informed by counsel that Attachment 5B was found within the custody of a library (the Library of Congress) – a place where, if authentic, it would likely be found.

77. Based on the information that Attachments 5A and 5B were found in a library and on finding library catalog and online index records for Document 5, I conclude that Document 5 is an authentic document and that Attachment 5B is an authentic copy of Document 5. After a thorough review of the cover, title page, verso of title page, and table of contents of Attachments 5A and 5B, I conclude that Attachment 5B is the same publication that was identified in Attachment 5A.

### **Public Accessibility**

78. Attachment 5C is a download of the WorldCat record for Document 2. As discussed above WorldCat is the most comprehensive catalog that documents the bibliographic information about books, journals, tech reports, dissertations, etc., that are available in libraries world-wide. Attachment 2B indicates that Document 2 was accessible by title: Splash 2: FPGAs in a custom computing machine; by authors: Duncan A. Buell, Jeffrey M. Arnold, and Walter J. Kleinfelder. Document 2 was also accessible by subject headings: Splash 2 (Computer); Electronic digital computers – Design and construction; Field programmable gate arrays.

79. The WorldCat record indicates, as of August 4, 2018, that 83 libraries around the world hold this monograph among these is the MIT Libraries.

80. Attachment 5D is a download from the Online Public Access Catalog (OPAC) of the MIT Libraries. This OPAC records indicates the book was available at the MIT Libraries with the call number: QA76.8.S65.B84 1996 in the Barker Engineering Library Stacks.

81. Attachment 5E is a download from the Online Public Access Catalog (OPAC) of the MIT Libraries that provides the MARC record for Document 5. The fields indicate through the MARC format title, author, publisher, etc. In the 910 field MIT has a unique practice of inserting the initials of the cataloger and the date the book was cataloged (this practice was observed by me during my years at MIT Libraries). The 910 field in this record reads: “tn961003” indicating the cataloger with the initials of tn cataloged Document 1 on October 3, 1996, in sequence from the date it was checked into the MIT Libraries by acquisitions as evidenced above by the date: August 8, 1996. The catalog information would have been entered into the GEAC Library Management System (LMS) then in use at the MIT Libraries, and would have been immediately available in the OPAC (called Barton). Thus, based on the date stamp of August 8, 1996 included with Attachment 5A and the records shown in Attachment 5E (cataloging date of October 3, 1996), I conclude that



Document 5 would have been available in the stacks of the MIT Libraries no later than the middle of October 1996.

### **Conclusion**

82. Based on the evidence presented here, I conclude that Attachments 5A and 5B are authentic copies of Document 5, which is a book by Duncan Buell and others with the title of “Splash 2: FPGAs in a Custom Computing Machine.” Volume 9 published by Wiley-IEEE Computer Society Press in 1996. Based on the MIT Library’s records showing the receipt of this document (through Attachment 5A) on August 8, 1996, the MARC records in Attachment 5E indicating that the book was catalogued on October 3, 1996, it is my opinion that Document 5 was publicly accessible to and in actual use by ordinarily skilled researchers no later than the middle of October 1996.

**Document 6: Yong-Jin Jeong and Wayne P. Burlson, “VLSI Array Algorithms and Architectures for RSA Modular Multiplication,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems. Volume 5, Number 2 (June 1997): 211-217.**

### **Authentication**

83. Document 6 is an article by Yong-Jin Jeong and Wayne B. Burlson titled “VLSI Array Algorithms and Architectures for RSA Modular Multiplication,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems. Volume 5, Number 2 (June 1997): 211-217.

84. Attachment 6A includes the June 1997 issue of IEEE Transactions on Very Large Scale Integration (VLSI) Systems. Volume 5, Number 2: cover/title-

page/table of contents, publication details, page 161, and Document 6 (pages 211 – 217). On page 161 is the stamp of the Library of Congress, Copyright Office and the date of receipt, September 22, 1997. Attachment 6A was provided to me by the Wisconsin TechSearch (WTS) from the Library of Congress.

85. Document 6 is available on-line through IEEE Xplore. Attachment 6B is a download from IEEE Xplore through Purdue University Libraries on August 29, 2018 at <https://ieeexplore-ieee-org.ezproxy.lib.purdue.edu/stamp/stamp.jsp?tp=&arnumber=585224>.

86. Attachment 6A is in a condition that creates no suspicion about its authenticity. Specifically, Attachment 6A is not missing any intermediate pages, the text on each page appears to flow seamlessly from one page to the next, and there are no visible alterations to the document. Attachment 6A was found within the custody of a library, the Library of Congress – a place where, if authentic, it would likely be found. The text of the article is also consistent in Attachment 6A and 6B. I see no reason to question the authenticity of the cover and its stamp or other preliminary pages in Attachment 6A. There are no missing pages and the text of the article flows seamlessly from one page to the next.

87. Based on finding Attachment 6A in a library, on finding Document 6 online at IEEE Xplore, I conclude that Document 6 is an authentic document and that Attachment 6A is an authentic copy of Document 6.

## **Public Accessibility**

88. Attachment 6C is the WorldCat record for IEEE Transactions on Very Large Scale Integration (VLSI) Systems. The WorldCat record shows that the IEEE Transactions on Very Large Scale Integration (VLSI) Systems is held by 530 libraries world-wide as of August 2018 among these is the Library of Congress.

89. Attachment 6D is the Library of Congress OPAC (public catalog) record for IEEE Transactions on Very Large Scale Integration (VLSI) Systems. The OPAC record documents that the Library of Congress subscribed from volume 1, no. 1 (March 1993) – to the present. Toward the end of the record, is a section titled “Older Receipts” in which it states holdings “v.1-v.14 (1993-2006:May). It also shows subject access through subject headings: Integrated circuits – Very large scale integration-Design and construction – Periodicals; Integrated circuits – Very large scale integration-Design and construction; and VLSI.

90. Attachment 6E is the MARC record available for IEEE Transactions on Very Large Scale Integration (VLSI) Systems from the Library of Congress. The MARC record indicates the “Unbound” (current) issues are shelved in the Newspaper & Current Periodical Reading Room (Madison LM133), when this issue was current (unbound) it would have been available in Madison LM133. Since there are no special notes in this record to indicate otherwise, I would expect that this document was checked-in and sent to the Newspaper & Current Periodical Reading

Room for public access within a few days of receipt by the Library of Congress. Thus, based on the date stamp of September 22, 1997 included with Attachment 6A and the records shown in Attachment 6D & 6E, I conclude that Attachment 6A would have been available in the Newspaper & Current Periodical Reading Room of the Library of Congress no later than the end of September 1997.

91. Attachment 6F is a paper by William L. Freking and Keshab K. Parhi, “Ring-Planarized Cylindrical Arrays with Application to Modular Multiplication,” Proceedings IEEE International Conference on Application-Specific Systems, Architectures, and Processors, Boston, MA, USA, (July 10-12, 2000): 497-506. On page 504, References, citation number 10 is for Document 6.

## **Conclusion**

92. Based on the evidence presented here, I conclude that Attachment 6A is an authentic copy of Document 6, the article by Yong-Jin Jeong and Wayne B. Burlson titled “VLSI Array Algorithms and Architectures for RSA Modular Multiplication,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems. Volume 5, Number 2 (June 1997): 211-217. Based on the Library of Congress record showing the receipt of this document on September 22, 1997, the lack of special notes in the document’s records, and the fact this this document was cited in a paper presented in July 2000, it is my opinion that Document 6 was

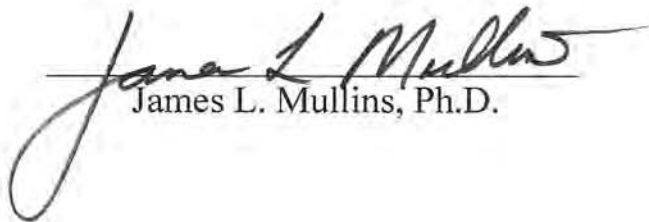
publicly accessible to and in actual use by ordinarily skilled researchers no later than the end of September 1997.

## **CONCLUSION**

93. I reserve the right to supplement my opinions in the future to respond to any arguments that Patent Owner or its expert(s) may raise and to take into account new information as it becomes available to me.

94. I declare that all statements made herein of my knowledge are true, and that all statements made on information and belief are believed to be true, and that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Executed this 30<sup>th</sup> day of August, 2018  
in Williamsburg, Virginia

  
James L. Mullins, Ph.D.

APPENDIX A

JAMES L. MULLINS

Prior Art Documentation Librarian Services, LLC

106 Berrow, Williamsburg, VA 23188

[jlmullins@priorartdoclib.com](mailto:jlmullins@priorartdoclib.com)

ph. 765 479 4956

**Experience:**

- 2018-present Dean of Libraries Emeritus and Esther Ellis Norton Professor Emeritus.
- 2011-2017 Dean of Libraries & Esther Ellis Norton Professor.
- 2004-2011 Dean of Libraries & Professor Purdue University, West Lafayette, IN.
- 2000-2004 Assistant/Associate Director for Administration, MIT Libraries, Massachusetts Institute of Technology, Cambridge, MA.
- 1996-2000 University Librarian & Director, Falvey Memorial Library. Villanova University, Villanova, PA.
- 1978-1996 Director of Library Services, Indiana University South Bend.
- 1974-1978 Associate Librarian, Indiana University Bloomington, School of Law.
- 1974-1978 Instructor/Catalog Librarian. Georgia Southern College (now University).

**Teaching Experience:**

- 1977-1996 Associate Professor (part-time), School of Library and Information Science, Indiana University. Subjects taught: Cataloging, Management, and Academic Librarianship.

**Education:**

The University of Iowa. Honors Bachelor of Arts in History, Religion and Political Science.

The University of Iowa. Master of Arts in Library Science.

Indiana University. Doctor of Philosophy. Concentration: Academic Library Administration. Emphasis: Legal Librarianship.

**Awards and Recognition:**

2017 Wilmeth Active Learning Center/Library of Engineering and Science, Grand Reading Room, was announced by President Mitch Daniels, Purdue University,

that it would be re-named the James L. Mullins Reading Room to honor his leadership and reputation in the academic library profession. September 2017.  
Portrait unveiled December 2017.

2017 Distinguished Alumnus Award by the School of Informatics and Computing, Indiana University, Bloomington. Given June 25, 2017.

2016 Hugh C. Atkinson Memorial Award, jointly sponsored by the four divisions of the American Library Association (ALA), June 27, 2016.

2015 ACRL Excellence in University Libraries Award, April 23, 2015.

Named Esther Ellis Norton Professor of Library Science by Purdue Trustees, December 11, 2011.

International Review Panel to evaluate the University of Pretoria Library, February 20 – 24, 2011. Pretoria, South Africa.

**Publications: (selected)**

A Purdue Icon: creation, life, and legacy, edited by James L. Mullins, Founder's Series, Purdue University Press, 138pp., August 2017.

“The policy and institutional framework.” In *Research Data Management, Practical Strategies for Information Professionals*, edited by Ray, J M. Purdue University Press, pp.25-44, 2014.

“DataCite: linking research to data sets and content.” In Benson, P and Silver, S. *What Editors Want: An Author's Guide to Scientific Journal Publishing*. University of Chicago Press, pp. 21-23, December 2012.

“Library Publishing Services: Strategies for Success,” with R. Crow, O. Ivins, A. Mower, C. Murray-Rust, J. Ogburn, D Nesdill, M. Newton, J. Speer, C. Watkinson. *Scholarly Publishing and Academic Resources Coalition (SPARC)*, version 2.0, March 2012.

“The Changing Definition and Role of Collections and Services in the University Research Library.” *Indiana Libraries*, Vol 31, Number 1 (2012), pp.18-24.

“Are MLS Graduates Being Prepared for the Changing and Emerging Roles that Librarians must now assume within Research Libraries?” *Journal of Library Administration*. Volume 52, Issue 1, 2012, p. 124-132



Baykoucheva, Svetla. What Do Libraries Have to Do with e-Science?: An Interview with James L. Mullins, Dean of Purdue University Libraries. *Chem. Inf. Bull.* [Online] 2011, 63 (1), 45-49.

<http://www.acscinf.org/publications/bulletin/63-1/mullins.php> (accessed Mar 16, 2011).

“The Challenges of e-Science Data-set Management and Scholarly Communication for Domain Sciences and Technology: a Role for Academic Libraries and Librarians,” chapter in, *The Digital Deluge: Can Libraries Cope with e-Science?*” Deanna B. Marcum and Gerald George, editors, Libraries Unlimited/Teacher Ideas Press, 2009. (a monograph publication of the combined proceedings of the KIT/CLIR proceedings).

“Bringing Librarianship to e-Science,” *College and Research Libraries*. vol. 70, no. 3, May 2009, editorial.

“The Librarian’s Role in e-Science” *Joho Kanri (Journal on Information Processing and Management)*, Japan Science and Technology Agency (formerly Japan Information Center of Science and Technology), Tokyo, Japan. Translated into Japanese by Taeko Kato. March 2008.

*The Challenge of e-Science Data-set Management to Domain Sciences and Engineering: a Role for Academic Libraries and Librarians*,” KIT (Kanazawa Institute of Technology)/CLIR (Council of Library and Information Resources) International Roundtable for Library and Information Science, July 5-6, 2007. Developments in e-science status quo and the challenge, The Japan Foundation, 2007.

“An Administrative Perspective,” Chapter 14, *Proven Strategies for Building an Information Literacy Program*, Susan Curzon and Lynn Lampert, editors, Neal-Schuman Publishers, Inc., New York, 2007. pp. 229-237.

*Library Management and Marketing in a Multicultural World*, proceedings of the IFLA Management and Marketing (M&M) Section, Shanghai, China, August 16-17, 2006, edited. K.G. Saur, Munchen, Germany, June 2007. 390 pp.

*Top Ten Assumptions for the Future of Academic Libraries and Librarians: a report from the ACRL Research Committee*, with Frank R. Allen and Jon R. Hufford. *College & Research Libraries*, April 2007, vol.68, no.4. pp.240-241, 246.

*To Stand the Test of Time: Long-term Stewardship of Digital Data Sets in Science and Engineering*. A report to the National Science Foundation from the ARL

Workshop on New Collaborative Relationships: the Role of Academic Libraries in the Digital Data Universe. September 26-27, 2006, Arlington, VA. p.141.  
<http://www.arl.org/bm~doc/digdatarpt.pdf>.

“Enabling Interaction and Quality in a Distributed Data DRIS,” *Enabling Interaction and Quality: Beyond the Hanseatic League*. 8th International Conference on Current Research Information Systems, with D. Scott Brandt and Michael Witt. Promoted by euro CRIS. Leuven University Press, 2006. pp.55-62. Editors: Anne Garns Steine Asserson and Eduard J. Simons.

“Standards for College Libraries, the final version approved January 2000,” prepared by the ACRL College Libraries Standards Committee (member), *C&RL News*, March 2000, p.175-182.

“Standards for College Libraries: a draft,” prepared by the ACRL College Libraries Section, Standards Committee (member), *C&RL News*, May 1999, p. 375-381.

“Statistical Measures of Usage of Web-based Resources,” *The Serials Librarian*, vol. 36, no. 1-2 (1999) p. 207-10.

“An Opportunity: Cooperation between the Library and Computer Services,” in *Building Partnerships: Computing and Library Professionals*. Edited by Anne G. Lipow and Sheila D. Creth. Berkeley and San Carlos, CA, Library Solutions Press, 1995. p. 69-70.

“Faculty Status of Librarians: A Comparative Study of Two Universities in the United Kingdom and How They Compare to the Association of College and Research Libraries Standards,” in *Academic Librarianship, Past, Present, and Future: a Festschrift in Honor of David Kaser*. Englewood, Colorado; Libraries Unlimited, 1989. p. 67-78. Review in: *College & Research Libraries*, vol. 51, no. 6. November 1990, p. 573-574.

### **Presentations: (representative)**

“How Long the Odyssey? Transitioning the Library and Librarians to Meet the Needs and Expectations of the 21st Century University,” David Kaser Lecture, School of Informatics & Computing, Indiana University, Bloomington, IN, November 16, 2015.

Presentation at University of Cape Town, Cape Town, South Africa, August 20, 2015.

“The Challenge of Discovering Science and Technology Information,” Moderator, International Federation of Library Associations (IFLA) Science and Technological Libraries Section Program, Cape Town, South Africa, August 18, 2015.

“An Odyssey in Data Management: Purdue University,” International Federation of Library Associations (IFLA) Research Data Management: Finding Our Role – A program of the Research Data Alliance, Cape Town, South Africa, August 17, 2015.

Presentation at University of Pretoria, Pretoria, South Africa, August 11, 2015.

Co-Convener with Sarah Thomas, Harvard University, at the Harvard Purdue Symposium on Data Management, Harvard University, Cambridge, MA, June 15-18, 2015.

“Strategic Communication,” panel discussion on the Director’s role and perspective on library communications at Committee on Institutional Cooperation (CIC) Center for Library Initiatives (CLI) Annual Conference, University of Illinois Urbana-Champaign, May 20, 2015.

“Issues in Data Management,” panel discussion moderated by Catherine Woteki, United States Undersecretary for Research, Education & Economics at 20th Agriculture Network Information Collaborative (AgNIC) Annual Meeting in the National Agricultural Library, Beltsville, MD, May 6, 2015.

“Active learning/IMPACT & the Active Learning Center at Purdue University,” Florida Institute of Technology, Melbourne, FL, February 11, 2015.

“Science+art=creativity: libraries and the new collaborative thinking,” panel moderator, International Federation of Library Associations (IFLA) 80th General Conference and Assembly, Lyon, France, August 19, 2014.

“Purdue University The Active Learning Center—A new concept for a library,” Association of University Architects 59th Annual National Conference, University of Notre Dame, South Bend, IN, June 23, 2014.

“Big Data & Implications for Academic Libraries,” keynote speaker, Greater Western Library Alliance (GWLA) Cyber-infrastructure Conference, Kansas City, MO, May 28, 2014.

“Research Infrastructure,” panel moderator, Association of Research Libraries (ARL) 164th Membership Meeting, Ohio State University, Columbus, OH, May 7, 2014.

“An Eight Year Odyssey in Data Management: Purdue University,” International Association of Scientific and Technological University Libraries (IATUL) 2013 Workshop Research Data Management: Finding Our Role, University of Oxford, UK, December 2013.

“Purdue University Libraries & Press: from collaboration to integration,” Ithaka Sustainable Scholarship, The Evolving Digital Landscape: New Roles and Responsibilities in Higher Education, libraries as publishers, New York, New York, October 2013.

“Tsinghua and Purdue: Research Libraries for the 21st Century,” Tsinghua University, Tsinghua, China, August 2013.

“Purdue Publishing Experience in the Libraries Publishing Coalition,” Association of American University Presses Annual Meeting, Press-Library Coalition Panel, Boston, Massachusetts, June 21, 2013.

“Indiana University Librarians Day: Purdue University Libraries Ready for the 21st Century,” Indiana University Purdue University Indianapolis (IUPUI), June 7, 2013.

“Purdue University Libraries and Open Access; CNI Project Update,” Coalition for Networked Information, San Antonio, TX, April 5, 2013.

Memorial Resolution, honoring Joseph Brannon, to the Board of the Association of College & Research Libraries, Seattle, WA, January 2013.

“An overview of sustaining e-Science collaboration in an Academic Research Library—the Purdue experience,” Duraspace e-Science Institute webcast, October 17, 2012.

“The Role of Libraries in Data Curation, Access, and Preservation: an International Perspective,” Panel Moderator, 78th General Conference and Assembly, International Federation of Library Associations, Helsinki, Finland, August 15, 2012.

“21st Century Libraries,” moderator of First Plenary Session, International Association of Technological University Libraries 33rd Annual Conference, Singapore, June 4, 2012.

“Planning for New Buildings on Campus,” panel presenter, University of Calgary Building Symposium on Designing Libraries for the 21st Century, Calgary, Alberta, Canada, May 17, 2012.

“Data Management and e-Science, the Purdue Response.” Wiley-Blackwell Executive Seminar-2012, Washington, DC, March 23, 2012.

“An overview of Sustaining e-Science Collaboration in Academic Research Libraries and the Purdue Experience.” Leadership & Career Development Program Institute, Association of Research Libraries (ARL). Houston, TX, March 21, 2012.

“An overview of Data Activities at Purdue University in response to Data Management Requirements.” Coalition for Academic Scientific Computation (CASC). Arlington, VA, September 8, 2011.

“Getting on Track with Tenure,” Association of College and Research Libraries (ACRL) Research Program Committee. Washington, DC, June 26, 2011.

“Integration of the Press and Libraries Collaboration to Promote Scholarly Communication,” Association of Library Collections & Technical Services (ALCTS) Scholarly Communication Interest Group – American Library Association, New Orleans, Louisiana, June 25, 2011.

“Cooperation for improving access to scholarly communication,” with N. Lossau (Germany), C. Mazurek (Poland), J. Stokker (Australia), panel moderator and presenter, Second Plenary Session, International Association of Scientific and Technological University Libraries (IATUL) 32nd Conference 2011, Warsaw, Poland. May 29-June 2, 2011.

“Riding the Wave of Data,” STM Annual Spring Conference 2011. Trailblazing & transforming scholarly publishing 2011. Washington, D.C., April 28, 2011.

“Confronting old assumptions to assume new roles: physical and operational integration of the Press and Libraries at Purdue University,” keynote speaker, 2011 BioOne Publishers & Partners Meeting. Washington, D. C., April 22, 2011.

“Are MLS Graduates Being Prepared for the Changing and Emerging Roles that Librarians must now assume within Research Libraries?” University of Oklahoma Libraries Seminar, March 4, 2011, Oklahoma City, Oklahoma.

“The Future Role of University Librarians,” the University of Cape Town, South Africa, February 25, 2011.

“New Roles for Librarians: the Application of Library Science to Scientific/Technical Research – Purdue University – a case study. International Council for Science and Technology (ICSTI); Ottawa, Canada. June 9, 2009.

“Reinventing Science Librarianship: Models for the Future,” Association of Research Libraries / Coalition for Networked Information. October 16-17th, 2008, Arlington, VA. Moderator and convener of Data Curation: Issues and Challenges.

“Practical Implementation and Opportunities Created at Purdue University,” African Digital Curation Conference, Pretoria, South Africa, (live video transmission), February 12, 2008.

Keynote speaker. “*Scholarly Communication & Academe: The Winter of Our Discontent*,” XXVII Charleston Conference on Issues in Book and Serial Acquisition, Charleston, South Carolina. November 8, 2007.

Keynote speaker. “*Enabling Access to Scientific & Technical Data-sets in e-Science: a role for Library and Archival Sciences*,” Greater Western Library Alliance (GWLA), Tucson, Arizona. September 17, 2007. A meeting of library directors and vice presidents for research of member institutions.

“*The Challenge of e-Science Data-set Management to Domain Sciences and Engineering: a Role for Academic Libraries and Librarians*,” KIT (Kanazawa Institute of Technology)/CLIR (Council of Library and Information Resources) International Roundtable for Library and Information Science, July 5-6, 2007. Invited to participate by the Deputy Librarian of Congress.

International Association of Technological University Libraries (IATUL), Stockholm, Sweden. June 8, 2007. Invited paper, *Enabling International Access to Scientific Data-sets: creation of the Distributed Data Curation Center (D2C2)*.

“A New Collaboration for Librarians: The Principles of Library and Archival Sciences Applied to the Curation of Datasets,” Symposium of the Libraries and the College of Engineering, University of Louisville, April 6, 2007.



“Purdue University Libraries: Through Pre-eminent Innovation and Creativity, Meeting the Challenges of the Information Age,” Board of Trustees, Purdue University, February 15, 2007.

ARL Workshop on New Collaborative Relationships: The Role of Academic Libraries in the Digital Data Universe, September 26-27, 2006, Arlington, VA. Invited participant.

NARA and SDSC: A partnership. A panel before the National Science Foundation, June 27, 2006. Arlington, VA. Invited participant.

“Kaleidoscope of Scientific Literacy: fusing new connections,” with Diane Rein, American Library Association, Association of College and Research Libraries, Science & Technology Section, Annual Conference, New Orleans, June 26th, 2006.

“Leadership for Learning: Building a Culture of Teaching in Academic Libraries – an administrative perspective,” American Library Association, Association of College and Research Libraries, Instruction Section, Annual Conference, New Orleans, June 25th, 2006.

“Building an interdisciplinary research program in an academic library: how the Libraries’ associate dean for research makes a difference at Purdue University,” International Association of Technological University Libraries (IATUL), Porto, Portugal, May 23rd, 2006.

“Enabling Interaction and Quality in a Distributed Data DRIS,” *Enabling Interaction and Quality: Beyond the Hanseatic League*. 8th International Conference on Current Research Information Systems, with D. Scott Brandt and Michael Witt. Promoted by euro CRIS, Bergen, Norway, May 12th, 2006, Brandt and Witt presented in person

“Interdisciplinary Research,” with D. Scott Brandt, Coalition for Networked Information (CNI) Spring Meeting: Project Briefing, Washington, D.C., April 3rd, 2006.

“An Interview with Purdue’s James Mullins,” a podcast submitted by Matt Pasiewicz, on *Educause Connect*, [http://connect.educause.edu/James\\_L\\_Mullins\\_Interview\\_CNI\\_2005](http://connect.educause.edu/James_L_Mullins_Interview_CNI_2005).

“Managing Long-Lived Digital Data-sets and their Curation: Interdisciplinary Policy Issues,” Managing Digital Assets Forum, Association of Research Libraries (ARL), Washington, D.C., October 28th, 2005.

“The Odyssey of a Librarian.” Indiana Library Federation (ILF), District 2 Meeting, South Bend, Indiana. October 4th, 2005.

“New College Library Standards,” Standards Committee Presentation, ALA, Chicago, July 7, 2000.

SUNY Library Directors, Lake George, New York. “*The College Library Standards: a Tool for Assessment.*” April 5, 2000.

Tri-State College Library Association, *Finding You Have Talents You Never Knew You Had*, Penn State Great Valley, March 25, 2000.

*Using Web Statistics*, American Library Association, New Orleans, June 24, 1999.

Keynote speaker at the JSTOR Workshop, January 29, 30, 1999. University of Pennsylvania, Philadelphia, PA.

“The New Standards for Electronic Resources Statistics,” Society of Scholarly Publishers, Washington, D.C., September 17, 1998.

“Evaluating Online Resources: Now that you’ve got them what do you do?,” joint presenter with Chuck Hamaker, LSU, at the NASIG Conference, Boulder, Colorado. June 1998.

“What Employers Are Looking for in New Librarians?” Pennsylvania Library Association, Philadelphia. September 26, 1997.

“The Theory of Matrix Management” panel presentation of the Comparative Library Organization Committee of the Library Organization and Management Section of the Library Administration and Management Association, a division of the American Library Association, Annual Meeting, Chicago, June 24, 1990.

**Professional Involvement: (summary of recent emphasis)**

The focus for my professional involvement and research has moved recently toward managing massive data-sets. This has resulted in working with faculty in the sciences and technology to determine how librarians can collaborate in managing, curating, and preserving data-sets for future access and documentation. This has included various speaking opportunities as well as participation in



planning with the National Science Foundation (NSF) on ways in which librarians can be integrated more completely into the funded research process. Participation in the Kanazawa Institute of Technology/Council of Library Resources Roundtable was particularly rewarding and provided new opportunities to share with international colleagues the issues surrounding data-set management. I was the champion for the creation of the Distributed Data Curation Center (D2C2) at Purdue University (<http://d2c2.lib.purdue.edu/>).

Throughout my career, beginning with my dissertation, I have been actively involved with assessing and evaluating libraries. In the fall of 1999, I contacted twenty-two academic library directors to determine whether the need was also felt by others. The response was overwhelmingly affirmative. This resulted in a meeting at ALA Midwinter, January 2000. A formal meeting followed at Villanova University in April 2000. As convener, I helped to form the University Libraries Group (ULG), modeled after the Oberlin Group for college libraries. The ULG is made up of university libraries that support diverse wide-ranging programs through doctoral level and have a level of support that places them in the top tier of academic institutions. A few of the member libraries, along with Villanova, are William and Mary, Wake Forest, Lehigh, Carnegie-Mellon, Tufts, Marquette, Miami of Ohio, and Southern Methodist.

In 1994 appointed to the Standards Committee, College Section, Association of College and Research Libraries. During the next six years, the Committee concentrated on changing the focus of the standards from quantitative analysis of input and output factors to emphasis on assessment of the outcome. Culmination of the work was a re-issue of the *Standards for College Libraries* in 2000. The knowledge gained through my work experience enabled me to formulate the changes needed in the standards. This work allowed for close collaboration with accrediting agencies, both professional and regional.

During this same time another focus emerged, the impact of digital resources. Through my work on the JSTOR Statistics Task Force, standards were developed on the collection of use of electronic databases. This Standard was later adopted in 1998 by the International Consortium of Library Consortia (ICOLC).

In 2002, the American Library Association appointed me to serve as the liaison to the Marketing and Management Section of the International Federation of Library Associations (IFLA).

**Professional Service: (representative list)**

Nominations Committee, Association of Research Libraries (ARL), 2016.

Steering Committee, Scholarly Publishing and Academic Resources Coalition (SPARC), 2016 – 2017.

“Excellence in Library Services,” Chair, Review Team, University of Hong Kong, Hong Kong, August 24-27, 2015.

Chair, Management Advisory Board, 2015-2017; Member, Scientific Advisory Board, arXiv, Cornell University, 1/1/2013 – present.

Advisory Board for the Wayne State University School of Library and Information Science, July 2012 – present.

Advisory Board for Microsoft Academic Search, 2012 – 2015. Redmond, WA.

Transforming Research Libraries, a Strategic Direction Steering Committee of the Association of Research Libraries (ARL), 2012-2015.

Science and Technology section, representing ARL, International Federation of Library Associations (IFLA), Chair, 2013 – 2017; Member, 2011 to present.

Member of University of Pretoria, South Africa, Library Review Committee. August 2013.

Co-chair, Local Arrangements Planning Committee for 2013 Conference, Association of College and Research Libraries (ACRL), a division of the American Library Association (ALA).

Association of Research Libraries Leadership & Career Development Program Mentor, 2011-2017.

e-Science Task Force, Association of Research Libraries. July 2006 – present. Chair, October 2011 – October 2012.

Board of Directors, International Association of Technological University Libraries (IATUL). January 2008 – December 2014.

Midwest Collaborative for Library Services (MCLS); Board Member, October 2010 – December 2012.

Chair, Library Directors, Committee on Institutional Cooperation (CIC), July 2010 – June 2012.

Board of Directors, Association of Research Libraries (ARL); October 2008 – October 2011.

Scholarly Communication Steering Committee, Association of Research Libraries (ARL)

2008-2011.

Editorial Board, College and Research Libraries, Association of College and Research Libraries, American Library Association. January 2008 – December 2014.

Chair, Organizing Committee for IATUL Conference 2010, June 21-24, 2010, Purdue University, West Lafayette, Indiana/Chicago, Illinois.

Conference Planning Committee for National Conference of the Association of College and Research Libraries, 2009, Seattle, Washington.

Research Committee, Association of College and Research Libraries, ACRL, division of ALA. 2002-2007, chair, 2005-2007.

Association of Research Libraries, Search and Screen Committee, Executive Director. March – January 2008.

Center for Research Libraries, Board of Directors. April 2006 – April 2012.

Academic Libraries of Indiana, Board of Directors, 2004 – present. Vice-president, 2005-2007. President, 2007- 2009.

ALA Representative to the International Federation of Library Associations (IFLA), Marketing and Management (M&M) Section, initial term 2003-2007, re-appointed for second term, 2007-2011.

Invited to represent Research Libraries at the ACRL/3M Wonewok Retreat to assess Marketing of Academic Libraries, October 2002.

Hugh A. Atkinson Award Committee, LAMA Representative, ALA, 2001-2005.

Program Committee, Library Administrators and Management Association (LAMA), a division of ALA. 1996-2001.

ACRL, Standards and Accreditation Committee, a division of ALA. Liaison to RBMS Section of ACRL. 1997-2002.

Elected to the Executive Committee of LAMA, LOMS, a division of the American Library Association, 1998-2000. Nominated as Chair/Elect for 2003 – 2005.

Columbia University Press Advisory Committee. 1996 - 2000.

LITA/LAMA Conference Evaluation Committee, Pittsburgh, Pennsylvania, October 1996.

“New Learning Communities,” Coalition for Networked Information, Indianapolis. November 19-21, 1995. Facilitator for invitational, national conference committed to developing collaborative learning and teaching techniques, involving librarians.

Planning Committee-Evaluation. LITA/LAMA 1996 Conference, Pittsburgh. This first conference, to be held jointly between two divisions of ALA, will focus on new technologies within libraries.

Indiana Cooperative Library Services Authority (InCoLSA), elected to Executive Committee, April 1991, served as President in 1993-94. InCoLSA is a statewide network of academic, public, school and special libraries that supports library cooperation for cataloging, interlibrary loan, collection development and application of new technologies.

Governor’s Conference on Libraries and Information Services. Served on Planning Committee, Academic Libraries Representative, appointed by the Governor to represent academic libraries in Indiana, Chair, Finance Committee, April 1989-July 1991.

Indiana Library Endowment Foundation Board, 1984-92. Charter Member, 1984, President, 1988-1992. 2004-2005.

### **University Service: (summary)**

Served on search and screen committees for senior positions including chancellor, dean and directors; most recently I have been asked to serve on the search committee for the provost of Purdue University. At MIT service included the Library Council & appointment to the Administrative Council by President Vest, 2001-2003 & Member of the Faculty Committee on the Library System. At Purdue appointed by the President to the Search Committee for the Provost, October 2007 to May 2008; member of the Capital Projects Committee, and IT Operational Oversight Committee as senior academic dean, 2008-2014;

Global Council, Global Policy Institute, 2012 – 2016.

Academic Program Excellence and Rankings (APER) project team, 2014.

Representative of the Academic Deans on the Re-engineering Business Operations, Purdue University, 2016 –

Academic Deans Council chaired by Provost – 2004 – 2017.

University Promotion and Tenure Committee – 2006 – 2017.

“Outstanding Team Award, Electronic Reserve Project,” served as Chair, recognition awarded by the President of Villanova University to one team who made an outstanding contribution to the operations of the University, selected by a committee of administrators, faculty, and staff. Awarded September 9, 1999.

Nominated for the IUSB Lundquist Award, 1995 & 1996. The Lundquist award is given to faculty who have “exhibited excellence in teaching, scholarly or artistic achievement, and diversified relevant service...”

# Attachment 1A

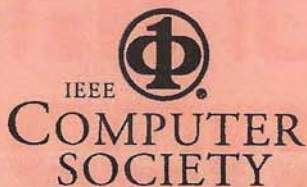


**IEEE Symposium on  
FPGAs FOR  
CUSTOM  
COMPUTING  
MACHINES**

**April 16–18, 1997  
Napa Valley, California**

Edited by Kenneth L. Pocek and Jeffrey Arnold

Sponsored by the IEEE Computer Society Technical Committee on Computer Architecture



# *FCCM'97*

---

---





Copyright © 1997 by The Institute of Electrical and Electronics Engineers, Inc.  
All rights reserved

*Copyright and Reprint Permissions:* Abstracting is permitted with credit to the source. Libraries may photocopy beyond the limits of US copyright law, for private use of patrons, those articles in this volume that carry a code at the bottom of the first page, provided that the per-copy fee indicated in the code is paid through the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923.

Other copying, reprint, or republication requests should be addressed to: IEEE Copyrights Manager, IEEE Service Center, 445 Hoes Lane, P.O. Box 133, Piscataway, NJ 08855-1331.

*The papers in this book comprise the proceedings of the meeting mentioned on the cover and title page. They reflect the authors' opinions and, in the interests of timely dissemination, are published as presented and without change. Their inclusion in this publication does not necessarily constitute endorsement by the editors, the IEEE Computer Society, or the Institute of Electrical and Electronics Engineers, Inc.*

IEEE Computer Society Order Number PR08159  
ISBN 0-8186-8159-4  
ISBN 0-8186-8160-8 (case)  
ISBN 0-8186-8161-6 (microfiche)  
IEEE Order Plan Catalog Number 97TB100186  
ISSN 1082-3409

JK7895  
IG56I35  
1987

*Additional copies may be ordered from:*

IEEE Computer Society  
Customer Service Center  
10662 Los Vaqueros Circle  
P.O. Box 3014  
Los Alamitos, CA 90720-1314  
Tel: +1-714-821-8380  
Fax: +1-714-821-4641  
E-mail: cs.books@computer.org

IEEE Service Center  
445 Hoes Lane  
P.O. Box 1331  
Piscataway, NJ 08855-1331  
Tel: +1-908-981-1393  
Fax: +1-908-981-9667  
mis.custserv@computer.org

IEEE Computer Society  
13, Avenue de l'Aquilon  
B-1200 Brussels  
BELGIUM  
Tel: +32-2-770-2198  
Fax: +32-2-770-8505  
euro.ofc@computer.org

IEEE Computer Society  
Ooshima Building  
2-19-1 Minami-Aoyama  
Minato-ku, Tokyo 107  
JAPAN  
Tel: +81-3-3408-3118  
Fax: +81-3-3408-3553  
tokyo.ofc@computer.org

Editorial production by Bob Werner

Cover art production Joe Daigle/Studio Productions

Printed in the United States of America by Technical Communication Services

IEEE  
COMPUTER  
SOCIETY



97-80098

# Table of Contents

## Symposium on Field-Programmable Custom Computing Machines — FCCM'97

<b>Introduction</b> .....	ix
<b>Program Committee</b> .....	x
<b>Session 1: Device Architecture</b>	
An FPGA Architecture for DRAM-based Systolic Computations..... <i>N. Margolus</i>	2
Garp: A MIPS Processor with a Reconfigurable Coprocessor..... <i>J. Hauser, J. Wawrzynek</i>	12
A Time-Multiplexed FPGA..... <i>S. Trimberger, D. Carberry, A. Johnson, J. Wong</i>	22
<b>Session 2: Communication Applications</b>	
An FPGA-Based Coprocessor for ATM Firewalls..... <i>J. McHenry, P. Dowd, T. Carrozzi, F. Pellegrino, W. Cocks</i>	30
A Wireless LAN Demodulator in a Pamette: Design and Experience..... <i>T. McDermott, P. Ryan, M. Shand, D. Skellern, T. Percival, N. Weste</i>	40
<b>Session 3: Run Time Reconfiguration</b>	
Incremental Reconfiguration for Pipelined Applications..... <i>H. Schmit</i>	47
Compilation Tools for Run-Time Reconfigurable Designs..... <i>W. Luk, N. Shirazi, P. Cheung</i>	56
A Dynamic Reconfiguration Run-Time System..... <i>J. Burns, A. Donlin, J. Hogg, S. Singh, M. de Wit</i>	66
<b>Session 4: Architectures for Run Time Reconfiguration</b>	
The Swappable Logic Unit: A Paradigm for Virtual Hardware..... <i>G. Brebner</i>	77



The Chimaera Reconfigurable Functional Unit.....	87
<i>S. Hauck, T. Fry, M. Hosler, J. Kao</i>	
<b>Session 5: Architecture</b>	
Computing Kernels Implemented with a Wormhole RTR CCM .....	98
<i>R. Bittner Jr., P. Athanas</i>	
Mapping Applications to the RaPiD Configurable Architecture .....	106
<i>C. Ebeling, D. Cronquist, P. Franklin, J. Secosky, S. Berg</i>	
Defect Tolerance on the Teramac Custom Computer .....	116
<i>B. Culbertson, R. Amerson, R. Carter, P. Kuekes, G. Snider</i>	
<b>Session 6: Performance</b>	
Systems Performance Measurement on PCI Pamette.....	125
<i>L. Moll, M. Shand</i>	
The RAW Benchmark Suite: Computation Structures for General Purpose Computing.....	134
<i>J. Babb, M. Frank, V. Lee, E. Waingold, R. Barua, M. Taylor, J. Kim, S. Devabhaktuni, A. Agarwal</i>	
<b>Session 7: Software Tools</b>	
Automated Field-Programmable Compute Accelerator Design using Partial Evaluation.....	145
<i>Q. Wang, D. Lewis</i>	
FPGA Synthesis on the XC6200 using IRIS and Trianus/Hades (Or, from Heaven to Hell and back again) .....	155
<i>R. Woods, S. Ludwig, J. Heron, D. Trainor, S. Gehring</i>	
High Level Compilation for Fine Grained FPGAs .....	165
<i>M. Gokhale, E. Gomersall</i>	
<b>Session 8: CAD Applications</b>	
Acceleration of an FPGA Router .....	175
<i>P. Chan, M. Schlag</i>	
Fault Simulation on Reconfigurable Hardware .....	182
<i>M. Abramovici, P. Menon</i>	

## Session 9: Image Processing Applications

Automated Target Recognition on SPLASH 2..... <i>M. Rencher, B. Hutchings</i>	192
Real-Time Stereo Vision on the PARTS Reconfigurable Computer..... <i>J. Woodfill, B. Von Herzen</i>	201
Increased FPGA Capacity Enables Scalable, Flexible CCMs: An Example from Image Processing..... <i>J. Greenbaum, M. Baxter</i>	211

## Session 10: Arithmetic Applications

Comparison of Arithmetic Architectures for Reed-Solomon Decoders in Reconfigurable Hardware..... <i>C. Paar, M. Rosner</i>	219
Implementation of Single Precision Floating Point Square Root on FPGAs..... <i>Y. Li, W. Chu</i>	226

## Poster Papers

Datapath-Oriented FPGA Mapping and Placement for Configurable Computing..... <i>T. Callahan, J. Waurzynek</i>	234
Mapping a Real-Time Video Algorithm to a Context-Switched FPGA..... <i>S. Kelem</i>	236
A Parallel Hardware Evolvable Computer POLYP..... <i>U. Tangen, L. Schulte, J. McCaskill</i>	238
Laser Defect Correction Applications to FPGA Based Custom Computers..... <i>G. Chapman, B. Dufort</i>	240
Speech Recognition HMM Training on Reconfigurable Parallel Processor..... <i>H. Yun, A. Smith, H. Silverman</i>	242
Efficient Implementation of the DCT on Custom Computers..... <i>N. Bergmann, Y. Chung, B. Gunther</i>	244
On Acceleration of the Check Tautology Logic Synthesis Algorithm using an FPGA-based Reconfigurable Coprocessor..... <i>J. Cong, J. Peck</i>	246
<b>Index of Authors</b> .....	249



# Mapping Applications to the RaPiD Configurable Architecture\*

Carl Ebeling, Darren C. Cronquist, Paul Franklin, Jason Secosky, and Stefan G. Berg

Department of Computer Science and Engineering  
University of Washington  
Box 352350  
Seattle, WA 98195-2350

## Abstract

The goal of the RaPiD (Reconfigurable Pipelined Datapath) architecture is to provide high performance configurable computing for a range of computationally-intensive applications that demand special-purpose hardware. This is accomplished by mapping the computation into a deep pipeline using a configurable array of coarse-grained computational units. A key feature of RaPiD is the combination of static and dynamic control. While the underlying computational pipelines are configured statically, a limited amount of dynamic control is provided which greatly increases the range and capability of applications that can be mapped to RaPiD. This paper illustrates this mapping and configuration for several important applications including a FIR filter, 2-D DCT, motion estimation, and parametric curve generation; it also shows how static and dynamic control are used to perform complex computations.

## 1 Introduction

Field-programmable custom computing machines have attracted a lot of attention recently because of their promise to deliver the high performance provided by special purpose hardware along with the flexibility of general purpose processors. Unfortunately, the promise of configurable computing has yet to be realized in spite of some very successful examples [1, 9]. There are two main reasons for this.

First, configurable computing platforms are currently implemented using commercial FPGAs. These FPGAs are necessarily very fine-grained so they can be used to implement arbitrary circuits, but the overhead of this generality exacts a high price in density and performance. Compared to general purpose processors (including digital signal processors), which use highly optimized functional units that operate in bit-parallel fashion on long data words, FPGAs are somewhat inefficient for performing logical operations and

even worse for ordinary arithmetic. FPGA-based computing has the advantage only on complex bit-oriented computations like count-ones, find-first-one, or complicated bit-level masking and filtering.

Second, programming an FPGA-based configurable computer is akin to designing an ASIC. The programmer either uses synthesis tools that deliver poor density and performance or designs the circuit manually, which requires both intimate knowledge of the FPGA architecture and substantial design time. Neither alternative is attractive, particularly for simple computations that can be described in a few lines of C code.

Our response to these two problems is RaPiD, a coarse-grained configurable architecture for constructing deep computational pipelines. RaPiD is aimed at regular, computation-intensive tasks like those found in digital signal processing (DSP). RaPiD provides a large number of ALUs, multipliers, registers and memory modules that can be configured into the appropriate pipelined datapath. The datapaths constructed in RaPiD are linear arrays of functional units communicating in mostly nearest-neighbor fashion. Systolic algorithms [4], for example, map very well into RaPiD datapaths, allowing us to take advantage of the considerable research on compiling computations to systolic arrays [5, 7]. However, RaPiD is not limited to implementing systolic algorithms; a pipeline can be constructed which comprises different computations at different stages and at different times.

We begin with an overview of the RaPiD architecture; for a more detailed description see [3]. We then give a general description of how computations map to RaPiD using a FIR filter as an example, and then present how the architectural features of RaPiD are used to perform more complex computations found in 2-D DCT, motion estimation, and parametric curve generation.

## 2 The RaPiD Datapath Architecture

RaPiD is a linear array of functional units which is configured to form a mostly linear computational pipeline. This array of functional units is divided into identical cells which are replicated to form a complete array. Figure 1 shows the cell used in RaPiD-1, the

\*This work was supported in part by the Defense Advanced Research Projects Agency under Contract DAAH04-94-G0272. D. Cronquist was supported in part by an IBM fellowship. P. Franklin was supported by an NSF fellowship.



first version of the RaPiD architecture. This cell comprises an integer multiplier, three integer ALUs, six general-purpose "datapath registers" and three small local memories. A typical single-chip RaPiD array would contain between 8 and 32 of these cells.

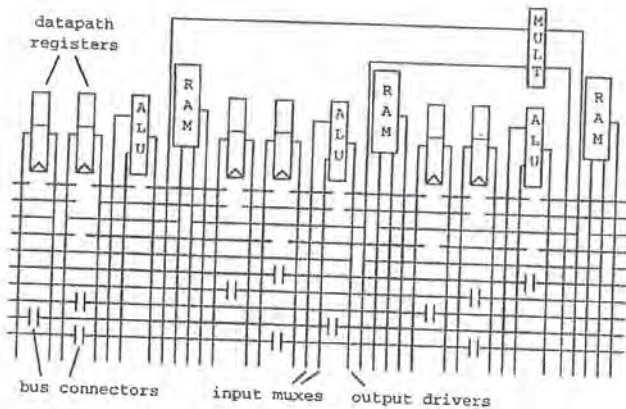


Figure 1: A basic RaPiD cell which is replicated left to right to form a complete chip. RaPiD-1 contains 16 cells similar to this one, with fourteen 16-bit buses.

## 2.1 Datapath Composition

The functional units are interconnected using a set of segmented buses that run the length of the datapath. The functional units use a  $n : 1$  multiplexer to select their data inputs from one of the  $n - 2$  bus segments in the adjacent tracks. The additional inputs provide fixed zero or feedback lines, which can be used to clear and hold register values, or to use an ALU as an accumulator. Each functional unit output includes optional registers to accommodate pipeline delays and a set of tristate drivers to drive their output onto one or more bus segments.

The buses in different tracks are segmented into different lengths to make the most efficient use of the connection resources. In some tracks, adjacent bus segments can be connected together by a bus connector as shown in Figure 1. This connection can be programmed in either direction via a unidirectional buffer or pipelined with up to three register delays, allowing data pipelines to be built in the bus structure itself.

RaPiD's ALUs perform the usual logical and arithmetic operations on one word of data. The ALUs can be chained for wide-integer operations. The multiplier inputs two single-word numbers and produces a double-word result, shifted by a statically programmed amount to maintain a given fixed-point representation. Both words of the result are available as separate outputs.

The datapath registers serve a variety of purposes in RaPiD. These registers can be used to store constants loaded during initialization and temporary val-

ues. They can be used as additional multiplexers to simplify control; like any functional unit, the registers can be disabled. They are also used while routing RaPiD applications to connect bus segments in different tracks and/or for additional pipeline delays.

In many applications, the data is partitioned into blocks which are loaded once, saved locally, reused as needed, and then discarded. The local memories provided in each cell of the datapath serve this purpose. Each local memory has a specialized datapath register used as an address register; one of the bus inputs to this address register is replaced by an incrementing feedback path. Like the SILOs found in the Philips VSP [8], this supports the common case of sequential memory accesses. More complex addressing patterns can be generated using registers and ALUs in the datapath.

Input and output data enter and exit RaPiD via I/O streams at each end of the datapath. Each stream contains a FIFO filled with data required or with results produced by the computation. The datapath explicitly reads from an input stream to obtain the next input data value and writes to an output stream to store a result.

External memory operations are carried out independent of the RaPiD array via three I/O streams by placing FIFOs between the array and a memory controller. In addition to carrying out the memory operations, the memory controller generates statically determined sequences of addresses for each stream. If the datapath reads a value from an empty FIFO or writes a value to a full FIFO, the datapath is stalled until the FIFO is ready.

## 2.2 Control Path

For the most part, the structure of a pipeline is statically configured. However, there are almost always some pipeline control signals that must be dynamic. For example, constants are loaded into datapath registers during initialization but then remain unchanged. The load signals of the datapath registers thus take on different values during initialization and computation. More complex examples include double-buffering the local memories and performing data-dependent calculations.

The control signals are thus divided into static control signals provided by configuration memory as in ordinary FPGAs, and control signals which can be dynamically programmed on every cycle. RaPiD is programmed for a particular application by first mapping the computation onto a datapath pipeline. The static programming bits are used to construct this pipeline and the dynamic programming bits are used to schedule the datapath operations over time. These dynamic control bits are provided by a small pipelined control path, not by the more typical local microprogrammed, SIMD, or VLIW control.



Dynamic control is implemented by inserting a few "context" bits each cycle into a pipelined control path that parallels the datapath. This context contains all the information required by the various pipeline stages to compute their dynamic control signals. The control path contains 1-bit segmented buses similar in structure to the datapath buses, as shown in Figure 2. (Signals which can be dynamic but do not need to change during a particular computation are connected to the static zero line.) Control values are inserted by a global pipeline controller at one end of the control path and are passed from stage to stage where they are applied to the appropriate control signals. Since applications generally use only a few dynamic control signals and use similar pipeline stages, the number of control signals in the control path is relatively small.

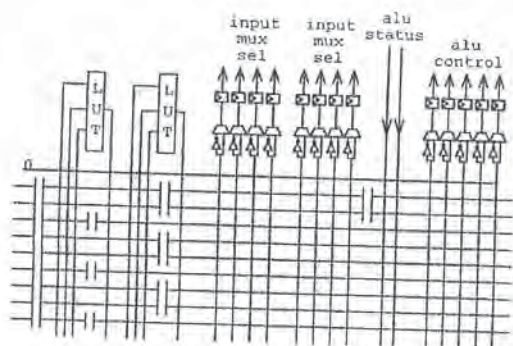


Figure 2: Dynamic control generation for part of a RaPiD cell; these control buses are one bit wide.

Each dynamic control signal is derived from the information contained in the control path. Usually the signal is simply connected to one of the bits in the control path, but in more complex cases lookup-tables embedded in the control path are used to compute the control signal based on more information including bits in the control path, status from ALUs in the datapath, or feedback when implementing simple FSM controllers. The generation of dynamic control is illustrated in detail in the applications that follow.

### 2.3 RaPiD-1 Design Features

Most of the design and layout of the RaPiD-1 chip, the first implementation of the RaPiD architecture, is complete. This section presents those details of RaPiD-1 useful in understanding the performance results discussed for each application presented in the following sections.

RaPiD-1's datapath is based on 16-bit fixed-point integers; to accommodate this, the multipliers can be statically programmed to shift their 32-bit output arbitrarily. Each RaPiD-1 cell contains three ALUs, one multipliers, and three 32-word local memories. Fourteen tracks are provided for the segmented data

buses, which are supplemented by the zero and feedback inputs available to each functional input. The 16 cells each have the functional units shown in Figure 1, in addition to control logic and up to 15 control buses. The RaPiD-1 array is designed to be clocked at 100MHz, and reconfiguration time for the array is conservatively estimated to be 2000 cycles.

## 3 Programming Model

Mapping applications to RaPiD involves designing the underlying datapath and providing the dynamic control required for the different parts of the computation. The control design can be complicated because control signals are generated at different times and travel at different rates. We have designed the RaPiD B programming language to accommodate these control patterns. Our RaPiD B compiler which produces a placed and routed implementation along with the dynamic control program is nearly complete. This section first describes a FIR (Finite Impulse Response) filter, a simple application useful for illustrating some of the basic features of RaPiD. It then briefly presents the timing models used by RaPiD B and by the remainder of this paper.

### 3.1 FIR Filter Computation

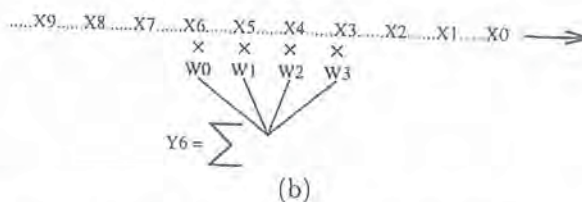
Digital FIR filters are used in many signal processing applications, typically for eliminating unwanted frequency components from a signal. Figure 3a gives a specification for a FIR filter with  $NumTaps$  taps and  $NumX$  inputs. The filter weights are stored in the  $W$  array, the input in the  $X$  array, and the output in the  $Y$  array (starting at array location  $NumTaps - 1$ ). Figure 3b shows the entire computation required for a single output of a 4-tap FIR filter.

```

for i := NumTaps-1 to NumX-1
  Y[i] := 0
  for j := 0 to NumTaps-1
    Y[i] := Y[i] + X[i-j]*W[j]
  end
end
end

```

(a)



(b)

Figure 3: FIR filter. (a) Algorithm. (b) Computation for  $NumTaps=4$  and  $i=6$ .

The circuit in Figure 4a performs the entire computation for one output value in a single cycle; it is easily obtained by unrolling the inner loop of the program



in Figure 3a. Unfortunately, the circuit shown in Figure 4a has poor performance characteristics (note the combinational path through all of the adders, which scales linearly with the number of weights). A retimed version of this circuit is shown in Figure 4b; the retimed circuit performs substantially better than the original, particularly for larger computations.

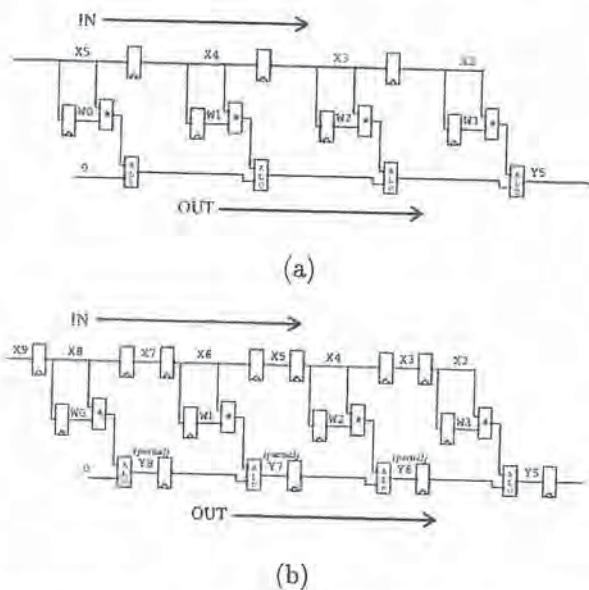


Figure 4: Schematic diagrams for four-tap FIR filter (a) as viewed in RaPiD B, grouping related computation and (b) as a high-performance pipelined implementation.

Specifying this retimed circuit directly is difficult because of the complexity of the relative timing of the internal data and control signals. It is much easier to specify the computation somewhat naively as in Figure 4a, knowing that retiming can transform it into a high-performance, pipelined circuit. This becomes particularly evident in circuits with more complicated control, and when more aggressive steps, such as using the pipeline stage available in RaPiD's multiplier, are needed to achieve the desired performance. Therefore, the RaPiD B compiler retimes the resulting netlist based on [6].

All of the applications presented in the following sections have been specified in a preliminary version of RaPiD B and simulated to validate the implementations described and the accompanying cycle count. For ease of explanation, the computations shown throughout this paper are shown before the full retiming performed by the RaPiD B compiler. A preliminary version of the RaPiD B toolset is nearly complete, including compilation, retiming, control synthesis, and full placement and routing of the resulting RaPiD circuit.

## 4 FIR Filter Implementation

### 4.1 Simple Case

As with most applications, there are a variety of ways to map a FIR filter to RaPiD. The choice of mapping is driven by the parameters of both the RaPiD array and the application. For example, if the number of taps is less than the number of RaPiD multipliers, then each multiplier is assigned to multiply a specific weight. The weights are first preloaded into datapath registers whose outputs drive the input of a specific multiplier. Pipeline registers are used to stream the  $X$  inputs and  $Y$  outputs. Since each  $Y$  output must see  $NumTaps$  inputs, the  $X$  and  $Y$  buses must be pipelined at different rates. Figure 5a shows one cell of the FIR filter (several stages are shown in Figure 4b) with the  $X$  input bus doubly pipelined and the  $Y$  input bus singly pipelined.

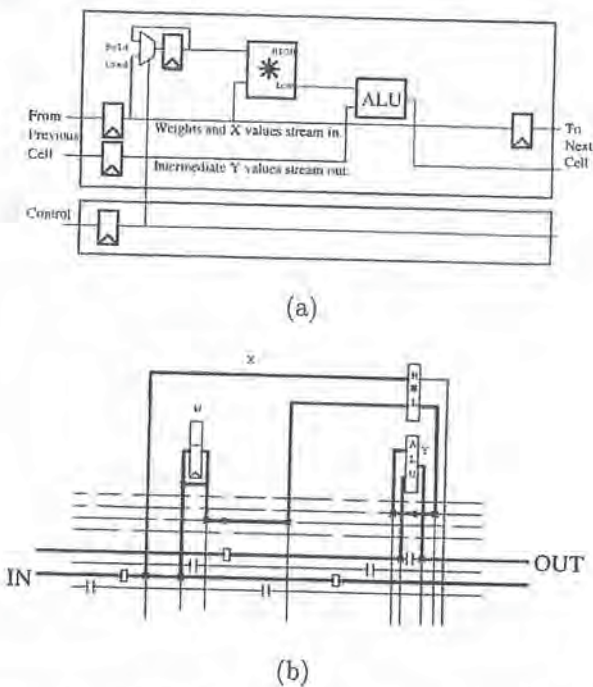


Figure 5: (a) Netlist for one cell of the simple FIR filter. (b) One tap of the FIR filter mapped to the RaPiD array (this is replicated to form more taps).

This implementation maps easily to the RaPiD array, as shown for one tap in Figure 5b. For clarity, all unused functional units are removed, and used buses are highlighted. The bus connectors from Figure 1 are left open to represent no connection and boxed to represent a register. The control for this mapping consists of two phases of execution: loading the weights and computing the output results. In the first phase, the weights are sent down the  $IN$  double pipeline along with a singly pipelined control bit which connects the



input of each datapath register to the  $IN$  bus. When the final weight is inserted, the control bit is switched, and the input is connected to the feedback line. Since the control bit travels twice as fast as the weights, each datapath register will hold a unique weight. No special signals are required to begin the computation; the second phase implicitly starts when the control bit goes low.

#### 4.2 Increasing the Number of Taps

If the number of taps exceeds the number of RaPiD multipliers, the multipliers must be time-shared between several taps. This can be achieved in RaPiD by using a local memory to store several weights per stage. Figure 6 shows our implementation for this mapping. Unlike the simple case, we make the arbitrary choice for doubly pipelining the  $Y$  output values and singly pipelining the  $X$  input values.

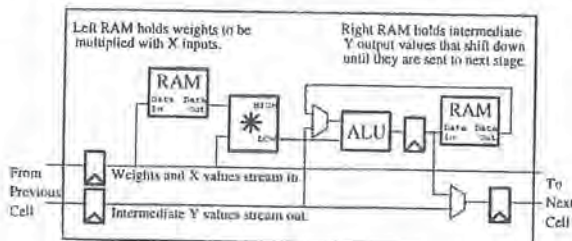


Figure 6: Netlist for one cell of extended FIR filter. The top pipelined bus streams in the  $X$  inputs (the weights during initialization) while the bottom bus streams out the intermediate  $Y$  values.

As a new  $X$  is read from external memory, the first stage replicates it and presents it to the input datapath for several cycles. Each stage can multiply this  $X$  by its weights in turn and add it to one of its stored intermediate values. At this point a new  $X$  value will be fetched from memory and the cycle repeats.

There are the same number of intermediate values as there are weights per stage. These intermediate values are stored in a second local memory. Let's examine the stage holding weights  $W_{55}$ ,  $W_{54}$ ,  $W_{53}$ , and  $W_{52}$  (four taps per stage). A new input value  $X_{20}$  appears on the input datapath. In four cycles the partial sums for  $Y_{75}$ ,  $Y_{74}$ ,  $Y_{73}$ , and  $Y_{72}$  will be computed. These are stored in that order in the local memory holding the intermediate values. At this point,  $X_{20}$  moves to the next pipeline stage followed by the intermediate value  $Y_{72}$ . The next input,  $X_{21}$ , appears on the input datapath along with the intermediate value  $Y_{76}$  from the previous stage. Now the partial sums for  $Y_{76}$ ,  $Y_{75}$ ,  $Y_{74}$ , and  $Y_{73}$  are computed.

#### 4.3 FIR Performance

When the number of taps is a multiple of 16 the weights can be partitioned evenly across the stages

and the allocated functional units are fully utilized. RaPiD-1 (Section 2.3) can therefore operate at very near its peak performance of 1.6 GOPS (where GOPS is a billion multiply-accumulates per second).

## 5 Discrete Cosine Transform

The discrete cosine transform (DCT) is used frequently in signal processing and graphics applications. For example, the 2-D DCT is used in JPEG/MPEG data compression to convert an image from the spatial domain to the frequency domain. A 2-D DCT can be decomposed into two sequential 1-D DCTs. We first describe how the 1-D DCT can be computed on RaPiD and then show how two 1-D DCTs can be composed to perform a 2-D DCT.

### 5.1 1-D DCT

An  $N$ -point 1-D DCT partitions an input vector  $A$  into  $N$ -element sub-vectors, and for each resulting sub-vector  $A_h$  computes

$$y_{hi} = \sum_{n=0}^{N-1} a_{hn} \cos \frac{\pi i}{2N} (2n+1) \quad (1)$$

for  $0 \leq i \leq N-1$ , where  $a_{hn}$  is the  $n$ -th element of sub-vector  $A_h$  (and the  $(hN+n)$ -th element of vector  $A$ ).<sup>1</sup> The  $N^2$  cosine terms are constant over all sub-vectors and hence can be read once as precomputed weights  $W$  where  $w_{ni} = \cos \frac{\pi i}{2N} (2n+1)$ . This reduces Equation 1 to

$$y_{hi} = \sum_{n=0}^{N-1} a_{hn} w_{ni}, \quad (2)$$

for  $0 \leq i \leq N-1$ . By viewing input vector  $A$  and weights  $W$  as matrices  $A$  and  $W$ , Equation 2 reduces to the matrix multiply  $Y = A \times W$ . Thus, to compute a 1-D DCT, RaPiD performs a matrix multiply.

To implement an 8 point 1-D DCT on an  $8 \times 8$  input matrix  $A$  (i.e. a 64-element vector), the entire  $8 \times 8$  weight matrix  $W$  is stored in RaPiD's local memories, one column per cell. Each cell of the resulting pipeline is configured as shown in Figure 7. The  $A$  matrix is passed through the array in row-major order. Within each cell, the local memory address is incremented each cycle, and a register accumulates the dot product of the stored column and the incoming row. When a cell receives the last element of a row, the resulting product is passed down an output pipeline, the address is cleared, and the cell is ready to compute the product of the next row on the next cycle. This effectively computes the matrix multiply of  $A \times W$ .

<sup>1</sup>To produce the final DCT result each  $y_{hi}$  must be multiplied by  $\sqrt{\frac{2}{N}} E_i$  where  $E_i = \frac{1}{\sqrt{2}}$  if  $i = 0$  and  $E_i = 1$  otherwise. For our purposes we ignore this scaling factor and focus on the computation of  $y_{hi}$ .



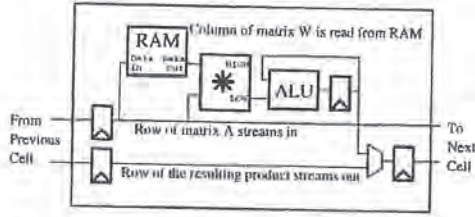


Figure 7: Netlist for one cell of a matrix multiply. The top pipelined bus streams in the  $A$  matrix (in row-major order) while the bottom bus streams out the resulting matrix product (also in row-major order). The top bus also streams the  $W$  columns into the local memories prior to the computation.

## 5.2 2-D DCT

An  $N \times N$  2-D DCT partitions an input matrix into sub-matrices of size  $N \times N$ , and for each resulting sub-matrix  $A$ , computes

$$y_{ji} = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} a_{mn} \cos \frac{\pi i}{2N} (2m+1) \cos \frac{\pi j}{2N} (2n+1) \quad (3)$$

for  $0 \leq i, j \leq N-1$ .<sup>2</sup> As with the 1-D DCT, Equation 3 is reduced using the  $N^2$  precomputed  $W$  weights, yielding

$$y_{ji} = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} a_{mn} w_{mi} w_{nj} \quad (4)$$

for  $0 \leq i, j \leq N-1$ . Extracting  $w_{mi}$  from the inner summation leaves

$$z_{mj} = \sum_{n=0}^{N-1} a_{mn} w_{nj}, \quad (5)$$

and thus

$$y_{ji} = \sum_{m=0}^{N-1} z_{mj} w_{mi} \quad (6)$$

for  $0 \leq i, j \leq N-1$ .

As seen in Equation 5 and Equation 6, both  $z_{mj}$  and  $y_{ji}$  are equivalent to  $N \times N$  matrix multiplies. However, since the  $z_{mj}$  values are produced in row-major order but required in column-major order, the results from the  $z_{mj}$  DCT must be transposed prior to computing  $y_{ji}$  as illustrated in Figure 8. In addition, since both input streams are read in row-major order, it might be desirable to produce row-major output (potentially reducing memory stalls), requiring yet another transform (i.e. output  $y_{ij}$  instead of  $y_{ji}$ ). The resulting computation is  $((A \times W)^T \times W)^T$ .

<sup>2</sup>To produce the final DCT result each  $y_{ji}$  must be multiplied by  $\frac{2}{N} E_i E_j$ . As with 1-D DCT, we ignore this scaling factor and focus on the computation of  $y_{ji}$ .

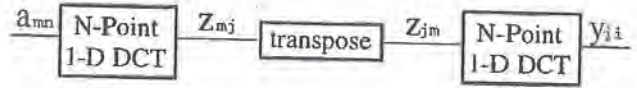


Figure 8: 2-D  $N \times N$  DCT

We show the implementation of an  $8 \times 8$  2-D DCT on a 16-cell RaPiD array. Consider an  $M \times N$  image and an  $8 \times 8$  weight matrix  $W$ . First, the image is divided into  $\frac{MN}{64}$  sub-images of size  $8 \times 8$ . The computation for each sub-image  $A$  is outlined in Figure 9.

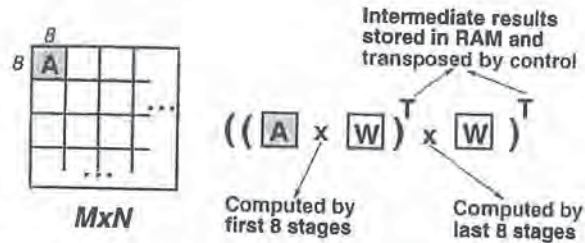


Figure 9: To compute 2-D DCT, an  $M \times N$  image is partitioned into  $8 \times 8$  sub-images. RaPiD computes each 1-D DCT by multiplying the sub-image by an  $8 \times 8$  weight matrix.

Since a 2-D DCT performs two multiplies by the same weight matrix,  $W$  is loaded only once: one column per cell in both the first 8 cells and last 8 cells. The transpose in between matrix multiplies is performed with two local memories per cell: one to store products of the current sub-image and the other to store the products of the *previous* sub-image. During the computation of the current sub-image, the transpose of the previous sub-image computation is passed to the next 8 cells. The datapath for one RaPiD cell of a 2-D DCT is shown in Figure 10.

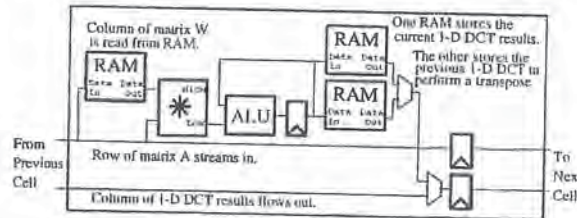


Figure 10: Netlist for one cell of 2-D DCT. The top pipelined bus streams in the  $A$  matrix while the bottom bus streams out resulting 1-D DCT, transposed. The top bus also streams the  $W$  columns into the local memories prior to the computation.

## 5.3 DCT Control

Prior to computation, a 2-D DCT must load the  $W$  matrix into the local memories, one column per stage.



To take advantage of pipelined control, weights are passed down a data-bus in *row-major* order, while a control signal, traveling twice as slow as the data, raises the write signal of the appropriate local memories. As a result, all weights of the DCT can be preloaded using a single control bus. Most RaPiD control signals fit into such a simple, pipelined model since an operation occurring in one RaPiD stage usually occurs in the next stage on the next cycle.

Sometimes control is required which does not fit into the simple, pipelined model. At the end of the first 1-D DCT computation, results are stored one column per stage. To flow these results out in *column-major* order (that is, perform the transpose), the first local memory must be completely emptied onto the output bus, followed by the second, third, etc. Hence, the "empty" control signal must stay on for eight consecutive cycles in the first stage, and then eight cycles in the second stage, etc. Possible solutions include dedicating a control bus to every stage or using one control bus with eight registers per stage. The solution requiring the fewest resources configures two buses and one 3-LUT per stage as a simple finite state machine, as shown in Figure 11.

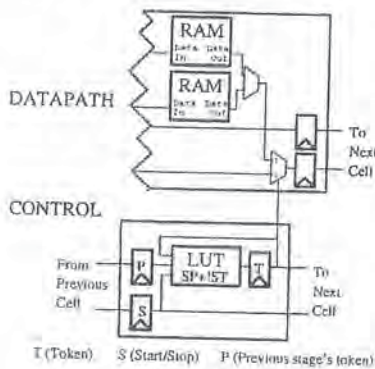


Figure 11: A simple state machine performs the transpose using two buses, one LUT, and three registers per stage.

Three control registers are used in the state machine:  $T$  is the token,  $S$  is the start/stop bit, and  $P$  is the previous stage's token delayed by a cycle. The LUT is configured as a multiplexer of  $P$  and  $T$  with select bit  $S$  (i.e.  $T = S \& P + !S \& T$ ). If  $S$  is low, the token is held; if  $S$  is high, the token is passed to the next stage. When a stage has a token, its results are emptied from a local memory onto the output bus. This operation repeats in each consecutive stage, effectively transposing the 1-D DCT results.

To initiate the transpose, the stream controller places a one into the first  $P$  register every 64 cycles and a one into the first  $S$  register every 8 cycles. Notice that the token hold length is solely determined by the frequency of the start/stop signal and does not affect the number of control buses, LUTs, or registers

needed. Thus, the size of this state-machine control is fixed no matter how long each stage must hold a token.

#### 5.4 DCT Performance

A 2-D DCT performs many consecutive  $8 \times 8$  matrix multiplies, allowing initialization, finalization, and re-configuration times to be small compared to the total computation performed. For example, RaPiD-1 (Section 2.3) incurs a setup overhead of only 0.5% to compute the 2-D DCT of a  $720 \times 576$  image. As a result, RaPiD-1 performs very close to its peak of 1.6 GOPS on 2-D DCT (where GOPS is a billion multiply accumulates per second).

### 6 Motion Estimation

Motion estimation is used in video data compression to reduce the amount of data required to represent a video frame. In most cases, objects do not move very much from one frame to the next. In motion estimation, a block in a frame is represented by the address of the most similar nearby block in the previous frame plus the differences between the two blocks. This section describes implementing motion estimation on RaPiD.

Motion estimation has few data dependencies, providing flexibility in the order of computations and greater parallelism. RaPiD favors computations that are not memory bound. The prodigious amount of computation and few memory accesses make motion estimation an ideal candidate for RaPiD.

To compute the motion estimation of an  $M \times N$  reference image, the image is divided into  $\frac{MN}{64} 8 \times 8$  reference blocks (RB). The reference blocks are compared with blocks of a prior video frame, the query image. For each reference block RaPiD computes the minimum absolute *block difference* (point-to-point difference) of all possible positions of the RB within a  $24 \times 24$  query window (QW) of the query image, as shown in Figure 12. The result is a vector which points to the RB yielding the minimum block difference.

#### 6.1 Motion Estimation Implementation

With a 16-stage RaPiD array we implement motion estimation using  $16 \times 16$  super reference blocks, which are comprised of four  $8 \times 8$  reference blocks, and  $32 \times 32$  super query windows. The super RB and a  $32 \times 16$  section of the query window are stored in RaPiD's local memories, one column per stage. This mapping yields the best reuse of RB and QW values for the available local memory. A stage of the resulting pipeline is shown in Figure 13.

The block difference between a super RB and super QW is computed row by row. For each row, a stage performs an absolute-difference and accumulates the



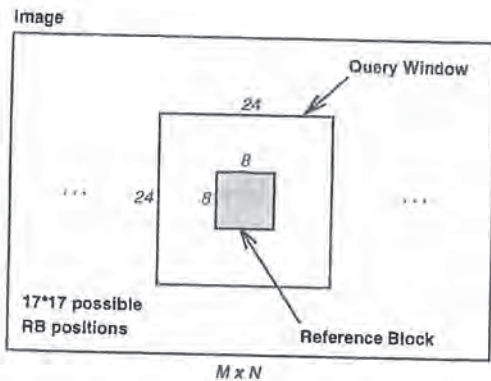


Figure 12: The image is partitioned into  $8 \times 8$  RBs. Motion estimation of the RB within a  $24 \times 24$  QW is determined by finding the minimum block difference for all positions of the RB within the QW.

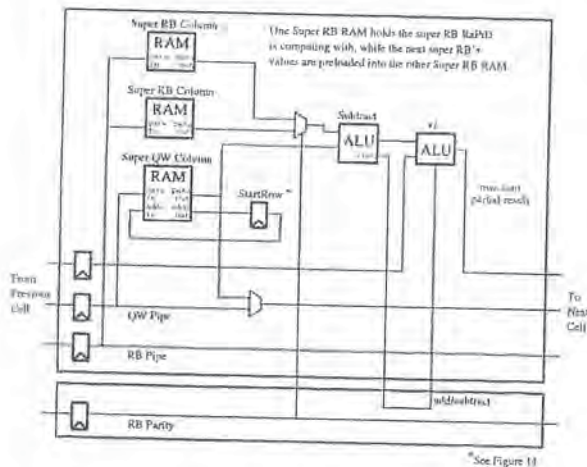


Figure 13: Cell configuration for motion estimation compute stages. 16-bit data and 1-bit control lines are drawn in separate boxes. To achieve an absolute-difference the sign bit of the Subtract ALU controls the function of the +/- ALU.

result with the absolute-difference of the prior stage. This operation happens in the same way as the FIR filter of Section 4. The last stage totals all of the row sums to produce the block difference and determines the minimum block difference for each RB of the super RB.

The netlist for motion estimation, presented in Figure 13, shows how two dynamic control lines control an ALU and super RB local memory selection. The absolute-difference-accumulate operation is implemented by controlling the function of the +/- ALU with the sign of the subtract ALU.

The local memory used for the super RB is double-buffered, with one local memory used for the current computation while the other is being preloaded with the next super RB. The parity control signal is used to determine which local memory to use for computation and which to use for preloading. The parity signal toggles when a motion vector for the current super RB is output.

## 6.2 Motion Estimation Data Flow

To obtain the most reuse of data we perform block differences in column-major order. That is, the super RB starts in the upper right corner of the super QW and proceeds down the rows before shifting left one column.

A left shift of the super RB is implemented by shifting the super QW columns right, to the next stage. When a super QW value is no longer needed, the value is shifted to the next stage and a new value is shifted in from the prior stage. The first stage gets new super QW values from the QW input stream.

Super QW values are reused between block differences by storing the address of the starting row of the super QW in the StartRow register (Figure 14). When a block difference completes, the QW column local memory address is set to StartRow and StartRow is incremented. StartRow is reset when the super RB is shifted left one column.

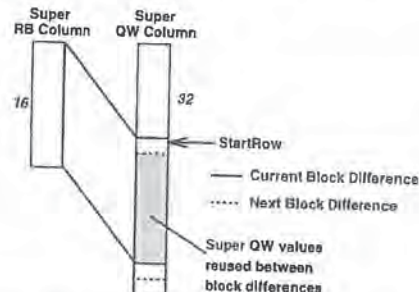


Figure 14: The super RB column shifts down through super QW column, performing a block difference at each step. StartRow is the address of the first row of the block difference.



Moving the super RB from right to left allows super QW values to be reused between sets of block differences. Figure 15 shows how the last columns of the current super QW are the first columns used in the super QW of the next super RB computation. This data motion removes the need to preload super QW values for the next set of block differences.

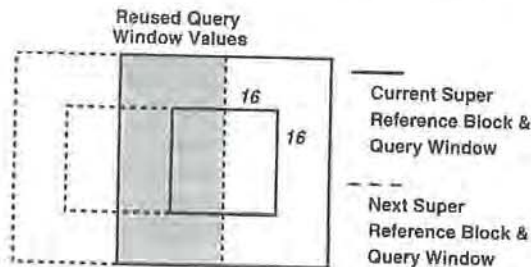


Figure 15: The last super QW columns used to compute motion estimation for a super RB are reused in the computation for the next RB.

The only time data loading stalls computation is the beginning of a row of super RBs. In this case the required super QW values were not used with the prior super RB and must be loaded. The next section shows that the cost is minor, being amortized over a long computation.

### 6.3 Motion Estimation Performance

Motion estimation is not a memory bound computation and with our implementation no memory stalls are encountered. The cycles not spent computing absolute-difference-accumulation operations are due to initialization, finalization, reconfiguration, and the loading of super QWs. For an image of size  $720 \times 576$ , using RaPiD-1, loading the super QW costs 18,432 cycles for motion estimation of one frame<sup>3</sup>. The overhead of loading and reconfiguration time take less than 0.03% of the total number of cycles. As a result, a RaPiD-1 array performs close to its peak speed of 1.6 GOPS (where GOPS is a billion absolute-difference-accumulates per second).

The speedup of motion estimation scales well as the data size grows and with future versions of RaPiD. As data size grows, the cycles used to load super QWs will grow linearly, while the cycles spent in computation grow with the square of the data size. Thus as the data size grows a larger percentage of cycles will be spent computing.

Future versions of RaPiD will have more stages and larger local memories per stage, increasing the number of RBs per super RB and thus the amount of parallelism. Typical images also use 8-bit data, allowing

<sup>3</sup>The super QW must be preloaded  $576/16$  times and a preload takes  $16 * 32$  cycles, resulting in 18,432 cycles.

us to double gauge RaPiD's 16-bit data path, gaining another factor of two in speedup.

## 7 Parametric Curve Generation

This section describes how arbitrary 2-D Bézier curves with four control points<sup>4</sup> can be computed by RaPiD using Apex, an architecture for generating a large class of parametric curves and surfaces [2]. Apex differs from the previous applications in that it maps a triangular data-flow onto RaPiD as shown in Figure 16. Each node in the tree performs a weighted average on the two inputs values and passes the result to the parent node. In symbolic form this is equivalent to

$$V_s(t) \leftarrow (1-t)V_{\text{left}} + tV_{\text{right}} = V_{\text{left}} + (V_{\text{right}} - V_{\text{left}})t$$

The root node produces a new point of the Bézier curve for each  $t$ . The nodes are mapped onto the RaPiD stages in the order indicated by the numbers inside the nodes (Figure 16). This particular mapping minimizes the communication between nodes.

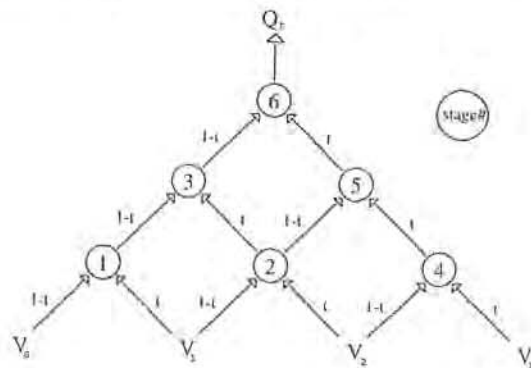


Figure 16: Data-flow graph for computation of the Bézier curve  $Q_t$  described by the control points  $V_i$ . Each node performs a weighted average (weights are edge labels) of its two inputs.

### 7.1 Apex Implementation

The algorithm can be split into initialization and computation. During initialization, the control points are loaded (e.g. stages 1, 2, and 4 in Figure 16) and a  $\Delta t$  increment is specified for  $t$ . Then the repetitive computation starts in which each node increments its private copy of  $t$  by  $\Delta t$  and performs the required computation. During the computational phase, no further external inputs are required.

Computing a 2-D Bézier curve produces two coordinate values per point. The two values can be computed independently. Since we only need six stages

<sup>4</sup>With very little additional effort this can be changed to Bézier curves of arbitrary dimension and with up to six control points on a 16-cell RaPiD array.



per coordinate value (see Figure 17), both can be computed in parallel using a total of twelve stages.

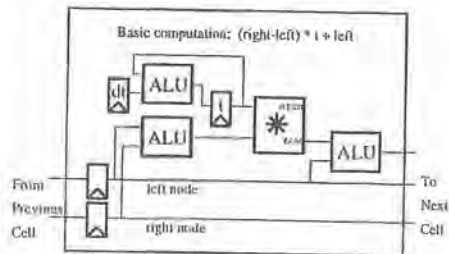


Figure 17: Netlist for one cell of Apex. The  $dt$  register is loaded from a datapath (not shown) before computation begins. Leaf nodes have two additional registers holding the constant control points.

The accuracy and resolution is limited by that  $t$  and  $\Delta t$  which in our current implementation is represented by a 16 bit register. The value  $t$  can be computed in the first stage using two registers (i.e. 32 bits) to substantially reduce the forward differencing errors. It would then propagate down the pipeline.

## 7.2 Apex Performance

Apex outputs a new point of the Bézier curve every cycle with relatively small initialization overhead. If we assume that 100 1000-point curves are displayed before reconfiguration is necessary, the setup overhead is only 0.2% for RaPiD-1 (Section 2.3) and it would perform at nearly 1.2 GOPS (where one OP is one weighted average). This is close to peak performance with the small loss in performance due to the fact that four cells are not used in the computation.

## 8 Conclusion and Future Directions

RaPiD represents an efficient configurable computing solution for regular computationally-intensive applications. In this paper, we have described how four different applications are mapped to the RaPiD array. These applications require a particular set of architectural features provided by RaPiD. We believe this feature set enables RaPiD to perform a wide range of different computations. By combining the appropriate amount of static and dynamic control, RaPiD achieves substantially reduced control overhead relative to FPGA-based and general-purpose processors. RaPiD is optimized for highly predictable and regular computations, reducing the control overhead. The assumption is that RaPiD will be integrated closely with a RISC engine on the same chip. The RISC would control the overall computational flow, performing the unstructured computations which it does best, while farming out the heavy-duty, brute-force computation to RaPiD.

Several challenges remain. The range of RaPiD applications needs to be extended, and integrated applications comprising different computations need to be investigated. The RaPiD B programming model needs to be evaluated and new compiler optimizations implemented. Finally, we would like to investigate how parallel language and compiling methods can be applied to programming RaPiD applications at a higher level.

## Acknowledgments

We would like to thank Larry McMurchie and Chris Fisher for their contributions to the RaPiD project.

## References

- [1] J. M. Arnold et al. The Splash 2 processor and applications. In *Proceedings IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pages 482-5. IEEE Comput. Soc. Press, 1993.
- [2] T. DeRose et al. Apex: two architectures for generating parametric curves and surfaces. *Visual Computer*, 5:264-76, 1989.
- [3] C. Ebeling, D. C. Cronquist, and P. Franklin. RaPiD—reconfigurable pipelined datapath. In R. Hartenstein and M. Glesner, editors, *6th International Workshop on Field-Programmable Logic and Compilers*, Lecture Notes in Computer Science, pages 126-135. Springer-Verlag, September 1996.
- [4] H. Kung. Let's design algorithms for VLSI systems. Technical Report CMU-CS-79-151, Carnegie-Mellon University, January 1979.
- [5] P. Lee and Z. M. Kedem. Synthesizing linear array algorithms from nested FOR loop algorithms. *IEEE Transactions on Computers*, 37(12):1578-98, 1988.
- [6] C. E. Leieron and J. B. Saxe. Retiming synchronous circuitry. *Algorithmica*, 6:5-35, 1991.
- [7] D. I. Moldovan and J. A. B. Fortes. Partitioning and mapping algorithms into fixed size systolic arrays. *IEEE Transactions on Computers*, C-35(1):1-12, 1986.
- [8] K. A. Vissers et al. Architecture and programming of two generations video signal processors. *Microprocessing & Microprogramming*, 41(5-6):373-90, 1995.
- [9] J. E. Vuillemin et al. Programmable active memories: reconfigurable systems come of age. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 4(1):56-69, 1996.

# Attachment 1B



CSDL Home (/csdl) » F (/csdl/proceedings/f/list.html) » FCCM (/csdl/proceedings/fccm/index.html) » 1997 (/csdl/proceedings/fccm/1997/index.html) » TABLE OF CONTENTS (/csdl/proceedings/fccm/1997/8159/00/index.html)

Search the CSDL



## Mapping applications to the RaPiD configurable architecture

(/csdl/proceedings/fccm/1997/8159/00/81590106-abs.html)

### Field-Programmable Custom Computing Machines, Annual IEEE Symposium on (1997)

Napa Valley, CA

Apr. 16, 1997 to Apr. 18, 1997

ISSN: 1082-3409

ISBN: 0-8186-8159-4

pp: 106

DOI Bookmark: <http://doi.ieeecomputersociety.org/10.1109/FPGA.1997.624610> (<http://doi.ieeecomputersociety.org/10.1109/FPGA.1997.624610>)

C. Ebeling (/web/search?cs\_search\_action=advancedsearch&searchOperation=exact&search-options=dl&searchText=C.+Ebeling) , Dept. of Comput. Sci. & Eng., Washington Univ., Seattle, WA, USA

D.C. Cronquist (/web/search?cs\_search\_action=advancedsearch&searchOperation=exact&search-options=dl&searchText=D.C.+Cronquist) , Dept. of Comput. Sci. & Eng., Washington Univ., Seattle, WA, USA

P. Franklin (/web/search?cs\_search\_action=advancedsearch&searchOperation=exact&search-options=dl&searchText=P.+Franklin) , Dept. of Comput. Sci. & Eng., Washington Univ., Seattle, WA, USA

J. Secosky (/web/search?cs\_search\_action=advancedsearch&searchOperation=exact&search-options=dl&searchText=J.+Secosky) , Dept. of Comput. Sci. & Eng., Washington Univ., Seattle, WA, USA

S.G. Berg (/web/search?cs\_search\_action=advancedsearch&searchOperation=exact&search-options=dl&searchText=S.G.+Berg) , Dept. of Comput. Sci. & Eng., Washington Univ., Seattle, WA, USA

### ABSTRACT

The goal of the RaPiD (Reconfigurable Pipelined Datapath) architecture is to provide high performance configurable computing for a range of computationally-intensive applications that demand special-purpose hardware. This is accomplished by mapping the computation into a deep pipeline using a configurable array of coarse-grained computational units. A key feature of RaPiD is the combination of static and dynamic control. While the underlying computational pipelines are configured statically, a limited amount of dynamic control is provided which greatly increases the range and capability of applications that can be mapped to RaPiD. This paper illustrates this mapping and configuration for several important applications including a FIR filter, 2-D DCT, motion estimation, and parametric curve generation; it also shows how static and dynamic control are used to perform complex computations.

### INDEX TERMS

motion estimation; RaPiD configurable architecture; mapping applications; reconfigurable pipelined datapath architecture; high performance configurable computing; special-purpose hardware; deep pipeline; coarse-grained computational units; computational pipelines; dynamic control; FIR filter; 2-D DCT; motion estimation; parametric curve generation; complex computations



## CITATION

S. Berg, D. Cronquist, J. Secosky, P. Franklin and C. Ebeling, "Mapping applications to the RaPiD configurable architecture," *Field-Programmable Custom Computing Machines, Annual IEEE Symposium on(FCCM)*, Napa Valley, CA, 1997, pp. 106.  
doi:10.1109/FPGA.1997.624610

## FULL ARTICLE

-  PDF
-  BUY
-  RSS Feed (</web/csd/rss-feeds>)
-  SUBSCRIBE (</web/csd/subscribe/>)

## CITATIONS

-  Plain Text (<https://cs-services.computer.org/csd/citation/proceedings/ascii/fccm/1997/8159/00/81590106>)
-  BibTex (<https://cs-services.computer.org/csd/citation/proceedings/bibtex/fccm/1997/8159/00/81590106>)
-  RIS (<https://cs-services.computer.org/csd/citation/proceedings/ris/fccm/1997/8159/00/81590106>)

## SEARCH

- Articles by C. Ebeling ([/web/search?cs\\_search\\_action=advancedsearch&searchOperation=exact&search-options=dl&searchText=C.+Ebeling](/web/search?cs_search_action=advancedsearch&searchOperation=exact&search-options=dl&searchText=C.+Ebeling))
- Articles by D.C. Cronquist ([/web/search?cs\\_search\\_action=advancedsearch&searchOperation=exact&search-options=dl&searchText=D.C.+Cronquist](/web/search?cs_search_action=advancedsearch&searchOperation=exact&search-options=dl&searchText=D.C.+Cronquist))
- Articles by P. Franklin ([/web/search?cs\\_search\\_action=advancedsearch&searchOperation=exact&search-options=dl&searchText=P.+Franklin](/web/search?cs_search_action=advancedsearch&searchOperation=exact&search-options=dl&searchText=P.+Franklin))
- Articles by J. Secosky ([/web/search?cs\\_search\\_action=advancedsearch&searchOperation=exact&search-options=dl&searchText=J.+Secosky](/web/search?cs_search_action=advancedsearch&searchOperation=exact&search-options=dl&searchText=J.+Secosky))
- Articles by S.G. Berg ([/web/search?cs\\_search\\_action=advancedsearch&searchOperation=exact&search-options=dl&searchText=S.G.+Berg](/web/search?cs_search_action=advancedsearch&searchOperation=exact&search-options=dl&searchText=S.G.+Berg))

## SHARE

- |  |  |   |  |
|--|--|---|--|
| Digg ( <a href="http://digg.com/submit?url=http%3A//doi.ieeecomputersociety.org/10.1109/FPGA.1997.624610&amp;title=Mapping%20applications%20to%20the">http://digg.com/submit?url=http%3A//doi.ieeecomputersociety.org/10.1109/FPGA.1997.624610&amp;title=Mapping applications to the</a> ) | Facebook ( <a href="https://www.facebook.com/sharer/sharer.php?u=http%3A//doi.ieeecomputersociety.org/10.1109/FPGA.1997.624610&amp;">https://www.facebook.com/sharer/sharer.php?u=http%3A//doi.ieeecomputersociety.org/10.1109/FPGA.1997.624610&amp;</a> ) | Google+ ( <a href="https://plus.google.com/share?url=http%3A//doi.ieeecomputersociety.org/10.1109/FPGA.1997.624610">https://plus.google.com/share?url=http%3A//doi.ieeecomputersociety.org/10.1109/FPGA.1997.624610</a> ) | LinkedIn ( <a href="https://www.linkedin.com/shareArticle?url=http%3A//doi.ieeecomputersociety.org/10.1109/FPGA.1997.624610&amp;">https://www.linkedin.com/shareArticle?url=http%3A//doi.ieeecomputersociety.org/10.1109/FPGA.1997.624610&amp;</a> ) |
|--|--|---|--|

RaPiD configurable architecture)	title=Mapping applications to the RaPiD configurable architecture)	title=Mapping applications to the RaPiD configurable architecture)	Reddit ( <a href="http://reddit.com/submit?url=http%3A//doi.ieeecomputersociety.org/10.1109/FPGA.1997.624610&amp;title=Mapping%20applications%20to%20the%20RaPiD%20configurable%20architecture">http://reddit.com/submit?url=http%3A//doi.ieeecomputersociety.org/10.1109/FPGA.1997.624610&amp;title=Mapping applications to the RaPiD configurable architecture)</a> )
----------------------------------	--	--	---

<a href="http://www.tumblr.com/share/link?url=http%3A//doi.ieeecomputersociety.org/10.1109/FPGA.1997.624610&amp;name=Mapping%20applications%20to%20the%20RaPiD%20configurable%20architecture">Tumblr (http://www.tumblr.com/share/link?url=http%3A//doi.ieeecomputersociety.org/10.1109/FPGA.1997.624610&amp;name=Mapping applications to the RaPiD configurable architecture)</a>	<a href="https://twitter.com/share?url=http%3A//doi.ieeecomputersociety.org/10.1109/FPGA.1997.624610&amp;text=Mapping%20applications%20to%20the%20RaPiD%20configurable%20architecture">Twitter (https://twitter.com/share?url=http%3A//doi.ieeecomputersociety.org/10.1109/FPGA.1997.624610&amp;text=Mapping applications to the RaPiD configurable architecture)</a>	<a href="http://www.stumbleupon.com/submit?url=http%3A//doi.ieeecomputersociety.org/10.1109/FPGA.1997.624610&amp;title=Mapping%20applications%20to%20the%20RaPiD%20configurable%20architecture">Stumbleupon (http://www.stumbleupon.com/submit?url=http%3A//doi.ieeecomputersociety.org/10.1109/FPGA.1997.624610&amp;title=Mapping applications to the RaPiD configurable architecture)</a>
--	---	---



# Mapping Applications to the RaPiD Configurable Architecture\*

Carl Ebeling, Darren C. Cronquist, Paul Franklin, Jason Secosky, and Stefan G. Berg

Department of Computer Science and Engineering  
University of Washington  
Box 352350  
Seattle, WA 98195-2350

## Abstract

The goal of the RaPiD (Reconfigurable Pipelined Datapath) architecture is to provide high performance configurable computing for a range of computationally-intensive applications that demand special-purpose hardware. This is accomplished by mapping the computation into a deep pipeline using a configurable array of coarse-grained computational units. A key feature of RaPiD is the combination of static and dynamic control. While the underlying computational pipelines are configured statically, a limited amount of dynamic control is provided which greatly increases the range and capability of applications that can be mapped to RaPiD. This paper illustrates this mapping and configuration for several important applications including a FIR filter, 2-D DCT, motion estimation, and parametric curve generation; it also shows how static and dynamic control are used to perform complex computations.

## 1 Introduction

Field-programmable custom computing machines have attracted a lot of attention recently because of their promise to deliver the high performance provided by special purpose hardware along with the flexibility of general purpose processors. Unfortunately, the promise of configurable computing has yet to be realized in spite of some very successful examples [1, 9]. There are two main reasons for this.

First, configurable computing platforms are currently implemented using commercial FPGAs. These FPGAs are necessarily very fine-grained so they can be used to implement arbitrary circuits, but the overhead of this generality exacts a high price in density and performance. Compared to general purpose processors (including digital signal processors), which use highly optimized functional units that operate in bit-parallel fashion on long data words, FPGAs are somewhat inefficient for performing logical operations and

even worse for ordinary arithmetic. FPGA-based computing has the advantage only on complex bit-oriented computations like count-ones, find-first-one, or complicated bit-level masking and filtering.

Second, programming an FPGA-based configurable computer is akin to designing an ASIC. The programmer either uses synthesis tools that deliver poor density and performance or designs the circuit manually, which requires both intimate knowledge of the FPGA architecture and substantial design time. Neither alternative is attractive, particularly for simple computations that can be described in a few lines of C code.

Our response to these two problems is RaPiD, a coarse-grained configurable architecture for constructing deep computational pipelines. RaPiD is aimed at regular, computation-intensive tasks like those found in digital signal processing (DSP). RaPiD provides a large number of ALUs, multipliers, registers and memory modules that can be configured into the appropriate pipelined datapath. The datapaths constructed in RaPiD are linear arrays of functional units communicating in mostly nearest-neighbor fashion. Systolic algorithms [4], for example, map very well into RaPiD datapaths, allowing us to take advantage of the considerable research on compiling computations to systolic arrays [5, 7]. However, RaPiD is not limited to implementing systolic algorithms; a pipeline can be constructed which comprises different computations at different stages and at different times.

We begin with an overview of the RaPiD architecture; for a more detailed description see [3]. We then give a general description of how computations map to RaPiD using a FIR filter as an example, and then present how the architectural features of RaPiD are used to perform more complex computations found in 2-D DCT, motion estimation, and parametric curve generation.

## 2 The RaPiD Datapath Architecture

RaPiD is a linear array of functional units which is configured to form a mostly linear computational pipeline. This array of functional units is divided into identical cells which are replicated to form a complete array. Figure 1 shows the cell used in RaPiD-1, the

\*This work was supported in part by the Defense Advanced Research Projects Agency under Contract DAAH04-94-G0272. D. Cronquist was supported in part by an IBM fellowship. P. Franklin was supported by an NSF fellowship.



first version of the RaPiD architecture. This cell comprises an integer multiplier, three integer ALUs, six general-purpose "datapath registers" and three small local memories. A typical single-chip RaPiD array would contain between 8 and 32 of these cells.

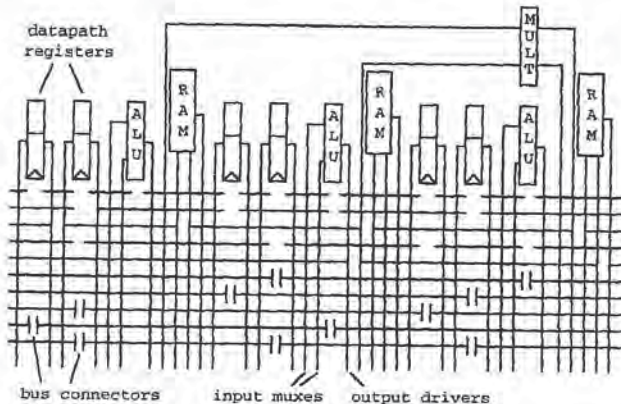


Figure 1: A basic RaPiD cell which is replicated left to right to form a complete chip. RaPiD-1 contains 16 cells similar to this one, with fourteen 16-bit buses.

## 2.1 Datapath Composition

The functional units are interconnected using a set of segmented buses that run the length of the datapath. The functional units use a  $n : 1$  multiplexer to select their data inputs from one of the  $n - 2$  bus segments in the adjacent tracks. The additional inputs provide fixed zero or feedback lines, which can be used to clear and hold register values, or to use an ALU as an accumulator. Each functional unit output includes optional registers to accommodate pipeline delays and a set of tristate drivers to drive their output onto one or more bus segments.

The buses in different tracks are segmented into different lengths to make the most efficient use of the connection resources. In some tracks, adjacent bus segments can be connected together by a bus connector as shown in Figure 1. This connection can be programmed in either direction via a unidirectional buffer or pipelined with up to three register delays, allowing data pipelines to be built in the bus structure itself.

RaPiD's ALUs perform the usual logical and arithmetic operations on one word of data. The ALUs can be chained for wide-integer operations. The multiplier inputs two single-word numbers and produces a double-word result, shifted by a statically programmed amount to maintain a given fixed-point representation. Both words of the result are available as separate outputs.

The datapath registers serve a variety of purposes in RaPiD. These registers can be used to store constants loaded during initialization and temporary val-

ues. They can be used as additional multiplexers to simplify control; like any functional unit, the registers can be disabled. They are also used while routing RaPiD applications to connect bus segments in different tracks and/or for additional pipeline delays.

In many applications, the data is partitioned into blocks which are loaded once, saved locally, reused as needed, and then discarded. The local memories provided in each cell of the datapath serve this purpose. Each local memory has a specialized datapath register used as an address register; one of the bus inputs to this address register is replaced by an incrementing feedback path. Like the SILOs found in the Philips VSP [8], this supports the common case of sequential memory accesses. More complex addressing patterns can be generated using registers and ALUs in the datapath.

Input and output data enter and exit RaPiD via I/O streams at each end of the datapath. Each stream contains a FIFO filled with data required or with results produced by the computation. The datapath explicitly reads from an input stream to obtain the next input data value and writes to an output stream to store a result.

External memory operations are carried out independent of the RaPiD array via three I/O streams by placing FIFOs between the array and a memory controller. In addition to carrying out the memory operations, the memory controller generates statically determined sequences of addresses for each stream. If the datapath reads a value from an empty FIFO or writes a value to a full FIFO, the datapath is stalled until the FIFO is ready.

## 2.2 Control Path

For the most part, the structure of a pipeline is statically configured. However, there are almost always some pipeline control signals that must be dynamic. For example, constants are loaded into datapath registers during initialization but then remain unchanged. The load signals of the datapath registers thus take on different values during initialization and computation. More complex examples include double-buffering the local memories and performing data-dependent calculations.

The control signals are thus divided into static control signals provided by configuration memory as in ordinary FPGAs, and control signals which can be dynamically programmed on every cycle. RaPiD is programmed for a particular application by first mapping the computation onto a datapath pipeline. The static programming bits are used to construct this pipeline and the dynamic programming bits are used to schedule the datapath operations over time. These dynamic control bits are provided by a small pipelined control path, not by the more typical local microprogrammed, SIMD, or VLIW control.



Dynamic control is implemented by inserting a few "context" bits each cycle into a pipelined control path that parallels the datapath. This context contains all the information required by the various pipeline stages to compute their dynamic control signals. The control path contains 1-bit segmented buses similar in structure to the datapath buses, as shown in Figure 2. (Signals which can be dynamic but do not need to change during a particular computation are connected to the static zero line.) Control values are inserted by a global pipeline controller at one end of the control path and are passed from stage to stage where they are applied to the appropriate control signals. Since applications generally use only a few dynamic control signals and use similar pipeline stages, the number of control signals in the control path is relatively small.

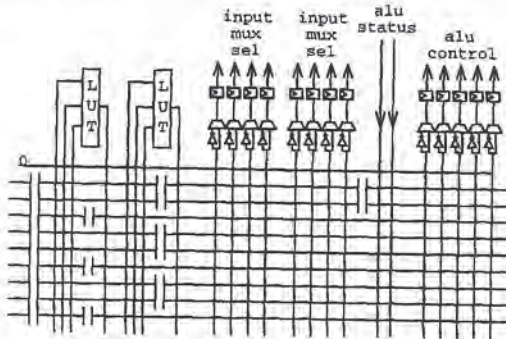


Figure 2: Dynamic control generation for part of a RaPiD cell; these control buses are one bit wide.

Each dynamic control signal is derived from the information contained in the control path. Usually the signal is simply connected to one of the bits in the control path, but in more complex cases lookup-tables embedded in the control path are used to compute the control signal based on more information including bits in the control path, status from ALUs in the datapath, or feedback when implementing simple FSM controllers. The generation of dynamic control is illustrated in detail in the applications that follow.

### 2.3 RaPiD-1 Design Features

Most of the design and layout of the RaPiD-1 chip, the first implementation of the RaPiD architecture, is complete. This section presents those details of RaPiD-1 useful in understanding the performance results discussed for each application presented in the following sections.

RaPiD-1's datapath is based on 16-bit fixed-point integers; to accommodate this, the multipliers can be statically programmed to shift their 32-bit output arbitrarily. Each RaPiD-1 cell contains three ALUs, one multiplier, and three 32-word local memories. Fourteen tracks are provided for the segmented data

buses, which are supplemented by the zero and feedback inputs available to each functional input. The 16 cells each have the functional units shown in Figure 1, in addition to control logic and up to 15 control buses. The RaPiD-1 array is designed to be clocked at 100MHz, and reconfiguration time for the array is conservatively estimated to be 2000 cycles.

## 3 Programming Model

Mapping applications to RaPiD involves designing the underlying datapath and providing the dynamic control required for the different parts of the computation. The control design can be complicated because control signals are generated at different times and travel at different rates. We have designed the RaPiD B programming language to accommodate these control patterns. Our RaPiD B compiler which produces a placed and routed implementation along with the dynamic control program is nearly complete. This section first describes a FIR (Finite Impulse Response) filter, a simple application useful for illustrating some of the basic features of RaPiD. It then briefly presents the timing models used by RaPiD B and by the remainder of this paper.

### 3.1 FIR Filter Computation

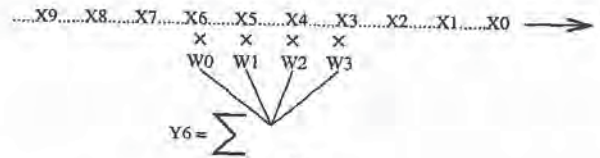
Digital FIR filters are used in many signal processing applications, typically for eliminating unwanted frequency components from a signal. Figure 3a gives a specification for a FIR filter with  $NumTaps$  taps and  $NumX$  inputs. The filter weights are stored in the  $W$  array, the input in the  $X$  array, and the output in the  $Y$  array (starting at array location  $NumTaps - 1$ ). Figure 3b shows the entire computation required for a single output of a 4-tap FIR filter.

```

for i := NumTaps-1 to NumX-1
  Y[i] := 0
  for j := 0 to NumTaps-1
    Y[i] := Y[i] + X[i-j]*W[j]
  end
end

```

(a)



(b)

Figure 3: FIR filter. (a) Algorithm. (b) Computation for  $NumTaps=4$  and  $i=6$ .

The circuit in Figure 4a performs the entire computation for one output value in a single cycle; it is easily obtained by unrolling the inner loop of the program



in Figure 3a. Unfortunately, the circuit shown in Figure 4a has poor performance characteristics (note the combinational path through all of the adders, which scales linearly with the number of weights). A retimed version of this circuit is shown in Figure 4b; the retimed circuit performs substantially better than the original, particularly for larger computations.

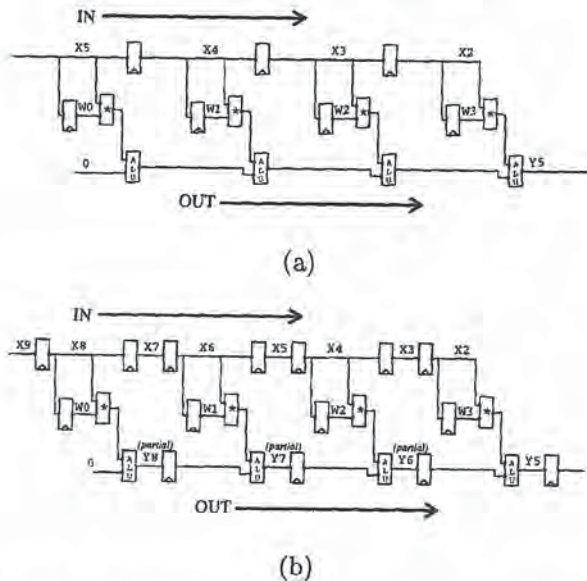


Figure 4: Schematic diagrams for four-tap FIR filter (a) as viewed in RaPiD B, grouping related computation and (b) as a high-performance pipelined implementation.

Specifying this retimed circuit directly is difficult because of the complexity of the relative timing of the internal data and control signals. It is much easier to specify the computation somewhat naively as in Figure 4a, knowing that retiming can transform it into a high-performance, pipelined circuit. This becomes particularly evident in circuits with more complicated control, and when more aggressive steps, such as using the pipeline stage available in RaPiD's multiplier, are needed to achieve the desired performance. Therefore, the RaPiD B compiler retimes the resulting netlist based on [6].

All of the applications presented in the following sections have been specified in a preliminary version of RaPiD B and simulated to validate the implementations described and the accompanying cycle count. For ease of explanation, the computations shown throughout this paper are shown before the full retiming performed by the RaPiD B compiler. A preliminary version of the RaPiD B toolset is nearly complete, including compilation, retiming, control synthesis, and full placement and routing of the resulting RaPiD circuit.

## 4 FIR Filter Implementation

### 4.1 Simple Case

As with most applications, there are a variety of ways to map a FIR filter to RaPiD. The choice of mapping is driven by the parameters of both the RaPiD array and the application. For example, if the number of taps is less than the number of RaPiD multipliers, then each multiplier is assigned to multiply a specific weight. The weights are first preloaded into datapath registers whose outputs drive the input of a specific multiplier. Pipeline registers are used to stream the  $X$  inputs and  $Y$  outputs. Since each  $Y$  output must see  $NumTaps$  inputs, the  $X$  and  $Y$  buses must be pipelined at different rates. Figure 5a shows one cell of the FIR filter (several stages are shown in Figure 4b) with the  $X$  input bus doubly pipelined and the  $Y$  input bus singly pipelined.

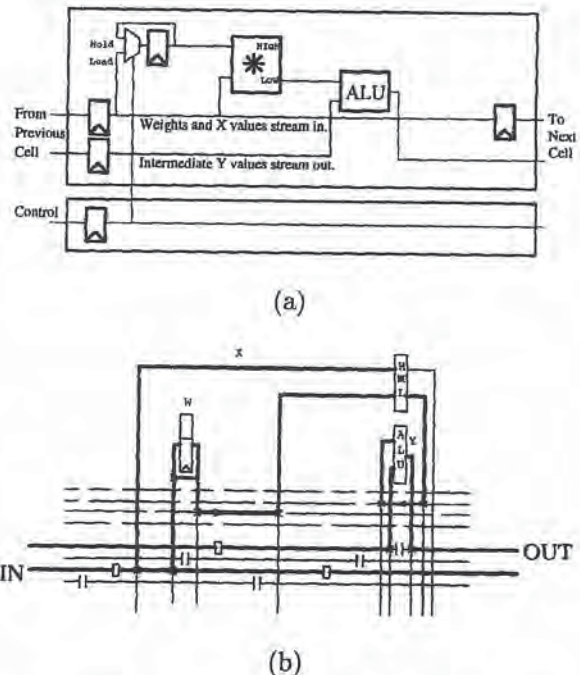


Figure 5: (a) Netlist for one cell of the simple FIR filter. (b) One tap of the FIR filter mapped to the RaPiD array (this is replicated to form more taps).

This implementation maps easily to the RaPiD array, as shown for one tap in Figure 5b. For clarity, all unused functional units are removed, and used buses are highlighted. The bus connectors from Figure 1 are left open to represent no connection and boxed to represent a register. The control for this mapping consists of two phases of execution: loading the weights and computing the output results. In the first phase, the weights are sent down the  $IN$  double pipeline along with a singly pipelined control bit which connects the



input of each datapath register to the  $IN$  bus. When the final weight is inserted, the control bit is switched, and the input is connected to the feedback line. Since the control bit travels twice as fast as the weights, each datapath register will hold a unique weight. No special signals are required to begin the computation; the second phase implicitly starts when the control bit goes low.

#### 4.2 Increasing the Number of Taps

If the number of taps exceeds the number of RaPiD multipliers, the multipliers must be time-shared between several taps. This can be achieved in RaPiD by using a local memory to store several weights per stage. Figure 6 shows our implementation for this mapping. Unlike the simple case, we make the arbitrary choice for doubly pipelining the  $Y$  output values and singly pipelining the  $X$  input values.

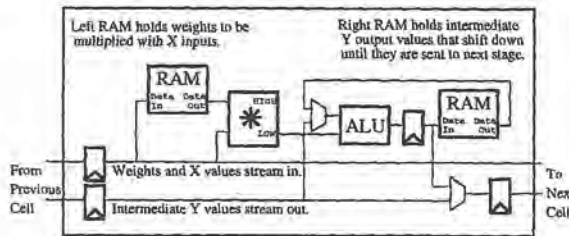


Figure 6: Netlist for one cell of extended FIR filter. The top pipelined bus streams in the  $X$  inputs (the weights during initialization) while the bottom bus streams out the intermediate  $Y$  values.

As a new  $X$  is read from external memory, the first stage replicates it and presents it to the input datapath for several cycles. Each stage can multiply this  $X$  by its weights in turn and add it to one of its stored intermediate values. At this point a new  $X$  value will be fetched from memory and the cycle repeats.

There are the same number of intermediate values as there are weights per stage. These intermediate values are stored in a second local memory. Let's examine the stage holding weights  $W_{55}$ ,  $W_{54}$ ,  $W_{53}$ , and  $W_{52}$  (four taps per stage). A new input value  $X_{20}$  appears on the input datapath. In four cycles the partial sums for  $Y_{75}$ ,  $Y_{74}$ ,  $Y_{73}$ , and  $Y_{72}$  will be computed. These are stored in that order in the local memory holding the intermediate values. At this point,  $X_{20}$  moves to the next pipeline stage followed by the intermediate value  $Y_{72}$ . The next input,  $X_{21}$ , appears on the input datapath along with the intermediate value  $Y_{76}$  from the previous stage. Now the partial sums for  $Y_{76}$ ,  $Y_{75}$ ,  $Y_{74}$ , and  $Y_{73}$  are computed.

#### 4.3 FIR Performance

When the number of taps is a multiple of 16 the weights can be partitioned evenly across the stages

and the allocated functional units are fully utilized. RaPiD-1 (Section 2.3) can therefore operate at very near its peak performance of 1.6 GOPS (where GOPS is a billion multiply-accumulates per second).

## 5 Discrete Cosine Transform

The discrete cosine transform (DCT) is used frequently in signal processing and graphics applications. For example, the 2-D DCT is used in JPEG/MPEG data compression to convert an image from the spatial domain to the frequency domain. A 2-D DCT can be decomposed into two sequential 1-D DCTs. We first describe how the 1-D DCT can be computed on RaPiD and then show how two 1-D DCTs can be composed to perform a 2-D DCT.

### 5.1 1-D DCT

An  $N$ -point 1-D DCT partitions an input vector  $A$  into  $N$ -element sub-vectors, and for each resulting sub-vector  $A_h$  computes

$$y_{hi} = \sum_{n=0}^{N-1} a_{hn} \cos \frac{\pi i}{2N} (2n+1) \quad (1)$$

for  $0 \leq i \leq N-1$ , where  $a_{hn}$  is the  $n$ -th element of sub-vector  $A_h$  (and the  $(hN+n)$ -th element of vector  $A$ ).<sup>1</sup> The  $N^2$  cosine terms are constant over all sub-vectors and hence can be read once as precomputed weights  $W$  where  $w_{ni} = \cos \frac{\pi i}{2N} (2n+1)$ . This reduces Equation 1 to

$$y_{hi} = \sum_{n=0}^{N-1} a_{hn} w_{ni}, \quad (2)$$

for  $0 \leq i \leq N-1$ . By viewing input vector  $A$  and weights  $W$  as matrices  $\mathbf{A}$  and  $\mathbf{W}$ , Equation 2 reduces to the matrix multiply  $\mathbf{Y} = \mathbf{A} \times \mathbf{W}$ . Thus, to compute a 1-D DCT, RaPiD performs a matrix multiply.

To implement an 8 point 1-D DCT on an  $8 \times 8$  input matrix  $\mathbf{A}$  (i.e. a 64-element vector), the entire  $8 \times 8$  weight matrix  $\mathbf{W}$  is stored in RaPiD's local memories, one column per cell. Each cell of the resulting pipeline is configured as shown in Figure 7. The  $\mathbf{A}$  matrix is passed through the array in row-major order. Within each cell, the local memory address is incremented each cycle, and a register accumulates the dot product of the stored column and the incoming row. When a cell receives the last element of a row, the resulting product is passed down an output pipeline, the address is cleared, and the cell is ready to compute the product of the next row on the next cycle. This effectively computes the matrix multiply of  $\mathbf{A} \times \mathbf{W}$ .

<sup>1</sup>To produce the final DCT result each  $y_{hi}$  must be multiplied by  $\sqrt{\frac{2}{N}} E_i$  where  $E_i = \frac{1}{\sqrt{2}}$  if  $i=0$  and  $E_i = 1$  otherwise. For our purposes we ignore this scaling factor and focus on the computation of  $y_{hi}$ .



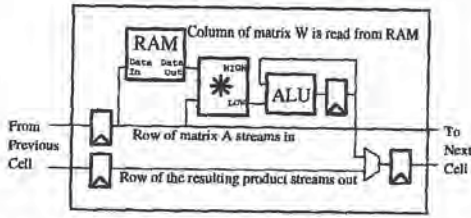


Figure 7: Netlist for one cell of a matrix multiply. The top pipelined bus streams in the  $A$  matrix (in row-major order) while the bottom bus streams out the resulting matrix product (also in row-major order). The top bus also streams the  $W$  columns into the local memories prior to the computation.

## 5.2 2-D DCT

An  $N \times N$  2-D DCT partitions an input matrix into sub-matrices of size  $N \times N$ , and for each resulting sub-matrix  $A$ , computes

$$y_{ji} = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} a_{mn} \cos \frac{\pi i}{2N} (2m+1) \cos \frac{\pi j}{2N} (2n+1) \quad (3)$$

for  $0 \leq i, j \leq N-1$ .<sup>2</sup> As with the 1-D DCT, Equation 3 is reduced using the  $N^2$  precomputed  $W$  weights, yielding

$$y_{ji} = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} a_{mn} w_{mi} w_{nj} \quad (4)$$

for  $0 \leq i, j \leq N-1$ . Extracting  $w_{mi}$  from the inner summation leaves

$$z_{mj} = \sum_{n=0}^{N-1} a_{mn} w_{nj}, \quad (5)$$

and thus

$$y_{ji} = \sum_{m=0}^{N-1} z_{mj} w_{mi} \quad (6)$$

for  $0 \leq i, j \leq N-1$ .

As seen in Equation 5 and Equation 6, both  $z_{mj}$  and  $y_{ji}$  are equivalent to  $N \times N$  matrix multiplies. However, since the  $z_{mj}$  values are produced in row-major order but required in column-major order, the results from the  $z_{mj}$  DCT must be transposed prior to computing  $y_{ji}$  as illustrated in Figure 8. In addition, since both input streams are read in row-major order, it might be desirable to produce row-major output (potentially reducing memory stalls), requiring yet another transform (i.e. output  $y_{ij}$  instead of  $y_{ji}$ ). The resulting computation is  $((A \times W)^T \times W)^T$ .

<sup>2</sup>To produce the final DCT result each  $y_{ji}$  must be multiplied by  $\frac{2}{N} E_i E_j$ . As with 1-D DCT, we ignore this scaling factor and focus on the computation of  $y_{ji}$ .

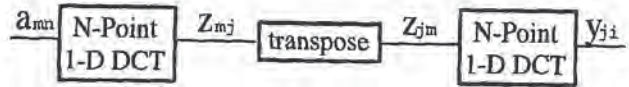


Figure 8: 2-D  $N \times N$  DCT

We show the implementation of an  $8 \times 8$  2-D DCT on a 16-cell RaPiD array. Consider an  $M \times N$  image and an  $8 \times 8$  weight matrix  $W$ . First, the image is divided into  $\frac{MN}{64}$  sub-images of size  $8 \times 8$ . The computation for each sub-image  $A$  is outlined in Figure 9.

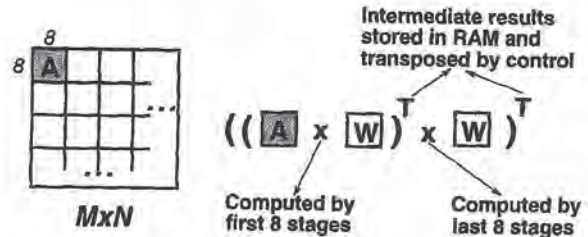


Figure 9: To compute 2-D DCT, an  $M \times N$  image is partitioned into  $8 \times 8$  sub-images. RaPiD computes each 1-D DCT by multiplying the sub-image by an  $8 \times 8$  weight matrix.

Since a 2-D DCT performs two multiplies by the same weight matrix,  $W$  is loaded only once: one column per cell in both the first 8 cells and last 8 cells. The transpose in between matrix multiplies is performed with two local memories per cell: one to store products of the current sub-image and the other to store the products of the *previous* sub-image. During the computation of the current sub-image, the transpose of the previous sub-image computation is passed to the next 8 cells. The datapath for one RaPiD cell of a 2-D DCT is shown in Figure 10.

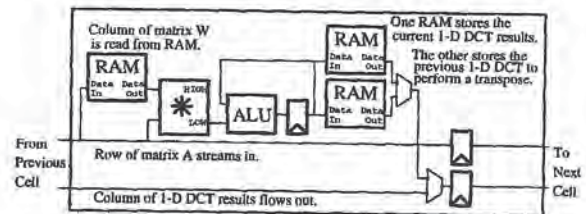


Figure 10: Netlist for one cell of 2-D DCT. The top pipelined bus streams in the  $A$  matrix while the bottom bus streams out resulting 1-D DCT, transposed. The top bus also streams the  $W$  columns into the local memories prior to the computation.

## 5.3 DCT Control

Prior to computation, a 2-D DCT must load the  $W$  matrix into the local memories, one column per stage.



To take advantage of pipelined control, weights are passed down a data-bus in *row-major* order, while a control signal, traveling twice as slow as the data, raises the write signal of the appropriate local memories. As a result, all weights of the DCT can be preloaded using a single control bus. Most RaPiD control signals fit into such a simple, pipelined model since an operation occurring in one RaPiD stage usually occurs in the next stage on the next cycle.

Sometimes control is required which does not fit into the simple, pipelined model. At the end of the first 1-D DCT computation, results are stored one column per stage. To flow these results out in *column-major* order (that is, perform the transpose), the first local memory must be completely emptied onto the output bus, followed by the second, third, etc. Hence, the “empty” control signal must stay on for eight consecutive cycles in the first stage, and then eight cycles in the second stage, etc. Possible solutions include dedicating a control bus to every stage or using one control bus with eight registers per stage. The solution requiring the fewest resources configures two buses and one 3-LUT per stage as a simple finite state machine, as shown in Figure 11.

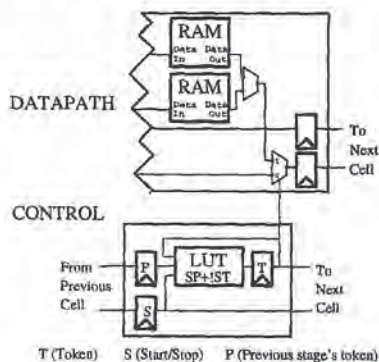


Figure 11: A simple state machine performs the transpose using two buses, one LUT, and three registers per stage.

Three control registers are used in the state machine:  $T$  is the token,  $S$  is the start/stop bit, and  $P$  is the previous stage’s token delayed by a cycle. The LUT is configured as a multiplexer of  $P$  and  $T$  with select bit  $S$  (i.e.  $T = S \& P + !S \& T$ ). If  $S$  is low, the token is held; if  $S$  is high, the token is passed to the next stage. When a stage has a token, its results are emptied from a local memory onto the output bus. This operation repeats in each consecutive stage, effectively transposing the 1-D DCT results.

To initiate the transpose, the stream controller places a one into the first  $P$  register every 64 cycles and a one into the first  $S$  register every 8 cycles. Notice that the token hold length is solely determined by the frequency of the start/stop signal and does not affect the number of control buses, LUTs, or registers

needed. Thus, the size of this state-machine control is fixed no matter how long each stage must hold a token.

#### 5.4 DCT Performance

A 2-D DCT performs many consecutive  $8 \times 8$  matrix multiplies, allowing initialization, finalization, and re-configuration times to be small compared to the total computation performed. For example, RaPiD-1 (Section 2.3) incurs a setup overhead of only 0.5% to compute the 2-D DCT of a  $720 \times 576$  image. As a result, RaPiD-1 performs very close to its peak of 1.6 GOPS on 2-D DCT (where GOPS is a billion multiply accumulates per second).

### 6 Motion Estimation

Motion estimation is used in video data compression to reduce the amount of data required to represent a video frame. In most cases, objects do not move very much from one frame to the next. In motion estimation, a block in a frame is represented by the address of the most similar nearby block in the previous frame plus the differences between the two blocks. This section describes implementing motion estimation on RaPiD.

Motion estimation has few data dependencies, providing flexibility in the order of computations and greater parallelism. RaPiD favors computations that are not memory bound. The prodigious amount of computation and few memory accesses make motion estimation an ideal candidate for RaPiD.

To compute the motion estimation of an  $M \times N$  reference image, the image is divided into  $\frac{MN}{64} 8 \times 8$  reference blocks (RB). The reference blocks are compared with blocks of a prior video frame, the query image. For each reference block RaPiD computes the minimum absolute block difference (point-to-point difference) of all possible positions of the RB within a  $24 \times 24$  query window (QW) of the query image, as shown in Figure 12. The result is a vector which points to the RB yielding the minimum block difference.

#### 6.1 Motion Estimation Implementation

With a 16-stage RaPiD array we implement motion estimation using  $16 \times 16$  super reference blocks, which are comprised of four  $8 \times 8$  reference blocks, and  $32 \times 32$  super query windows. The super RB and a  $32 \times 16$  section of the query window are stored in RaPiD’s local memories, one column per stage. This mapping yields the best reuse of RB and QW values for the available local memory. A stage of the resulting pipeline is shown in Figure 13.

The block difference between a super RB and super QW is computed row by row. For each row, a stage performs an absolute-difference and accumulates the



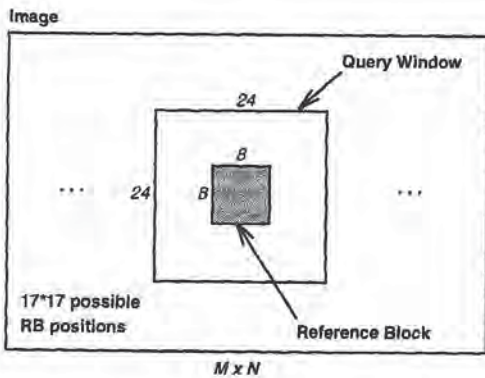


Figure 12: The image is partitioned into 8 x 8 RBs. Motion estimation of the RB within a 24 x 24 QW is determined by finding the minimum block difference for all positions of the RB within the QW.

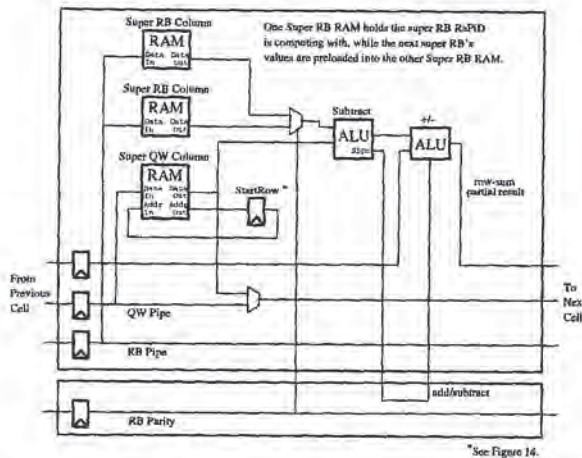


Figure 13: Cell configuration for motion estimation compute stages. 16-bit data and 1-bit control lines are drawn in separate boxes. To achieve an absolute-difference the sign bit of the Subtract ALU controls the function of the +/- ALU.

result with the absolute-difference of the prior stage. This operation happens in the same way as the FIR filter of Section 4. The last stage totals all of the row sums to produce the block difference and determines the minimum block difference for each RB of the super RB.

The netlist for motion estimation, presented in Figure 13, shows how two dynamic control lines control an ALU and super RB local memory selection. The absolute-difference-accumulate operation is implemented by controlling the function of the +/- ALU with the sign of the subtract ALU.

The local memory used for the super RB is double-buffered, with one local memory used for the current computation while the other is being preloaded with the next super RB. The parity control signal is used to determine which local memory to use for computation and which to use for preloading. The parity signal toggles when a motion vector for the current super RB is output.

## 6.2 Motion Estimation Data Flow

To obtain the most reuse of data we perform block differences in column-major order. That is, the super RB starts in the upper right corner of the super QW and proceeds down the rows before shifting left one column.

A left shift of the super RB is implemented by shifting the super QW columns right, to the next stage. When a super QW value is no longer needed, the value is shifted to the next stage and a new value is shifted in from the prior stage. The first stage gets new super QW values from the QW input stream.

Super QW values are reused between block differences by storing the address of the starting row of the super QW in the StartRow register (Figure 14). When a block difference completes, the QW column local memory address is set to StartRow and StartRow is incremented. StartRow is reset when the super RB is shifted left one column.

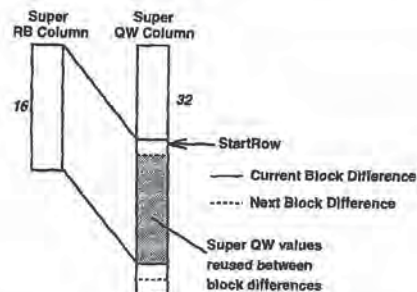


Figure 14: The super RB column shifts down through super QW column, performing a block difference at each step. StartRow is the address of the first row of the block difference.



Moving the super RB from right to left allows super QW values to be reused between sets of block differences. Figure 15 shows how the last columns of the current super QW are the first columns used in the super QW of the next super RB computation. This data motion removes the need to preload super QW values for the next set of block differences.

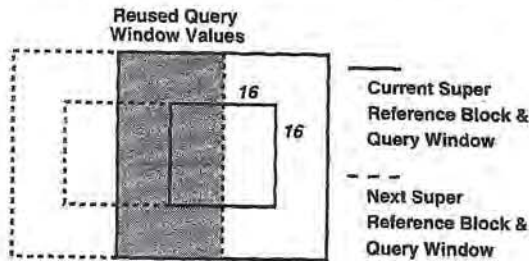


Figure 15: The last super QW columns used to compute motion estimation for a super RB are reused in the computation for the next RB.

The only time data loading stalls computation is the beginning of a row of super RBs. In this case the required super QW values were not used with the prior super RB and must be loaded. The next section shows that the cost is minor, being amortized over a long computation.

### 6.3 Motion Estimation Performance

Motion estimation is not a memory bound computation and with our implementation no memory stalls are encountered. The cycles not spent computing absolute-difference-accumulation operations are due to initialization, finalization, reconfiguration, and the loading of super QWs. For an image of size  $720 \times 576$ , using RaPiD-1, loading the super QW costs 18,432 cycles for motion estimation of one frame<sup>3</sup>. The overhead of loading and reconfiguration time take less than 0.03% of the total number of cycles. As a result, a RaPiD-1 array performs close to its peak speed of 1.6 GOPS (where GOPS is a billion absolute-difference-accumulates per second).

The speedup of motion estimation scales well as the data size grows and with future versions of RaPiD. As data size grows, the cycles used to load super QWs will grow linearly, while the cycles spent in computation grow with the square of the data size. Thus as the data size grows a larger percentage of cycles will be spent computing.

Future versions of RaPiD will have more stages and larger local memories per stage, increasing the number of RBs per super RB and thus the amount of parallelism. Typical images also use 8-bit data, allowing

<sup>3</sup>The super QW must be preloaded  $576/16$  times and a preload takes  $16 * 32$  cycles, resulting in 18,432 cycles.

us to double gauge RaPiD's 16-bit data path, gaining another factor of two in speedup.

## 7 Parametric Curve Generation

This section describes how arbitrary 2-D Bézier curves with four control points<sup>4</sup> can be computed by RaPiD using Apex, an architecture for generating a large class of parametric curves and surfaces [2]. Apex differs from the previous applications in that it maps a triangular data-flow onto RaPiD as shown in Figure 16. Each node in the tree performs a weighted average on the two inputs values and passes the result to the parent node. In symbolic form this is equivalent to

$$V_s(t) \leftarrow (1-t)V_{\text{left}} + tV_{\text{right}} = V_{\text{left}} + (V_{\text{right}} - V_{\text{left}})t$$

The root node produces a new point of the Bézier curve for each  $t$ . The nodes are mapped onto the RaPiD stages in the order indicated by the numbers inside the nodes (Figure 16). This particular mapping minimizes the communication between nodes.

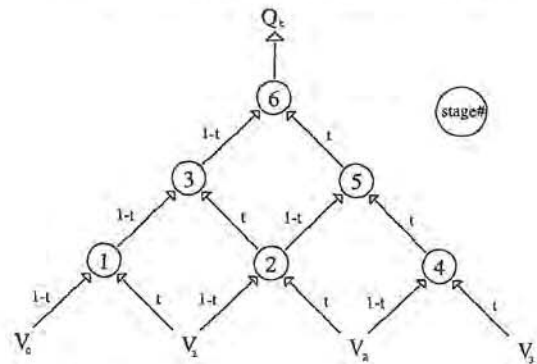


Figure 16: Data-flow graph for computation of the Bézier curve  $Q_t$  described by the control points  $V_i$ . Each node performs a weighted average (weights are edge labels) of its two inputs.

### 7.1 Apex Implementation

The algorithm can be split into initialization and computation. During initialization, the control points are loaded (e.g. stages 1, 2, and 4 in Figure 16) and a  $\Delta t$  increment is specified for  $t$ . Then the repetitive computation starts in which each node increments its private copy of  $t$  by  $\Delta t$  and performs the required computation. During the computational phase, no further external inputs are required.

Computing a 2-D Bézier curve produces two coordinate values per point. The two values can be computed independently. Since we only need six stages

<sup>4</sup>With very little additional effort this can be changed to Bézier curves of arbitrary dimension and with up to six control points on a 16-cell RaPiD array.



per coordinate value (see Figure 17), both can be computed in parallel using a total of twelve stages.

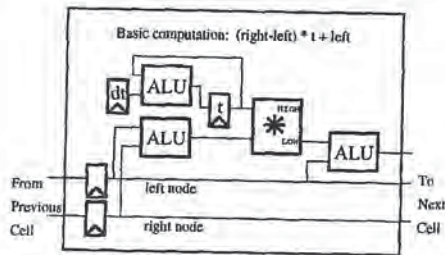


Figure 17: Netlist for one cell of Apex. The  $dt$  register is loaded from a datapath (not shown) before computation begins. Leaf nodes have two additional registers holding the constant control points.

The accuracy and resolution is limited by that  $t$  and  $\Delta t$  which in our current implementation is represented by a 16 bit register. The value  $t$  can be computed in the first stage using two registers (i.e. 32 bits) to substantially reduce the forward differencing errors. It would then propagate down the pipeline.

## 7.2 Apex Performance

Apex outputs a new point of the Bézier curve every cycle with relatively small initialization overhead. If we assume that 100 1000-point curves are displayed before reconfiguration is necessary, the setup overhead is only 0.2% for RaPiD-1 (Section 2.3) and it would perform at nearly 1.2 GOPS (where one OP is one weighted average). This is close to peak performance with the small loss in performance due to the fact that four cells are not used in the computation.

## 8 Conclusion and Future Directions

RaPiD represents an efficient configurable computing solution for regular computationally-intensive applications. In this paper, we have described how four different applications are mapped to the RaPiD array. These applications require a particular set of architectural features provided by RaPiD. We believe this feature set enables RaPiD to perform a wide range of different computations. By combining the appropriate amount of static and dynamic control, RaPiD achieves substantially reduced control overhead relative to FPGA-based and general-purpose processors. RaPiD is optimized for highly predictable and regular computations, reducing the control overhead. The assumption is that RaPiD will be integrated closely with a RISC engine on the same chip. The RISC would control the overall computational flow, performing the unstructured computations which it does best, while farming out the heavy-duty, brute-force computation to RaPiD.

Several challenges remain. The range of RaPiD applications needs to be extended, and integrated applications comprising different computations need to be investigated. The RaPiD B programming model needs to be evaluated and new compiler optimizations implemented. Finally, we would like to investigate how parallel language and compiling methods can be applied to programming RaPiD applications at a higher level.

## Acknowledgments

We would like to thank Larry McMurchie and Chris Fisher for their contributions to the RaPiD project.

## References

- [1] J. M. Arnold et al. The Splash 2 processor and applications. In *Proceedings IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pages 482–5. IEEE Comput. Soc. Press, 1993.
- [2] T. DeRose et al. Apex: two architectures for generating parametric curves and surfaces. *Visual Computer*, 5:264–76, 1989.
- [3] C. Ebeling, D. C. Cronquist, and P. Franklin. RaPiD—reconfigurable pipelined datapath. In R. Hartenstein and M. Glesner, editors, *6th International Workshop on Field-Programmable Logic and Compilers*, Lecture Notes in Computer Science, pages 126–135. Springer-Verlag, September 1996.
- [4] H. Kung. Let's design algorithms for VLSI systems. Technical Report CMU-CS-79-151, Carnegie-Mellon University, January 1979.
- [5] P. Lee and Z. M. Kedem. Synthesizing linear array algorithms from nested FOR loop algorithms. *IEEE Transactions on Computers*, 37(12):1578–98, 1988.
- [6] C. E. Leieron and J. B. Saxe. Retiming synchronous circuitry. *Algorithmica*, 6:5–35, 1991.
- [7] D. I. Moldovan and J. A. B. Fortes. Partitioning and mapping algorithms into fixed size systolic arrays. *IEEE Transactions on Computers*, C-35(1):1–12, 1986.
- [8] K. A. Vissers et al. Architecture and programming of two generations video signal processors. *Microprocessing & Microprogramming*, 41(5-6):373–90, 1995.
- [9] J. E. Vuillemin et al. Programmable active memories: reconfigurable systems come of age. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 4(1):56–69, 1996.



# Attachment 1C



Search WorldCat

Search

[Advanced Search](#) [Find a Library](#)

[<< Return to Search Results](#)

[Cite/Export](#)

[Print](#)

[E-mail](#)

[Share](#)

[Permalink](#)

[Add to list](#)

[Add tags](#)

[Write a review](#)

Rate this item: 1 2 3 4 5

## Proceedings, the 5th Annual IEEE Symposium on FPGAs for Custom Computing Machines, April 16-18, 1997, Napa Valley, California

### Get a Copy

[Find a copy in the library](#)

Author: [Kenneth L Pocek](#); [Jeffrey M Arnold](#); [IEEE Computer Society. Technical Committee on Computer Architecture.](#)

Publisher: Los Alamitos, Calif. : IEEE Computer Society Press, ©1997.

Edition/Format: Print book : Conference publication : English [View all editions and formats](#)

Rating: (not yet rated) 0 with reviews - Be the first.

Subjects: [Field programmable gate arrays -- Congresses.](#)  
[Computer engineering -- Congresses.](#)  
[Computer engineering.](#)  
[View all subjects](#)

More like this [Similar Items](#)

### Find a copy in the library

Enter your location:  [Find libraries](#)

Submit a complete postal address for best results.

Displaying libraries 1-6 out of 207 for all 8 editions (101 Independence Ave SE, Washington, DC 20540, USA)

Show libraries holding [just this edition](#)

[<< First](#) [< Prev](#) [1](#) [2](#) [3](#) [Next >](#) [Last >>](#)

Library	Held formats	Distance	
1. <a href="#">Library of Congress</a> Washington, DC 20540 United States	Book not held; <a href="#">1 other formats</a>	< 1 mile MAP IT	<a href="#">Library info</a> <a href="#">Ask a librarian</a> <a href="#">Add to favorites</a>
2. <b>Federal Communications Commission</b> Washington, DC 20554 United States	Book not held; <a href="#">1 other formats</a>	1 mile MAP IT	<a href="#">Library info</a> <a href="#">Add to favorites</a>
3. <a href="#">George Washington University</a> Washington, DC 20052 United States	Book not held; <a href="#">1 other formats</a>	2 miles MAP IT	<a href="#">Library info</a> <a href="#">Ask a librarian</a> <a href="#">Add to favorites</a>
4. <a href="#">Research Center, National Academies of Sciences, Engineering, and Medicine</a> Washington, DC 20001 United States	Book not held; <a href="#">1 other formats</a>	2 miles MAP IT	<a href="#">Library info</a> <a href="#">Add to favorites</a>
5. <b>Institute for Defense Analyses Library</b> <b>IDA Library</b> Alexandria, VA 22311 United States	Book not held; <a href="#">1 other formats</a>	7 miles MAP IT	<a href="#">Library info</a> <a href="#">Add to favorites</a>

6. [University of Maryland Libraries](#)  
**UMD Libraries**  
College Park, MD 20742 United States

Book not held;  
[1 other formats](#)

7 miles  
MAP IT

[Library info](#)  
[Search at this library](#)  
[Ask a librarian](#)  
[Add to favorites](#)

<< First < Prev 1 2 3 Next > Last >>

- Details

**Genre/Form:** Conference papers and proceedings  
Congresses

**Material Type:** Conference publication

**Document Type:** Book

**All Authors / Contributors:** [Kenneth L Pocek](#); [Jeffrey M Arnold](#); [IEEE Computer Society. Technical Committee on Computer Architecture.](#)

Find more information about:

**ISBN:** 0818689005 9780818689000 0818689021 9780818689024

**OCLC Number:** 733142130

**Notes:** "IEEE Computer Society order number PR08159"--Title page verso.  
"IEEE order plan catalog number 97TB100186"--Title page verso.

**Description:** x, 250 pages : illustrations ; 28 cm

**Responsibility:** sponsored by the IEEE Computer Society, IEEE Computer Society Technical Committee on Computer Architecture ; edited by Kenneth L. Pocek and Jeffrey Arnold.

- Reviews

User-contributed reviews

[Add a review](#) and share your thoughts with other readers. [Be the first.](#)

- Tags

[Add tags](#) for "Proceedings, the 5th Annual IEEE Symposium on FPGAs for Custom Computing Machines, April 16-18, 1997, Napa Valley, California". [Be the first.](#)

- Similar Items

Related Subjects: (4)

- [Field programmable gate arrays -- Congresses.](#)
- [Computer engineering -- Congresses.](#)
- [Computer engineering.](#)
- [Field programmable gate arrays.](#)

+ Linked Data



# Attachment 1D



1 of 1



BOOK

# Proceedings, the 5th Annual IEEE Symposium on Field-Programmable Custom ...

[Full Record](#)[MARC Tags](#)

000 02049cam a2200433 a 4500

001 712410

005 19980406123106.3

008 971119s1997 caua b 101 0 eng d

035 \_\_ |9 (DLC) 97080098

906 \_\_ |a 7 |b cbc |c copycat |d u |e opcn |f 19 |g y-gencatlg

955 \_\_ |a pb23 11-18-97 to cat; jg00 11-26-97; jg05 01-05-97; jg07 01-15-98

010 \_\_ |a 97080098

020 \_\_ |a 0818681594

020 \_\_ |a 0818681608 (case)

020 \_\_ |a 0818681616 (microfiche)

035 \_\_ |a (OCoLC)37949175

040 \_\_ |a GAT |c GAT |d DLC

042 \_\_ |a lccopycat

050 04 |a TK7895.G36 |b I35 1997

082 00 |a 621.39/5 |2 21

111 2\_ |a IEEE Symposium on FPGAs for Custom Computing Machines |d (1997 : |c Napa Valley, Calif.)

245 10 |a Proceedings, the 5th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, April 16-18, 1997, Napa Valley, California / |c sponsored by the IEEE Computer Society, IEEE Computer Society Technical Committee on Computer Architecture ; [edited by Kenneth L. Pocek and Jeffrey Arnold]

246 1\_ |i Half title: |a FCCM'97

246 30 |a 5th Annual IEEE Symposium on Field-Programmable Custom Computing Machines  
Petitioner Microsoft Corporation - Ex. 1066, p. 93

246 2\_ |a Fifth Annual IEEE Symposium on Field-Programmable Custom Computing Machines

246 14 |a IEEE Symposium on FPGAs for Custom Computing Machines  
 246 18 |a FPGAs for Custom Computing Machines  
 260 \_\_\_ |a Los Alamitos, Calif. : |b IEEE Computer Society Press, |c c1997.  
 300 \_\_\_ |a x, 250 p. : |b ill. ; |c 28 cm.  
 500 \_\_\_ |a "IEEE Computer Society order number PR08159"--T.p. verso.  
 500 \_\_\_ |a "IEEE order plan catalog number 97TB100186"--T.p. verso.  
 504 \_\_\_ |a Includes bibliographical references and index.  
 650 \_0 |a Field programmable gate arrays |x Congresses.  
 650 \_0 |a Computer engineering |x Congresses.  
 700 1\_ |a Pocek, Kenneth L.  
 700 1\_ |a Arnold, Jeffrey M.  
 710 2\_ |a IEEE Computer Society. |b Technical Committee on Computer Architecture.  
 920 \_\_\_ |a \*\* LC HAS REQ'D # OF SHELF COPIES \*\*  
 991 \_\_\_ |b c-GenColl |h TK7895.G36 |i I35 1997 |t Copy 1 |w BOOKS

[Request this Item](#)
 [LC Find It](#)

## Item Availability



CALL NUMBER [TK7895.G36 I35 1997](#)  
Copy 1

Request in Jefferson or Adams Building Reading Rooms

Status Not Charged

CALL NUMBER [TK7895.G36 I35 1997 FT MEADE](#)  
Copy 2

Request in Jefferson or Adams Building Reading Rooms - STORED OFFSITE

Status Not Charged



# Attachment 1E



BOOK

# Proceedings, the 5th Annual IEEE Symposium on Field-Programmable Custom ...

[Full Record](#)[MARC Tags](#)

Meeting name

[IEEE Symposium on FPGAs for Custom Computing Machines \(1997 : Napa Valley, Calif.\)](#)

Main title

Proceedings, the 5th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, April 16-18, 1997, Napa Valley, California / sponsored by the IEEE Computer Society, IEEE Computer Society Technical Committee on Computer Architecture ; [edited by Kenneth L. Pocek and Jeffrey Arnold]

Published/Created

Los Alamitos, Calif. : IEEE Computer Society Press, c1997.

[Request this Item](#) [LC Find It](#)

## More Information

LCCN Permalink

<https://lccn.loc.gov/97080098>

Description

x, 250 p. : ill. ; 28 cm.

ISBN

0818681594  
0818681608 (case)  
0818681616 (microfiche)

LC classification

TK7895.G36 I35 1997    Petitioner Microsoft Corporation - Ex. 1066, p. 96

6/29/2018

LC Online Catalog - Item Information (Full Record)

Variant title	Half title: FCCM'97 Fifth Annual IEEE Symposium on Field-Programmable Custom Computing Machines
Portion of title	5th Annual IEEE Symposium on Field-Programmable Custom Computing Machines
Cover title	IEEE Symposium on FPGAs for Custom Computing Machines
Spine title	FPGAs for Custom Computing Machines
Related names	Poczek, Kenneth L. Arnold, Jeffrey M. IEEE Computer Society. Technical Committee on Computer Architecture.
LC Subjects	Field programmable gate arrays--Congresses. Computer engineering--Congresses.
Browse by shelf order	TK7895.G36
Notes	"IEEE Computer Society order number PR08159"--T.p. verso. "IEEE order plan catalog number 97TB100186"--T.p. verso. Includes bibliographical references and index.
LCCN	97080098
Dewey class no.	621.39/5
Other system no.	(OCoLC)37949175
Type of material	Book

---

## Item Availability >

CALL NUMBER	TK7895.G36 I35 1997 Copy 1
Request in	Jefferson or Adams Building Reading Rooms
Status	Not Charged

CALL NUMBER

TK7895.G36 I35 1997 FT MEADE



6/29/2018

LC Online Catalog - Item Information (Full Record)

Copy 2

Request in

Jefferson or Adams Building Reading Rooms - STORED OFFSITE

Status

Not Charged

# Attachment 1F

# Specifying and Compiling Applications for RaPiD\*

Darren C. Cronquist, Paul Franklin, Stefan G. Berg, and Carl Ebeling

Department of Computer Science and Engineering  
University of Washington  
Box 352350  
Seattle, WA 98195-2350

## Abstract

Efficient, deeply pipelined implementations exist for a wide variety of important computation-intensive applications, and many special-purpose hardware machines have been built that take advantage of these pipelined computation structures. While these implementations achieve high performance, this comes at the expense of flexibility. On the other hand, flexible architectures proposed thus far have not been very efficient. RaPiD is a reconfigurable pipelined datapath architecture designed to provide a combination of performance and flexibility for a variety of applications. It uses a combination of static and dynamic control to efficiently implement pipelined computations. This control, however, is very complicated; specifying a computation's control circuitry directly would be prohibitively difficult.

This paper describes how specifications of a pipelined computation in a suitably high-level language are compiled into the control required to implement that computation in the RaPiD architecture. The compiler extracts a statically configured datapath from this description, identifies the dynamic control signals required to execute the computation, and then produces the control program and decoding structure that generates these dynamic control signals.

## 1 Introduction

The RaPiD architecture is a field-programmable architecture that allows pipelined computational structures to be constructed from an array of arithmetic units, registers and memories. These are interconnected and controlled using a combination of static control, which does not change during the computation, and dynamic control, which does. This paper deals with the problems of specifying linear pipelined computations and compiling a specification into the combination of static configuration and dynamic control required to program the RaPiD architecture.

We begin with a programming language that allows a pipelined computation to be described in both time and space. Specific operations are assigned to a specific pipeline stage at a specific time. Time is described using nested loops, while space is described by the innermost loop. Each iteration of this innermost loop is allocated to a specific pipeline stage at a specific time. Since pipelined computations are both regular and repetitive, descriptions in this form are usually quite concise.

When compiling a program, we take the classic approach of partitioning the implementation into datapath and control. The program describes the operations performed by each pipeline stage in each cycle. These operations determine the underlying pipelined datapath. Typically this datapath has a number of dynamic controls to change the functionality and interconnection of elements in the datapath during the computation. These controls are decoded from instruction bits passed down the array, which are in turn produced by a control program. Since each application needs different control, each will use the instruction bits and decoding structure differently, and will have its own control program.

Although the compiler was designed specifically for RaPiD, we believe the RaPiD-C language provides a clean and effective way to specify pipelined computations. For example, a different back-end to our compiler could be used to generate implementations in different technologies such as FPGAs or custom ASICs. In fact, we see the RaPiD architecture model used in a variety of ways. One possibility is to create a single very flexible implementation that could be used for a wide variety of different problems. This implementation would include a set of generic functional units and a very flexible set of interconnection resources. Another possibility would be to create a "custom" RaPiD implementation tailored specifically for one predetermined set of computations. This implementation would trade flexibility for reduced cost.

We begin by giving a brief overview of the RaPiD architecture model. We then present the matrix multiply application to motivate the approach we have taken for language design and compilation. Next, we present RaPiD-C language features and describe how

---

\*This work was supported in part by the Defense Advanced Research Projects Agency under Contract DAAH04-94-G0272. D. Cronquist was supported in part by a Gray fellowship. P. Franklin was supported in part by an NSF fellowship.



programs are written using the language constructs. Finally, we describe the compilation process used to generate and optimize datapath and its control.

## 2 The RaPiD Architecture

This section provides a brief overview of the architectural details of RaPiD which directly affect the compilation process. For a more thorough description of the architecture, see [1] and [2].

RaPiD is a coarse-grained field-programmable architecture for compute intensive applications. The architecture consists of an abundance of functional units such as ALUs and multipliers as well as general purpose registers (GP-REGs) and RAMs. As an example, a version of the architecture that we have used for benchmarking contains 96 GP-REGs, 48 32-entry RAMs, 48 ALUs, and 16 multipliers, all supporting 16-bit data operands. This benchmark version is approximately 100mm<sup>2</sup> in a .5 u process and runs conservatively at 100 MHz.

Such a large number of functional units must be interconnected in a cost-effective manner. Although a crossbar would provide the greatest flexibility, the myriad of functional units requiring connections—over 200 in the benchmark version—make this approach infeasible. Instead, RaPiD arranges the functional units linearly above a field-programmable segmented bus structure. A linear structure is easily manageable, yet it reduces implementation cost and control requirements tremendously. By using the small RAMs spread throughout the array as buffers, multidimensional tasks can be performed on RaPiD arrays. The underlying datapath, i.e. which functional units can forward results to each other, is configured statically on a per application basis. During the execution of an application, the data movement between functional units can change every cycle via a decoded instruction.

A simple example is shown in Figure 1. A register is used to hold a constant value, such as a coefficient for a FIR filter. The underlying datapath is statically configured so the register can load from either an input stream or its previous value. During the execution of the instruction stream, dynamic control directs data movement on a cycle-by-cycle basis. In summary, *static control* determines the extent that data can flow in a given application. *Dynamic control* determines the cycle by cycle data movement under the restrictions placed by static control. For example, the dynamic control would specify when the register should load from the input stream and when it should hold its value by loading from the feedback path.

The RaPiD architecture provides *hard* control signals, which are fixed by the configuration data, and *soft* control signals, which can change each clock cycle. Soft control signals drive ALU functions, input and output stream enables, RAM writes and increments, and multiplexer selects. As a result, RaPiD

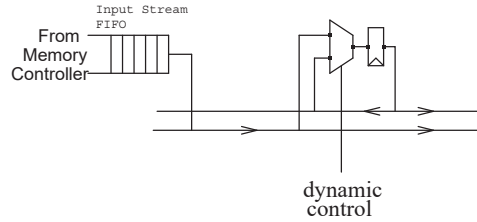


Figure 1: A register configured to load or hold its value for a constant multiply. The interconnect shown is configured statically but the dynamic control signals can change every cycle.

contains a substantial number of soft control signals—over 1000 in the benchmark version. To reduce the circuitry required to generate these signals, a pipelined control path that parallels the datapath is used to generate these signals from a narrow instruction (eg. 32 bits) inserted at the beginning of the array. This “instruction” contains all the information required by the various pipeline stages to compute their dynamic control signals. The control path comprises a set of 1-bit segmented busses similar in structure to the datapath busses, as shown in Figure 2. This bus structure allows the instruction bits to be individually manipulated as they proceed down the RaPiD array.

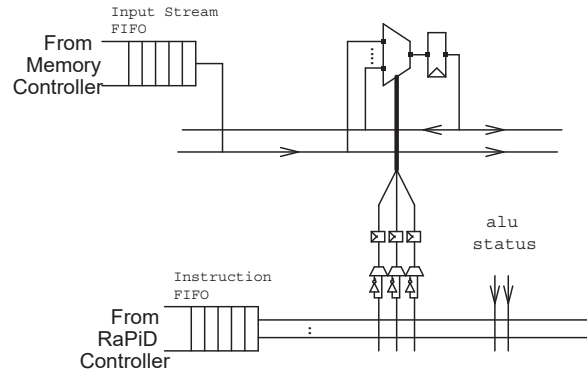


Figure 2: Soft control circuitry used to dynamically control the circuit shown in Figure 1

The majority of soft control bits are static for a given application, and are wired to a constant 0 or 1. Other bits are wired to one of the other control wires. This control often comes directly from the controller, pipelined as needed. However, more complex decoding using 3-input look-up tables (LUTs) can be used to decode several instruction bits into the appropriate control or to combine pipelined control with status information (e.g. ALU carry). The LUTs also contain optional registers, allowing for simple finite state machines (FSMs) occasionally required by non-pipelined control. Such FSMs can be used to activate a function for several cycles in each stage, one stage at a time.

If RaPiD supported deep pipelining within the control path, this could be done easily by placing enough registers between each stage. Instead, this can be constructed using a FSM; a stage can remember whether or not its RAM is active, and one instruction bit can be used to deactivate one stage and activate the next, requiring only two control lines. This is used in implementing a 2-D DCT on RaPiD [2].

The number of busses required in the control path varies by application, but is not large because control signals tend to be reused extensively. The benchmark version of the RaPiD architecture provides 31 busses, which can be pipelined and otherwise manipulated individually. This is more than enough for the current set of applications, even using non-optimal mappings produced by automated CAD tools.

## 2.1 Datapath Controller

The RaPiD array consumes one instruction per cycle which drives the beginning of the control buses. Generating this instruction stream is nontrivial because it often controls several tasks running in parallel; the matrix multiply example in the next section illustrates this. The RaPiD datapath controller contains several simple microcontrollers (instruction generators) whose output is combined to form instructions for the RaPiD array.

Each instruction generator executes a microprogram to generate a stream of instructions. The generators are optimized to handle nested and sequential loops; they also contain microinstructions for performing simple arithmetic synchronization.

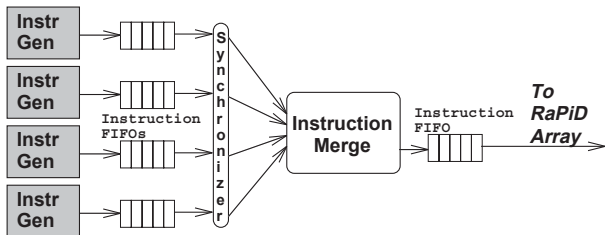


Figure 3: The RaPiD controller.

The instruction streams produced by the instruction generators are synchronized via SIGNAL and WAIT tags. The stream containing a WAIT tag is stalled until the matching SIGNAL occurs in another stream. After the instruction streams have been synchronized, they are combined in a bitwise fashion, and the final instruction then passes through the last instruction FIFO and proceeds to the RaPiD array.

RaPiD also contains address generators used for generating sequences of memory addresses used by the memory controller (Figure 4). The address generators use arithmetic microinstructions to produce address

sequences. The memory controller handles these memory requests by placing data in or removing data from the appropriate input or output FIFO.

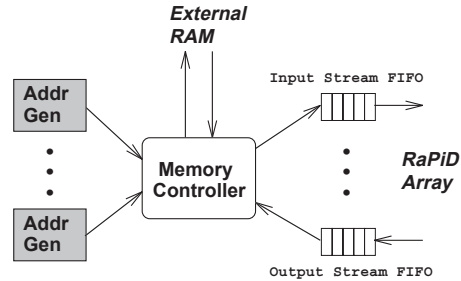


Figure 4: The RaPiD address generators and I/O streams.

## 3 Specification

To map an algorithm to the RaPiD array we have designed a new parallel programming language, RaPiD-C. Although created with RaPiD in mind, the methods of specification and compilation could be extended to other architectures for implementing pipelined computations. In addition, this programming technique could be used for ASIC design.

### 3.1 Motivation: Matrix Multiply

To motivate a specification language, we first look at a common, well-studied application—matrix multiply. The problem takes an  $L \times M$  matrix  $\mathbf{A}$  and an  $M \times N$  matrix  $\mathbf{B}$  and computes the  $L \times N$  matrix  $\mathbf{C} = \mathbf{A} \times \mathbf{B}$ , as shown in the nested-loop specification of Figure 5. As it stands, this high-level specification is far from a

```

for (i=0; i < L; i++)
  for (j=0; j < M; j++)
    for (k=0; k < N; k++)
      if (j==0) C[i][k] = A[i][j]*B[j][k];
      else C[i][k] += A[i][j]*B[j][k];

```

Figure 5: Nested loop specification for matrix multiply

mapping to a pipelined linear array. In particular, the parallelism and the data I/O are not specified, and the algorithm must be partitioned to fit on the target architecture.

Automating these processes is a difficult problem for an arbitrary specification, and one we do not solve here. Instead, we propose a language that is C-like and requires the programmer to specify the parallelism, data movement, and partitioning. To this end, the programmer uses well known techniques of loop transformation [5] and space/time mapping [4, 3]. The resulting specification is a nested loop where outer loops

specify time and the innermost loop space.<sup>1</sup> In the context of RaPiD-C, the space loop refers to a loop over the *stages* of an algorithm, where a stage is one iteration of the innermost loop. The compiler maps the entire stage loop to the target architecture by unrolling the loop to form a flat netlist. Hence, the stage loop, also called a **Sloop**, has implicit parallelism since it executes in a single cycle on the target architecture.<sup>2</sup> As a result, the programmer must permute and tile the loop-nest so that the computation required after unrolling the innermost loop will fit onto the target architecture. The remainder of the loop nest determines the number of times the **Sloop** is executed.

The programmer first transforms the original specification by choosing a loop to iterate over the stages, optimizing for speed, memory, scalability, etc. Partitioning of the algorithm is based on the number of functional units and available memory in the target architecture. For example, consider mapping matrix multiply to an architecture with  $S$  multipliers, at least 3 RAMs per multiplier, and  $R$  words of memory per RAM. The innermost loop is partitioned by the number of stages (i.e. multipliers) and the outer loops by the size of the RAMs. Since the stage loop is the innermost loop,  $k$  has been replaced by the stage iteration variable  $s$ . Loop permutation is applied, yielding the code in Figure 6.<sup>3</sup> From now on, instead of explicitly stating the innermost loop as **for** ( $s=0$ ;  $s < S$ ;  $s++$ ), we will simply write **Sloop**.

```

for (f=0; f < L; f+=R)
  for (g=0; g < M; g+=R)
    for (h=0; h < N; h+=S)
      for (i=0; i < R; i++)
        for (j=0; j < R; j++)
          for (s=0; s < S; s++)
            if (j+g==0) C[i+f][s+h] = A[i+f][j+g]*B[j+g][s+h];
            else C[i+f][s+h] += A[i+f][j+g]*B[j+g][s+h];

```

Figure 6: *Matrix multiply partitioned to space and time*

Memory accesses are determined by examining indexing in every stage on every cycle (recall that the **Sloop** executes in a single cycle). Since  $A[i+f][j+g]$  is independent of  $s$ , the appropriate **A** matrix value will be broadcast to the entire array on every cycle. Expression  $B[j+g][s+h]$  references  $R$  elements (the length of the  $j$  loop) of the **B** matrix in each stage, which can be stored in a RAM in each stage. Moreover, since  $h$  changes every  $R^2$  cycles, a new set of elements must be loaded every  $R^2$  cycles. To prevent the array from stalling, the RAMs can be double-buffered. Finally, expression  $C[i+f][s+h]$  references  $R$  elements (the length of the  $i$  loop) of the **C** matrix in

<sup>1</sup>Since RaPiD is a linear architecture we have a singly nested loop for space. An  $n$ -dimensional architecture would have an  $n$ -nested loop dedicated to space.

<sup>2</sup>In actuality, pipelining and retiming usually cause the **Sloop** to be executed on a diagonal of the time axis instead of the same cycle.

<sup>3</sup>For ease of presentation, we assume that  $R$  divides  $L$  and  $M$ , and that  $S$  divides  $N$ .

each stage, which again can be stored in a RAM in each stage. Since every stage produces a result on the same cycle, these results are pipelined down the array instead of being broadcast; the final stage stores the values from this pipeline into the **C** memory.

Matrix multiply can now be broken down into three processes: preload, computation, and output. These processes all run in parallel but need to be precisely synchronized to allow them to communicate. The preload loop-nest must complete one iteration of its  $h$ -loop before the computation loop-nest begins. The computation loop-nest must complete one iteration of its  $g$ -loop before the output loop-nest can begin. To simplify this specification, the language supports parallel loop nodes and Signal/Wait synchronization pairs. The double buffering of the **B** values is performed by a two-dimensional array of  $S \times 2$  RAMs which is indexed by the stage number and a boolean toggle bit to flip between the RAMs on the appropriate cycle. This sums up the major features of RaPiD-C, which is described in more detail in the next section. Pseudo-code for the RaPiD-C implementation of matrix multiply is shown in Figure 7. To clarify the structure of this code, we often write a control tree as shown in Figure 8. Control trees will be described in detail in the next section.

```

Par {
  // Preload loop-nest
  for (f1=0; f1 < L; f1+=R)
    for (g1=0; g1 < M; g1+=R)
      for (h1=0; h1 < N; h1+=S) {
        for (j1=0; j1 < R; j1++)
          for (i1=0; i1 < S; i1++)
            Sloop
              if (i1==s) ramB[s][!toggle][j1] = B[j1+g1][s+h1];
              Signal(comp); Wait(preload);
            }
      }
  // Computation loop-nest
  for (f2=0; f2 < L; f2+=R)
    for (g2=0; g2 < M; g2+=R)
      for (h2=0; h2 < N; h2+=S) {
        Signal(preload); Wait(comp);
        for (i2=0; i2 < R; i2++)
          for (j2=0; j2 < R; j2++)
            Sloop {
              if (i2==0 && j2==0) toggle = !toggle;
              if (j2==0 && g2==0)
                ramC[s](i2) = A[i2+f2][j2+g2]*ramB[s][toggle](j2);
              else
                ramC[s](i2) += A[i2+f2][j2+g2]*ramB[s][toggle](j2);
              if (j2==R-1 && g2==M-R) {
                pipeOut = ramC[s](i2);
                Signal(output);
              }
            }
      }
  // Output loop-nest
  for (f3=0; f3 < L; f3+=R)
    for (h3=0; h3 < N; h3+=S)
      for (i3=0; i3 < R; i3++) {
        Wait(output);
        for (j3=S-1; j3 ≥ 0; j3--)
          Sloop
            if (s==S-1) C[i3+f3][h3+j3] = pipeOut;
          }
      }
}

```

Figure 7: *Matrix multiply after I/O, ram allocation*



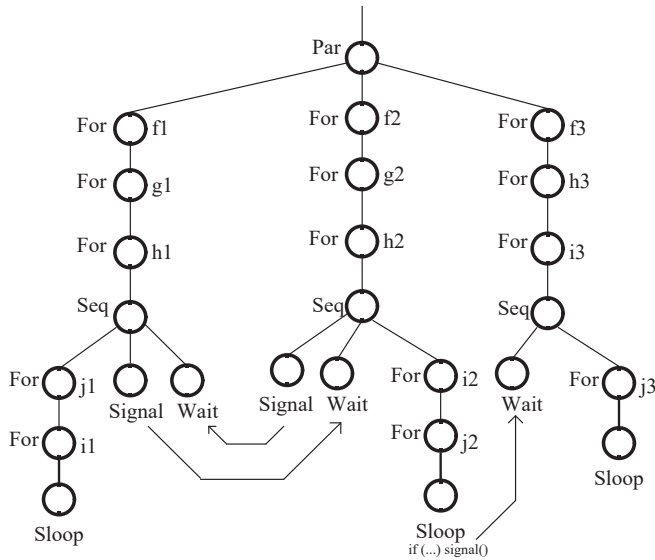


Figure 8: Control tree for matrix multiply

### 3.2 Control Trees

Control trees are a convenient representation of a RaPiD-C program's loop structure. They are particularly useful while manipulating a program's structure by performing loop transformations, and as an aid to explaining a program's control structure. This section presents them, while using them to explain the control structures available in RaPiD-C.

RaPiD-C uses a control tree to specify the loop structure of a particular application. In a complete RaPiD-C program, this tree is part of the code, as shown in Figure 7; however, while determining what a program's control tree should look like, it is often useful to draw it separately. This section uses control trees to reintroduce the control constructs already shown above.

Figure 9 shows two simple RaPiD-C trees. Tree (a) represents two nested loops in time. The inner loop contains the stages loop, or `Sloop`. This loop automatically iterates the reserved variable `s` over all of the stages ( $0 \leq s < S$ ). Tree (b) illustrates a computation split into two loops; the `j` loop will begin after the `i` loop completes. Note that each branch has its own `Sloop`.

Each `Sloop` in a control tree contains code to be compiled to the target architecture. Inside `Sloop` blocks, a programmer uses a C-like syntax with special objects representing some features of the architecture. Since each block actually executes `S` times, conditional statements can check the value of `s` to restrict code segments to specific stages. Conditionals can also compare against a `For` node's iteration count such as `i==3`. In addition, the conditionals `.first`, `.last`, or `.live` test on the first, last, or any iteration

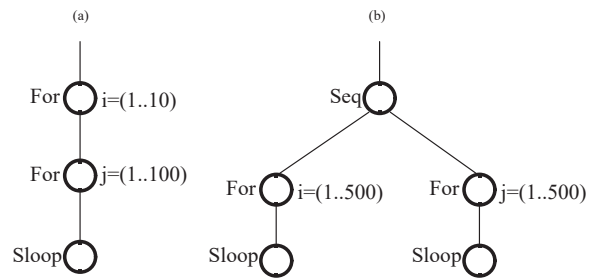


Figure 9: Simple RaPiD-C control trees. (a) Nested loops. (b) Loops to be run in sequence.

of a loop, respectively. Code which needed to be executed every 100th iteration can easily be coded in the `Sloop` for Figure 9(a) with the condition `j.last`.

Note that a condition specifying every 100th iteration of Figure 9(b) is more complex. Control trees should represent the actual control needed by an application. The RaPiD-C code for matrix multiply shown in Figure 7 contains only relatively simple conditions, indicating that its control tree captures the loop and control structure of the application.

RaPiD-C uses `Par` nodes to indicate branches which should run in parallel. RaPiD-C also contains synchronization primitives; a `Wait` node stalls until it receives a signal from either a `Signal` node or a signal statement in a `Sloop`. Figure 10 shows a control tree in which the right `Sloop` will start executing as the left `Sloop` begins its second iteration.

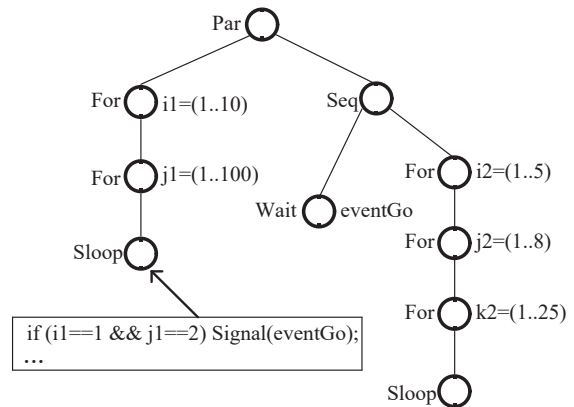


Figure 10: A Signal/Wait pair to run two nested loops in parallel, offset by one cycle

Control trees can also contain `Inf` nodes; they are similar to `For` nodes but execute on every cycle, halting immediately when all other control is exhausted. Table 1 summarizes the node types presented in this section.

Table 1: *Control tree node types in RaPiD-C*

Node Type	Children Execution	Length (in cycles)
Seq	In sequence, one at a time	Sum of lengths of children
Par	In parallel, all starting simultaneously	Length of longest child
For	Its single child, loop iteration times	# of iterations * child length
Inf	Its single child, many times	Number of cycles in entire tree
Sloop	No children	One cycle
Wait	No children	Until signaled
Signal	No children	Zero cycles

Table 2: *Data types in RaPiD-C*

Data Type	Specifies
Word	Single width variable
Long	Double width variable
Bool	Single bit value, used for conditional statements
Ram	Fixed size RAM local to a stage
Pipe	Inter-stage communicator

### 3.3 Communication

A RaPiD-C application needs to communicate among its stages and with the outside world. This is provided for with separate mechanisms. Communication with the outside world is done through array references. Inter-stage communication is accomplished using pipes that connect a number of stages together.

The programmer can specify an arbitrary number of external arrays that can be read from or written to inside a RaPiD-C program (see arrays A, B, and C in Figure 6). A limitation imposed on the programmer is that all references must be data independent since memory addressing is determined at compile time.

Pipes are used to communicate values between stages. A pipe is just a global bus with a number of optional registers between stages. The programmer can therefore use them to feed data into or out of the array or to communicate intermediate results from one stage to the next. Figure 7 shows an example of a pipeline used for writing the result matrix to the external array C. All stages output their final results to `pipeOut` at regular intervals. The last stage reads from `pipeOut` and stores the read values in array C.

### 3.4 More RaPiD-C Types

RaPiD-C has several predefined types to support both computation and data communication within an algorithm, as shown in Table 2.

The types `Word` and `Long` represent the single and double precision data types for use within a stage. For computation that is similar across several stages, typically an array of `Words` or `Longs` is defined.

Type type `Bool` is used for control defined by the programmer. For example, double buffering two RAMs requires a toggle signal. For example, the code `if (i2==0 && j2==0) toggle = !toggle;`

Table 3: *Operators for RAMs.*

Operator	Action
<code>ramFoo.address = x</code>	Set the address register to value x, mod size
<code>ramFoo.address++</code>	Increment the address register, mod size
<code>ramFoo = y</code>	Set the ram value for the current address to y (y is of type word)
<code>y = ramFoo</code>	Read the value for the current address
<code>ramFoo(i)</code>	Automatically address ram with respect to For loop <i>i</i>

flips the toggle bit on the appropriate cycle in matrix multiply (see Figure 7).

The type `Ram` specifies a fixed-size local memory in a stage of the target architecture. `Ram` is accessed via an implicit address register that can be assigned, cleared and incremented. Table 3 lists the valid operators on a ram.

The `Ram` type represents an architecture-specific device. When specifying applications targeted at other architectures, other architecture-specific types might be called for.

## 4 Compilation

A RaPiD-C program clearly specifies an algorithm's hardware requirements. As a matter of fact, the union of all `Sloop` blocks is very close to a structural description of the algorithm. One difference from a true structural description is that `Sloop` statements are specified sequentially but execute in parallel. A netlist must be generated to maintain these sequential semantics in a parallel environment. Another difference is that control is not explicit but instead embedded in a nested-loop structure. This control must be extracted into multiplexer select lines and functional unit control. Then, an instruction stream must be generated which can be decoded to form this control. A final difference from a true structural description is the implicit decoupling of data I/O. Address generators must be instantiated to take the data to and from memory at the appropriate time. Hence, compiling RaPiD-C into a structural description consists of four components: netlist generation, dynamic control extraction, instruction stream/decoder generation, and I/O address generation.

The compilation process produces a structural specification consisting entirely of components on the target architecture. The netlist is then mapped to the architecture via standard FPGA mapping techniques including pipelining, retiming, and place and route. The place and route solution fully specifies the static setting required to program the array.

### 4.1 Netlist Generation

Generating a parallel netlist from a sequential specification is straightforward. Consider the three types of

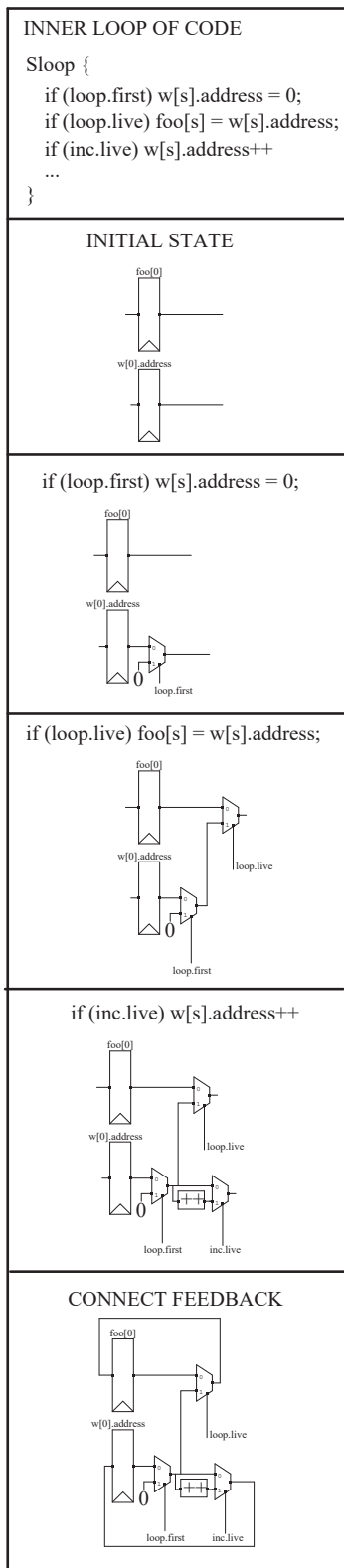


Figure 11: *Generation of a netlist from RaPiD-C.*

data dependencies found in sequential code: true (read after write), anti (write after read), and output (write after write). The idea is to convert variables into registers, noting that if a register is read and written on the same cycle, the register's value on the previous cycle is read. An anti-dependence requires the previous value of a variable be read, so a register is sufficient. A true dependence requires the current value of a variable to be read, so data forwarding is used. Finally, an output-dependence is implemented with a register whose input multiplexer gives priority to the latest write in the sequential code.

The compiler converts a RaPiD-C program into a structural specification by interpreting the union of all `Sloop` blocks for each value of `s`. During interpretation, the compiler instantiates registers for variables, ALUs for adds (and other operations), multipliers for multiplies, and multiplexers for `if-then-else` statements. Once interpretation is complete, the final value of each variable is connected to the input of the variable's register, creating state across cycles.

For example, Figure 11 shows the netlist construction for a small set of `Sloop` statements. During the first iteration of the loop, `s` is assigned to zero, creating the variables `foo[0]` and `w[0].address` which are initialized as registers. The first line of code adds a multiplexer to the address register which either holds its current value or loads zero, depending on the value of `loop.first`. The second line updates `foo[0]` to be the *current* value of the address register, if `loop.live` is true. The third line instantiates an incrementer and updates the value of the address register if `inc.live` is true. After the final reference to the variables `foo[0]` and `w[0].address`, the current values are connected as inputs to the original registers, providing support for dependencies across iterations of the control tree.

## 4.2 Dynamic Control Extraction

Dynamic control can take many forms depending on the versatility of the target architecture. The most common dynamic signals are used to time data movement, as in a multiplexer's select lines, and to specify a change in computation, as in a functional unit's operation signals. As a result, two key steps are required to generate dynamic control: multiplexer merging and functional unit merging.

### 4.2.1 Multiplexer Merging

The initial netlist generation forms two-input multiplexers for every conditional statement. These smaller multiplexers must be merged to match the size of multiplexers on the target architecture, potentially changing the required control.

To merge multiplexers a depth-first search of the initial two-input multiplexer netlist is performed starting at the output streams, since all required functional

units must be reachable from the output. When a previously unreachable functional unit is found, each input is replaced with a 16:1 multiplexer (or whatever size corresponds to the target architecture). The compiler then performs a depth-first search to determine reachable inputs. An input's select condition is the AND of multiplexer conditions along this path. Figure 12 shows the code and a portion of a generated netlist containing three multiplexers which are merged into a single multiplexer with three inputs. The REG1 input is found to be unreachable since the global context becomes  $b \wedge a \wedge \neg a = 0$ .

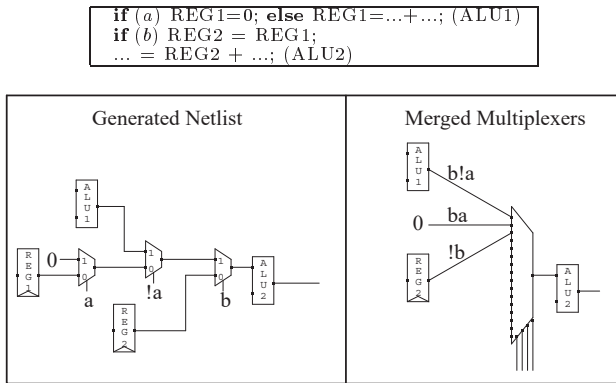


Figure 12: An example of multiplexer merging: three 2-input muxes are merged into a single mux.

#### 4.2.2 Functional Unit Merging

RaPiD-C uses symbols, such as  $*$  and  $+$ , to specify operations on data. Clearly, if the programmer uses a common subexpression, the compiler must be robust enough to map all instances of the expression to the same functional unit. Although a standard common-subexpression elimination algorithm would work for common-subexpressions, the dynamic control of the RaPiD array can optimize some *uncommon*-subexpressions. For example, the expressions  $x-y$  and  $x+y$  are different but could be mapped to the same ALU if they are not both in use during a common cycle. This would require a control signal to be generated to change the ALU function from an addition to a subtraction on the appropriate cycle.

Even expressions with different operands can be merged. If the expressions  $x+y$  and  $w-z$  are not in use during a common cycle, they could be merged by having both  $x$  and  $y$  reach one ALU multiplexer input, and both  $w$  and  $z$  the other. Now three dynamic control signals must be used to change between the two expressions. Since all three signals are equivalent, this second example doesn't use more control path resources than the first. However, the underlying

netlist (static control) becomes more complex, potentially stressing the available routing resources in the datapath; in some cases, the control becomes complex enough that the merging must be rejected.

To support both common and uncommon subexpression elimination, a list of functional units is maintained in every stage. In addition, a boolean *in-use* function is created for each functional unit to record the cycles of operation. The functional units' in-use functions determine when merging can occur. Two functional units can be merged if they are identical (i.e. a common-subexpression), or if their in-use functions are mutually exclusive and the union of their inputs doesn't exceed some internal maximum. This maximum is  $n$  for  $n$ -input multiplexers but clearly should be substantially smaller due to routing constraints. When there is a choice of functional units to merge, the pair with the larger number of common inputs is selected. Functional units which use static control to determine their functionality must be equivalent in these bits to be merged.

For example, consider an ALU with two data inputs (Left and Right) and four control inputs (F3, F2, F1, and F0). The in-use function, *InUse*, determines when the ALU is actually needed. If we are given *alu1* and *alu2* with mutually exclusive *InUse* functions, they can be merged into *alu3* by applying the code in Figure 13.

```

alu3.Left.Merge(alu1.Left, alu2.Left);
alu3.Right.Merge(alu2.Right, alu2.Right);

alu3.InUse = alu1.InUse || alu2.InUse;
alu3.F3 = alu1.InUse && alu1.F3 || alu2.InUse && alu2.F3;
...
alu3.F0 = alu1.InUse && alu1.F0 || alu2.InUse && alu2.F0;

```

Figure 13: Compiler's code to merge two ALUs. The function *Merge* adds all inputs to *alu3*'s input multiplexer and ORs the control of any common inputs.

#### 4.3 Data Dependent Dynamic Control

Some operations, such as maximum and absolute value, require data-dependent dynamic control to be generated. For example, the statement `sum += |x-y|` compiles to a netlist which connects the `sign` status signal of an ALU computing  $x-y$  to the add/subtract control input of a second ALU. If the sign is positive, the second ALU adds the first result to sum, and if negative it subtracts the first result from sum. In more complex data dependent conditions, decoding may be required as shown in the next section.

#### 4.4 Instruction and Decoder Generation

Each dynamic control signal is represented by a boolean function of the following variable types: an event in the control tree, a status bit from the datapath, and a condition on *s*. Examples of such



boolean variables include the first iteration of a For node (`i.first`), the carry condition on an ALU (`alu.carry`), and the equivalence of a For node and `s` (`i==s`), respectively. Each control signal is paired with an in-use function to aid optimization. Given this dynamic control information, a set of boolean functions, whose concatenation forms an *instruction*, must be found from which all dynamic signals can be decoded. This set of functions is limited by the instruction width of the target architecture.

Mapping all dynamic control into a fixed-width instruction involves finding common subexpressions within the dynamic control signals and using in-use information efficiently. In addition, this process may require multi-level minimization, Shannon decomposition and/or compilation to state machines, depending on the complexity of the dynamic control functions.

A dynamic signal comprised entirely of events from the control tree is independent of `s` and can be broadcast to all required stages. For example, the function `i.live && j.first` compiles directly to a bit in the instruction, which is then broadcast to each required functional unit and multiplexer. Since these variables are independent of `s`, they can be computed outside of the array by an external microcontroller.

A dynamic signal whose function is dependent on `s` may require decoding. This might be in the form of a state machine or a simple pipeline. For example, consider a RaPiD-C program containing a For node `i` and a dynamic control function `i==s`. Because the variable `i==s` is dependent on `s`, it can't be directly generated outside the array. However, if `i` has an increment of one and a range which includes zero to the number of stages, this dynamic control signal can be compiled into a singly pipelined control line driven by the boolean function `i==0`, which is independent of `s`. Similarly, a conditional of the form `i==ks` can be realized by creating a control pipeline with `k` registers per stage.

A dynamic control signal that is a function of more than one variable often requires special decoding. For example, the statement `if (i.first && X==Y) FOO = Z;` requires local decoding since the condition `X==Y` is compiled to the "is result zero?" output of an ALU subtract. The binary AND of this signal and `i.first` must be formed in the stage associated with the code, as is shown in Figure 14.

After generating the control path complete with decoding, the final step is to produce microprogram code that will generate the instruction stream. Since the set of boolean functions comprising the instruction consists entirely of boolean variables from the control tree, the microprogram is similar to the control tree itself. Each parallel task of the tree is mapped to an instruction generator, as shown in Figure 3. Hence, the number of instruction generators places a limit on the number of simultaneous parallel tasks in a RaPiD-C program.

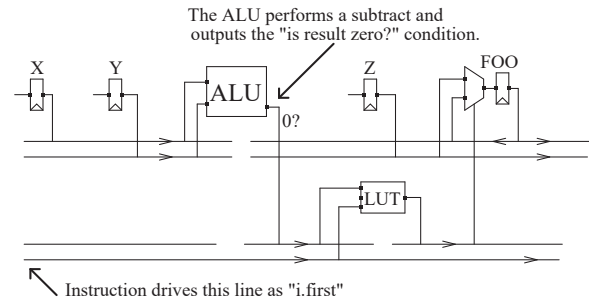


Figure 14: The dynamic condition `i.first && x==y` is generated by configuring a LUT as an AND gate.

An instruction bit depending only on variables from a single parallel task is generated by a single instruction generator. An instruction bit depending on variables from more than one parallel task must be decomposed (using two-level or multi-level minimization) into functions specific to a single parallel task. These functions are computed on their corresponding instruction generators and then later recombined in the merge unit to form the original instruction bit.

#### 4.5 I/O Address Generation

The input/output addresses are generated by a set of address generators, such as shown in Figure 4. Each array reference is extracted from the RaPiD-C specification and dedicated to an address generator.<sup>4</sup> The memory controller processes these requests in parallel with (and potentially ahead of) the computation on the RaPiD array.

For example, pseudo-code for generating the addresses for matrix `B` of matrix multiply is shown in Figure 15, where `B` represents the base offset of the array in memory. Matrix multiply requires three address generators, one for each array used.

```

for (f=0; f < L; f+=R)
  for (g=0; g < M; g+=R)
    for (h=0; h < N; h+=S)
      for (j=0; j < R; j++)
        for (s=0; s < S; s++)
          // Output the address associated with B[j+g][s+h]
          Output(B + (j+g)*N + (s+h));

```

Figure 15: Matrix `B` address generation

#### 4.6 Pipelining and Retiming

Although the target architecture's functional units and memories may be pipelined, the programmer can assume that the units are combinational to simplify the code. In addition, the programmer can specify

<sup>4</sup>Although there is a limit on the number of address generators, two or more references could be mapped to the same address generator if they occur on different cycles.

data using a broadcast model even though such broadcasts might not meet the required cycle time. A retiming step ensures that the final netlist adheres to the target architecture's pipeline structure and timing requirements. A retimed circuit will adhere to the cycle time of the target hardware, taking into account delays through RaPiD elements, as well as pipelining requirements present in the underlying architecture. Because placement cannot be done until retiming is performed, the retimer conservatively estimates routing delays between elements.

## 5 Future Work and Conclusions

There are several ways we plan to extend the capability of the RaPiD-C language and compiler to make them more powerful. Some are simple extensions to the compiler to generate more optimized datapaths. These extensions would rely on extracting more information from the control structure to allow better sharing of resources and a more efficient generation of control. Other extensions are more far-reaching such as incorporating automatic time-multiplexing. Currently the programmer must explicitly describe how the time-multiplexing is done, which can be complicated and error-prone. It would be better to present the programmer with an array of arbitrary length and map this to the physical array by automatically introducing time-multiplexing. Another extension would be to have the compiler infer data movement from a description of the computation. That is, the specification would indicate the operations and data items and the compiler would create the dataflow required to satisfy the computation.

One of the disadvantages to many configurable architectures is the difficulty of specifying and compiling the computation. In this paper we have presented a conceptually clean and effective way to specify a pipelined implementation for regular and repetitive computation. This language requires the programmer to map the computation to space and time, but provides simple and concise ways to do this. The compiler is then able to generate the appropriate configuration data and dynamic control structure to implement the computation in a RaPiD array.

The RaPiD-C language can be viewed either as a convenient, sufficiently high-level language for programmers to describe pipelined computations in, or as an intermediate language used by a parallel compiler to describe the space-time mapping derived from an even higher-level description of the computation. Such a compiler currently appears out of reach for many complex computations, but as research in parallelizing compilers progresses, we may reach the point where RaPiD-C is largely used only by the compiler back-end. Until then, it provides a relatively powerful and convenient way for programmers to program the RaPiD architecture.

## Acknowledgments

We would like to thank Larry McMurchie, Chris Fisher, and Miguel Figueroa for their contributions to the RaPiD project.

## References

- [1] C. Ebeling, D. C. Cronquist, and P. Franklin. RaPiD—reconfigurable pipelined datapath. In R. Hartenstein and M. Glesner, editors, *6th International Workshop on Field-Programmable Logic and Compilers*, Lecture Notes in Computer Science, pages 126–135. Springer-Verlag, September 1996.
- [2] C. Ebeling, D. C. Cronquist, P. Franklin, and S. Berg. Mapping applications to the rapid configurable architecture. In *Field-Programmable Custom Computing Machines (FCCM-97)*, 1997.
- [3] P. Lee and Z. M. Kedem. On high-speed computing with a programmable linear array. In *Proceedings. Supercomputing '88*, pages 425–32. IEEE Comput. Soc. Press, 1988.
- [4] D. Moldovan and J. A. B. Fortes. Partitioning and mapping algorithms into fixed size systolic arrays. *IEEE Transactions on Computers*, C-35(1):1–12, 1986.
- [5] M. E. Wolf and M. S. Lam. A loop transformation theory and an algorithm to maximize parallelism. *IEEE Transactions on Parallel and Distributed Systems*, 2(4):452–471, 1991.

# Attachment 2A



# IEEE Symposium on FPGAs for Custom Computing Machines

10-NOV-1997 BSDS BOSTON SPA  
LS23 780  
IEEE SYMPOSIUM ON FPGAS FOR CUSTOM COMPUTIN  
G MACHINES



4363.086450 YEAR 1997  
AC<sup>32</sup>

1/2

**April 16-18, 1997**  
**Napa Valley, California**

Edited by Kenneth L. Pocek and Jeffrey Arnold

Sponsored by the IEEE Computer Society Technical Committee on Computer Architecture





# PROCEEDINGS

## The 5th Annual IEEE Symposium on Field-Programmable Custom Computing Machines

April 16 – 18, 1997

Napa Valley, California

*Sponsored by*

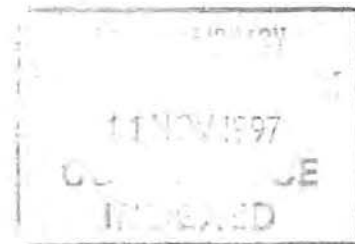
IEEE Computer Society

IEEE Computer Society Technical Committee on Computer Architecture



Los Alamitos, California

Washington • Brussels • Tokyo



Copyright © 1997 by The Institute of Electrical and Electronics Engineers, Inc.  
All rights reserved

*Copyright and Reprint Permissions:* Abstracting is permitted with credit to the source. Libraries may photocopy beyond the limits of US copyright law, for private use of patrons, those articles in this volume that carry a code at the bottom of the first page, provided that the per-copy fee indicated in the code is paid through the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923.

Other copying, reprint, or republication requests should be addressed to: IEEE Copyrights Manager, IEEE Service Center, 445 Hoes Lane, P.O. Box 133, Piscataway, NJ 08855-1331.

*The papers in this book comprise the proceedings of the meeting mentioned on the cover and title page. They reflect the authors' opinions and, in the interests of timely dissemination, are published as presented and without change. Their inclusion in this publication does not necessarily constitute endorsement by the editors, the IEEE Computer Society, or the Institute of Electrical and Electronics Engineers, Inc.*

IEEE Computer Society Order Number PR08159  
ISBN 0-8186-8159-4  
ISBN 0-8186-8160-8 (case)  
ISBN 0-8186-8161-6 (microfiche)  
IEEE Order Plan Catalog Number 97TB100186  
ISSN 1082-3409

*Additional copies may be ordered from:*

IEEE Computer Society  
Customer Service Center  
10662 Los Vaqueros Circle  
P.O. Box 3014  
Los Alamitos, CA 90720-1314  
Tel: + 1-714-821-8380  
Fax: + 1-714-821-4641  
E-mail: cs.books@computer.org

IEEE Service Center  
445 Hoes Lane  
P.O. Box 1331  
Piscataway, NJ 08855-1331  
Tel: + 1-908-981-1393  
Fax: + 1-908-981-9667  
mis.custserv@computer.org

IEEE Computer Society  
13, Avenue de l'Aquilon  
B-1200 Brussels  
BELGIUM  
Tel: + 32-2-770-2198  
Fax: + 32-2-770-8505  
euro.ofc@computer.org

IEEE Computer Society  
Ooshima Building  
2-19-1 Minami-Aoyama  
Minato-ku, Tokyo 107  
JAPAN  
Tel: + 81-3-3408-3118  
Fax: + 81-3-3408-3553  
tokyo.ofc@computer.org

Editorial production by Bob Werner

Cover art production Joe Daigle/Studio Productions

Printed in the United States of America by Technical Communication Services

IEEE  
  
COMPUTER  
SOCIETY



# Table of Contents

## Symposium on Field-Programmable Custom Computing Machines — FCCM'97

Introduction.....	ix
Program Committee.....	x
<b>Session 1: Device Architecture</b>	
An FPGA Architecture for DRAM-based Systolic Computations..... <i>N. Margolus</i>	2
Garp: A MIPS Processor with a Reconfigurable Coprocessor..... <i>J. Hauser, J. Wawrzynek</i>	12
A Time-Multiplexed FPGA..... <i>S. Trimberger, D. Carberry, A. Johnson, J. Wong</i>	22
<b>Session 2: Communication Applications</b>	
An FPGA-Based Coprocessor for ATM Firewalls..... <i>J. McHenry, P. Dowd, T. Carrozzi, F. Pellegrino, W. Cocks</i>	30
A Wireless LAN Demodulator in a Pamette: Design and Experience..... <i>T. McDermott, P. Ryan, M. Shand, D. Skellern, T. Percival, N. Weste</i>	40
<b>Session 3: Run Time Reconfiguration</b>	
Incremental Reconfiguration for Pipelined Applications..... <i>H. Schmit</i>	47
Compilation Tools for Run-Time Reconfigurable Designs..... <i>W. Luk, N. Shirazi, P. Cheung</i>	56
A Dynamic Reconfiguration Run-Time System..... <i>J. Burns, A. Donlin, J. Hogg, S. Singh, M. de Wit</i>	66
<b>Session 4: Architectures for Run Time Reconfiguration</b>	
The Swappable Logic Unit: A Paradigm for Virtual Hardware..... <i>G. Brebner</i>	77



The Chimaera Reconfigurable Functional Unit.....	87
<i>S. Hauck, T. Fry, M. Hosler, J. Kao</i>	
<b>Session 5: Architecture</b>	
Computing Kernels Implemented with a Wormhole RTR CCM .....	98
<i>R. Bittner Jr., P. Athanas</i>	
Mapping Applications to the RaPiD Configurable Architecture .....	106
<i>C. Ebeling, D. Cronquist, P. Franklin, J. Secosky, S. Berg</i>	
Defect Tolerance on the Teramac Custom Computer .....	116
<i>B. Culbertson, R. Amerson, R. Carter, P. Kuekes, G. Snider</i>	
<b>Session 6: Performance</b>	
Systems Performance Measurement on PCI Pamette.....	125
<i>L. Moll, M. Shand</i>	
The RAW Benchmark Suite: Computation Structures for General Purpose Computing.....	134
<i>J. Babb, M. Frank, V. Lee, E. Waingold, R. Barua, M. Taylor, J. Kim, S. Devabhaktuni, A. Agarwal</i>	
<b>Session 7: Software Tools</b>	
Automated Field-Programmable Compute Accelerator Design using Partial Evaluation.....	145
<i>Q. Wang, D. Lewis</i>	
FPGA Synthesis on the XC6200 using IRIS and Trianus/Hades (Or, from Heaven to Hell and back again) .....	155
<i>R. Woods, S. Ludwig, J. Heron, D. Trainor, S. Gehring</i>	
High Level Compilation for Fine Grained FPGAs .....	165
<i>M. Gokhale, E. Gomersall</i>	
<b>Session 8: CAD Applications</b>	
Acceleration of an FPGA Router .....	175
<i>P. Chan, M. Schlag</i>	
Fault Simulation on Reconfigurable Hardware .....	182
<i>M. Abramovici, P. Menon</i>	



<b>Session 9: Image Processing Applications</b>	
Automated Target Recognition on SPLASH 2..... <i>M. Rencher, B. Hutchings</i>	192
Real-Time Stereo Vision on the PARTS Reconfigurable Computer..... <i>J. Woodfill, B. Von Herzen</i>	201
Increased FPGA Capacity Enables Scalable, Flexible CCMs: An Example from Image Processing..... <i>J. Greenbaum, M. Baxter</i>	211
<b>Session 10: Arithmetic Applications</b>	
Comparison of Arithmetic Architectures for Reed-Solomon Decoders in Reconfigurable Hardware..... <i>C. Paar, M. Rosner</i>	219
Implementation of Single Precision Floating Point Square Root on FPGAs..... <i>Y. Li, W. Chu</i>	226
<b>Poster Papers</b>	
Datapath-Oriented FPGA Mapping and Placement for Configurable Computing..... <i>T. Callahan, J. Wawrzynek</i>	234
Mapping a Real-Time Video Algorithm to a Context-Switched FPGA..... <i>S. Kelem</i>	236
A Parallel Hardware Evolvable Computer POLYP..... <i>U. Tangen, L. Schulte, J. McCaskill</i>	238
Laser Defect Correction Applications to FPGA Based Custom Computers..... <i>G. Chapman, B. Dufort</i>	240
Speech Recognition HMM Training on Reconfigurable Parallel Processor..... <i>H. Yun, A. Smith, H. Silverman</i>	242
Efficient Implementation of the DCT on Custom Computers..... <i>N. Bergmann, Y. Chung, B. Gunther</i>	244
On Acceleration of the Check Tautology Logic Synthesis Algorithm using an FPGA-based Reconfigurable Coprocessor..... <i>J. Cong, J. Peck</i>	246
<b>Index of Authors</b> .....	249