J.T. McHenry, "Dictionary Search Application on Splash," tech. report, SRC, Bowie, Md., 1991.

J.T. McHenry and A. Kopser, "Keyword Searching on Splash," tech. report, SRC, Bowie, Md., 1991.

Mead Data Central, *LEXIS Quick Reference*, Mead Data Central, New York, 1976.

R. Meier, "Rapid Prototyping of a RISC Architecture for Implementation in FPGAs," *Proc. IEEE Symp. FPGAs for Custom Computing Machines*, CS Press, Los Alamitos, Calif., 1995, pp. 190–196.

P.J. Menchini, "An Introduction to VHDL," in J.P. Mermet, ed., *Fundamentals and Standards in Hardware Description Languages*, Kluwer Academic Publishers, Boston, 1993, pp. 359–384.

B. Miller, "Vital Signs of Identity," *IEEE Spectrum*, Vol. 31, No. 2, Feb. 1994, pp. 22–30.

G. Milne et al., "Realizing Massively Concurrent Systems on the SPACE Machine," *Proc. IEEE Workshop FPGAs for Custom Computing Machines*, CS Press, Los Alamitos, Calif., 1993, pp. 26–33.

L. Moll, J. Vuillemin, and P. Boucard, "High-Energy Physics on DEC PeRLe-1 Programmable Active Memory," *Proc. FPGA95*, ACM, ACM Press, New York, 1995, pp. 47–52.

S. Monaghan and C.P. Cowen, "Reconfigurable Multi-Bit Processor for DSP Applications in Statistical Physics," *Proc. IEEE Workshop FPGAs for Custom Computing Machines*, CS Press, Los Alamitos, Calif., 1993, pp. 103–111.

W. Moore and W. Luk, eds., *FPGAs*, Abingdon EE & CS Books, Abingdon, England, UK, 1992. (Proc., Oxford 1991 Int'l Workshop on Field Programmable Logic and Applications.)

W. Moore and W. Luk, eds., *J. of VLSI Signal Processing*, 1993, (Special Issue on Field-Programmable Gate Arrays.)

W. Moore and W. Luk, eds., *More FPGAs*, Abingdon EE & CS Books, Abingdon, England, UK, 1994. (Proc., Oxford 1993 Int'l Workshop on Field Programmable Logic and Applications.)

Q. Motiwala, "Optimizations for Acyclic Dataflow Graphs for Hardware-Software Codesign," master's thesis, Virginia Polytechnic Inst., Blacksburg, Va., 1994.

Nat'l Library of Medicine, *MEDLARS, The Computerized Literature Retrieval Services of the Nat'l Library of Medicine*, Publication NIH 79-1286, U.S. Dept. of Health, Education and Welfare, Washington, D.C., 1979.

M. Newman, W. Luk, and I. Page, "Constraint-Based Hierarchical Hardware Compilation of Parallel Programs," in R.W. Hartenstein and M.Z. Servít, eds., *Field-Programmable Logic: Architectures, Synthesis, and Applications*. Springer-Verlag, Berlin, 1994, pp. 220–229.

R.J. Offen, *VLSI Image Processing*, McGraw-Hill, New York, 1985.

J.K. Ousterhout, *Tcl and the Tk Toolkit*, Addison-Wesley, Reading, Mass., 1994.

I. Page and W. Luk, "Compiling Occam in FPGAs," in W. Moore and W. Luk, eds., *FPGAs*, Abingdon EE & CS Books, Abingdon, England, UK, 1991, pp. 271–283.

D.L. Perry, *VHDL*, McGraw-Hill, New York, 1991.

D.L. Perry, *VHDL*, McGraw-Hill, New York, 2nd ed., 1994.

W.K. Pratt, *Digital Image Processing*, Wiley, New York, 1978.

D.V. Pryor, M.R. Thistle, and N. Shirazi, "Text Searching on Splash 2," *Proc. IEEE Workshop FPGAs for Custom Computing Machines*, CS Press, Los Alamitos, Calif., 1993, 172–178.

G. Purcell and D. Mar, "SCOUT: Information Retrieval from Full-Text Medical Literature," Knowledge Systems Lab. Report KSL-92-35, Stanford Univ., Palo Alto, Calif., 1992.

G.M. Quénot et al., "A Reconfigurable Compute Engine for Real-Time Vision Automata Prototyping," *Proc. IEEE Workshop FPGAs for Custom Computing Machines*, CS Press, Los Alamitos, Calif., 1994, pp. 91–101.

R. Rachakonda, "Region Detection and Labeling in Real-time Using a Custom Computing Platform," master's thesis, Virginia Polytechnic Inst., Blacksburg, Va., 1994.

F. Raimbault et al., "Fine Grain Parallelism on a MIMD Machine Using FPGAs," *Proc. IEEE Workshop FPGAs for Custom Computing Machines*, CS Press, Los Alamitos, Calif., 1993, pp. 2–9.

N.K. Ratha, A.K. Jain, and D.T. Rover, "Fingerprint Matching on Splash 2," tech. report, Dept. of Computer Science, Michigan State Univ., East Lansing, Mich., Mar. 1994.

N.K. Ratha, A.K. Jain, and D.T. Rover, "Convolution on Splash 2," *Proc. IEEE Symp. FPGAs for Custom Computing Machines*, CS Press, Los Alamitos, Calif., 1995, pp. 204–213.

A. Rosenfeld and A. Kak, *Digital Picture Processing*, 2nd ed., Academic Press, New York, 1982.

G. Salton, *Automatic Text Processing*, Addison-Wesley, Reading, Mass., 1989.

G. Salton and M.J. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, New York, 1983.

D. Sankoff and J. Kruskal, eds., *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, Addison-Wesley, Reading, Mass., 1983.

J. Schlesinger and M. Gokhale, dBC Reference Manual. Tech. Report SRC-TR-92-068, Revision 2, SRC, Bowie, Md., 1993.

H. Schmit et al., "Behavioral Synthesis for FPGA-Based Computing," *Proc. IEEE Workshop FPGAs for Custom Computing Machines*, CS Press, Los Alamitos, Calif., 1994, pp. 125–133.

H. Schmit and D. Thomas, "Implementing Hidden Markov Modelling and Fuzzy Controllers in FPGAs," *Proc. IEEE Symp. FPGAs for Custom Computing Machines*, CS Press, Los Alamitos, Calif., 1995, pp. 214–221.

J. Serra, *Image Analysis and Mathematical Morphology*, Academic Press, London, 1982.

M. Shand, "Flexible Image Acquisition Using Reconfigurable Hardware," *Proc. IEEE Symp. FPGAs for Custom Computing Machines*, CS Press, Los Alamitos, Calif., 1995, pp. 125–134.

M. Shand, P. Bertin, and J. Vuillemin, "Hardware Speedups for Long Integer Multiplication," *ACM Symp. Parallel Algorithms and Architectures*, ACM, ACM Press, New York, 1990, pp. 138–145.

N. Shirazi, "Implementation of a 2-D Fast Fourier Transform on an FPGA-based Computing Platform," master's thesis, Virginia Polytechnic Inst., 1995.

N. Shirazi, A. Walters, and P. Athanas, "Quantitative Analysis of Floating-Point Arithmetic on FPGA-based Custom Computing Machines," *Proc. IEEE Symp. FPGAs for Custom Computing*, CS Press, Los Alamitos, Calif., Apr. 1995, pp. 155–162.

S. Singh, "Architectural Description for FPGA Circuits," *Proc. IEEE Symp. FPGAs for Custom Computing Machines*, CS Press, Los Alamitos, Calif., 1995, pp. 145–154.

S. Singh and P. Bellec, "Virtual Hardware for Graphics Applications Using FPGAs," *Proc. IEEE Workshop FPGAs for Custom Computing Machines*, CS Press, Los Alamitos, Calif., 1994, pp. 49–59.

N. Sitkoff et al., "Implementing a Genetic Algorithm on a Parallel Custom Computing Machine," *Proc. IEEE Symp. FPGAs for Custom Computing Machines*, CS Press, Los Alamitos, Calif., 1995, pp. 180–187.

C. Stanfill and B. Kahle, "Parallel Free-Text Search on the Connection Machine System," *Comm. of the ACM*, Vol. 29, No. 12, 1986, pp. 1229–1239.

J. Stigliani, *Writing SBus Device Drivers*, Sun Microsystems, Inc., Mountain View, Calif., 1990.

Synopsys, Inc., *Design Compiler Reference Manual*, Synopsys, Inc., Mountain View, Calif., 1991.

Synopsys, Inc., *VHDL Compiler Reference Manual*, Synopsys, Inc., Mountain View, Calif., 1991.

Synopsys, Inc., *FPGA Compiler Reference Manual*, Synopsys, Inc., Mountain View, Calif., 1994.

A. Tarmaster, "Median and Morphological Filtering of Images in Real Time Using an FPGA-based Custom Computing Platform," master's thesis, Virginia Polytechnic Inst., Blacksburg, Va., 1994.

Texas Instruments Inc., *The SN74ACT8800 Family Data Manual (SCSS006A)*, Texas Instruments Inc., Dallas, 1988.

Thinking Machines, Inc., *C* Programming Guide*, Thinking Machines, Inc., Cambridge, Mass., 1993.

D.E. Thomas and P.R. Moorby, *The Verilog Hardware Description Language*, Kluwer Academic Publishers, Boston, 1991.

S.M. Trimberger, ed., *Field Programmable Gate Array Technology*, Kluwer Academic Publishers, Boston, 1994.

Reference Manual for the Ada Programming Language, ANSI/MIL-STD-1815A-1983, U.S. Department of Defense, Washington, D.C., Feb. 1983.

L. Uhr, ed., *Parallel Computer Vision*, Academic Press, New York, 1987.

M. van Daalen, P. Jeavons, and J. Shawe-Taylor, "A Stochastic Neural Architecture That Exploits Dynamically Reconfigurable FPGAs," *Proc. IEEE Workshop FPGAs for Custom Computing Machines*, CS Press, Los Alamitos, Calif., 1993, pp. 202–212.

D.E. Van den Bout, "The Anyboard: Programming and Enhancements," *Proc. IEEE Workshop FPGAs for Custom Computing Machines*, CS Press, Los Alamitos, Calif., 1993, pp. 68–78.

G. VanDerWal and P. Burt, "A VLSI Pyramid Chip for Multiresolution Image Analysis," *Int'l J. of Computer Vision*, Vol. 8, No. 3, 1992, pp. 177–189.

R. Vogt, *Automatic Generation of Morphological Set Recognition Algorithms*, Springer-Verlag, New York, 1989.

J. Vuillemin et al., "Programmable Active Memories: Reconfigurable Systems Come of Age," *IEEE Trans. VLSI Systems*, to be published in Mar. 1996.

M. Wazlowski et al., "PRISM II: Compiler and Architecture," *Proc. IEEE Workshop FPGAs for Custom Computing Machines*, CS Press, Los Alamitos, Calif., 1993, pp. 9–17.

J.H. Wegstein, *An Automated Fingerprint Identification System*, Special Publication 500–89, Nat'l Bureau of Standards, Washington, D.C., 1982.

R. Wieler, Z. Zhang, and R. McLeod, "Emulating Static Faults Using a Xilinx Based Emulator," *Proc. IEEE Symp. FPGAs for Custom Computing Machines*, CS Press, Los Alamitos, California, 1995, pp. 110–115.

M. Wirthlin and B. Hutchings, "A Dynamic Instruction Set Computer," *Proc. IEEE Symp. FPGAs for Custom Computing Machines*, CS Press, Los Alamitos, Calif., 1995, pp. 99–107.

M.J. Wirthlin, B.L. Hutchings, and K.L. Gilson, "The Nano Processor: A Low Resource Reconfigurable Processor," *Proc. IEEE Workshop FPGAs for Custom Computing Machines*, CS Press, Los Alamitos, Calif., 1994, pp. 23–31.

D. Wo and K. Forward, "Compiling to the Gate Level for a Reconfigurable Co-Processor," *Proc. IEEE Workshop FPGAs for Custom Computing Machines*, CS Press, Los Alamitos, Calif., 1994, pp. 147–155.

L.F. Wood, "High Performance Analysis and Control of Complex Systems Using Dynamically Reconfigurable Silicon and Optical Fiber Memory," *Proc. IEEE Workshop FPGAs for Custom Computing Machines*, CS Press, Los Alamitos, Calif., 1993, pp. 132–142.

Xilinx, Inc., *The Programmable Gate Array Data Book*, Xilinx, Inc., San Jose, Calif., 1993.

Xilinx, Inc., *The XC4000 Data Book*, Xilinx, Inc., San Jose, Calif. 1994.

C.-C. Yeh, C.-H. Wu, and J.-Y. Juang, "Design and Implementation of a Multicomputer Interconnection Using FPGAs," *Proc. IEEE Symp. FPGAs for Custom Computing Machines*, CS Press, Los Alamitos, Calif., 1995, pp. 56–60.

Index

Ada, 36, 50
AFIS, *see* Automatic Fingerprint Identification System
Algotronix, Ltd., 4, 7, 95
Analytic Instruments Inc., 24
Aptix, 181
arch, fingerprint, 123
Array Board, 12, 13, 19
  architecture, 16–17
  implementation, 25–30
  programming, 29
Atmel Corp., 4
attached processors, 6, 169, 171
Automatic Fingerprint Identification System, 119

band-pass pyramids, 145
Bank Register, 21
Batley's formula, 119
broadcast, 17
Brown University, 3, 95, 183
Burroughs Corp.
  B1700, 2, 174
bypass mode, 25

C*, 80
Center for Computing Sciences, *see* Supercomputing Research Center
CERN, 177
CHAMP, 6, 174
CLB, *see* Configurable Logic Block
clock, 18
  free-running, 57
  hardware, 24
  implementation, 24
  regulation of system, 18
  setting frequency, 58
  SIMD, 57
  single-step, 18
  software, 24, 57
  variable frequency, 24
`comp.arch.fpga` newsgroup, 3
compression, 177
Concurrent Logic, Inc., 4
  CLi6005 FPGA, 37
Configurable Logic Block, 4
  flip-flops, 169
configuration register, 30
Control Element, 20
  entity declaration, 62
  implementation, 28
  programming view, 56–57
control/status register, *see* CSR
convolutional filtering, 177
coprocessors, 5–6, 169, 173–174
core point, fingerprint, 123
corner turning, 24
Cray Research
  YMP processor, 2
cross-correlation example, 81
crossbar, 16–17, 181
  configuration of, 30, 68–69

crossbar *continued*
dataflow modes, 170
implementation, 28–29
programming view, 56
CSR, 25

data-driven model, 175
Datacube MaxVideo 200, 162
dbC, 49, 77–95, 174, 176
De La Rue Printrak, 119
DEC, *see* Digital Equipment Corp.
Department of Defense, 180
Development Board, 19, 57
implementation, 21
device driver, 74–75
diagnostic software, 75–76
Digital Equipment Corp., *see* Paris research lab, DEC's
digital signal processor, 172
dilation, 146
direct memory access, *see* DMA
discrete Fourier transform, 147
DMA, 12, 19
DMA Channel
daughterboard, 20
implementation, 23
DNA sequence, *see* sequence comparison
DoD, *see* Department of Defense
double loop, fingerprint, 123
DSP, *see* digital signal processor

edge detection, 16
`edif2xnf`, 53, 56, 70
edit distance, 98
dynamic programming algorithm, 98
modular encoding, 105
erosion, 146

FBI, *see* Federal Bureau of Investigation
Federal Bureau of Investigation, 118, 183
Field Programmable Gate Array, 2, 4–5, 11, 20, 37
architecture, 172
fingerprint
matching algorithm, 125–128
performance, 137–139
registration, 126
FIR filter, 186–189
FPGA, *see* Field Programmable Gate Array
Futurebus+, 12, 19, 181

Ganglion, 5
Gaussian pyramid, 145, 154
generic SIMD instructions, 82, 84
genetic database search, *see* sequence comparison
global OR signal, 18, 43
global tri-state signal, 28, 54
Gordon Bell prize
1989, 34
GTS signal, *see* global tri-state signal

handshake register, 30, 58
hard macros, 12, 52, 61
Henry formula, 117
high-pass filters, 145
host computer
programming view, 57–58
Hough transform, 2, 147
Human Genome Initiative, 97

IDA, *see* Institute for Defense Analyses
Identification register, 25
IEEE, 3, 50
image expansion, 158
image processing, 141–163, 177
fingerprint, 119
performance, 159–162
image pyramid, 153
image pyramid generation, 153
image subtraction, 158
Input Output Block, 4
exploiting flip-flops, 56, 187
Institute for Defense Analyses, 183
instruction set synthesis, 84
Intel Corp.
8086 processor, 173
Interface Board, 12, 19
architecture, 17–18
implementation, 21–25
memory, 24
programming view, 57
interrupt register, 30
interrupts, 24
IOB, *see* Input Output Block

Laplacian pyramid, 146, 157
LDG, 32, 46, 78, 179
LED register, 26
LEXIS, 110
`libsplash.a`, *see* runtime library
Light-Emitting Diodes, *see* LED register
linear data path, 13–14, 20

Lockheed Sanders, 174
Logic Description Generator, *see* LDG
logic synthesis, 6, 48
Logica, 119
loop, fingerprint, 123
low-pass filter, 144
low-pass pyramids, 145

macro instructions, 92–94
mask register, 30
mathematical morphology, 146
median filtering, 146, 150–153
MEDLARS, 110
memory
  architecture of, 44, 167–168
  host access to, 21, 28
  initialization, 69
  mapped into address space, 58
Michigan State University, 183
minutia, 118, 123
  matching, 126
Model Technologies, Inc., 182
MPL, 80

National Cancer Institute
  Dept. of Mathematical Biology, 180
National Center for Biotechnology Information, 183
National Semiconductor Corp., 4
NCI, *see* National Cancer Institute
nearest-neighbor communication, 88
NEC Information Systems, 119
North American Morpho, 119

`opPar`, *see* generic SIMD instructions
Oxford University, 95

P-NAC, 31, 97
PAM, *see* Paris research lab, DEC's
Paris research lab, DEC's, 166, 174
  PeRLe, 2
  PeRLe-1, 6, 171, 177
Paris research lab, DEC's
  PeRLe-0, 6
pattern recoginition systems, 121
PeRLe, *see* Paris research lab, DEC's
physical mapping, 48
placement and routing, 6
`poly` data type, 81
Princeton Nucleic Acid Comparator, *see* P-NAC
Princeton University, 31
PRISM, 3, 5, 183
Processing Element, 20
  entity declaration, 61
  implementation, 26–28
  programming, 24–25
  programming view, 56–57
Processor-in-Memory (PIM), 79
protein sequence, *see* sequence comparison
PRS, *see* pattern recoginition systems
pyramid, 145, *see* Gaussian pyramid, Laplacian pyramid

Quick and Dirty Board, *see* Development Board
Quickturn Design Systems, Inc., 178

rapid prototyping, 177
RBus, 14, 20
  data register, 58
readback, 24–25, 29
  role in symbolic debugging, 58, 169
real-time control, 177
reduction operation, 80, 89–91
reset, 25, 29
ridge, fingerprint, 123
`robocop`, 76
RSA decryption, 2, 166
RSA encryption, 166
runtime library, 54, 73

SBus, 12, 19
  Adapter Board, 19
  address space, 18, 21
  choice of, 38
  DMA performance, 75
  slave accesses, 22
sequence comparison, 15, 100–104, 111, 182
  bidirectional algorithm, 100, 103
  dbC example, 94–95
  performance, 107
SIMD Bus, 13, 20
  data register, 58
SIMD model, 11, 13, 17
single-instruction multiple-data, *see* SIMD model
size estimation, *see* utilization
Sobel operators, 145
SPARCstation 2, 12, 19, 38
special-purpose devices, 5

Splash 1, 6, 179
  architecture, 31–32
Splash 2, 179
`Splash 2 Library`, 51, 61
Splash 2 simulator, 51, 66–70
  configuring, 67–68
SRC, *see* Supercomputing Research Center
Sun Microsystems, Inc., 12, 19, 38
Supercomputing Research Center, 4
Synopsys, Inc., 182
  Design Compiler, 53, 70
  FPGA Compiler, 53, 71, 168, 182
systolic, 13

`T2` debugger, 55, 72–73
tags, 14
  valid data, 57
Tcl language, 55
TERASYS, 79, 181, 184
Texas Instruments
  crossbar chip, 28, 41, 181
text searching
  16-bit approach, 115
  8-bit implementation, 113–114
  algorithm, 111–112
  general approach, 111
  performance, 114, 116
Thinking Machines Corp.
  CM-2, 2, 81, 183
  CM-2X, 5
timing analysis, 49
tolerance box, 128
`trigger` debugger, 32
`tsdb` debugger, 55, 76

utilization, 56

valley, fingerprint, 123
Verilog, 51
VHDL, 36–37, 49–51, 182
  choice of, 36, 45
  history of, 50
  pipelining in, 189
  synchronous processes in, 187
VHSIC initiative, 47, 50
Viewlogic, 32
Virginia Polytechnic Institute and State University, 183
virtual computer, 3
VMEbus, 34, 39
VTSplash, 142

whirl, fingerprint, 123

X0, 13, 17
  purpose, 43
  use in dbC, 86, 89
  use in fingerprint matching, 132–133
XACT editor, 32
XBLOX, 168
Xilinx, 2, 4, 7, 11
  `apr` tool, 33
  choice of, 38
  Netlist Format (XNF), 53
  XC3090 FPGA, 32, 182
  XC4010 FPGA, 4, 11–12, 182
XL, 15
  entity declaration, 63
  implementation, 23–24
  purpose, 43
  use in dbC, 86
  use in text search, 111
`xnfer`, 54, 56, 71
XR, 15
  implementation, 23–24
  purpose, 43
  use in text search, 112

Contributors

A. Lynn Abbott, Bradley Department of Electrical Engineering, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061. 703-231-4472

Jeffrey M. Arnold, Center for Computing Sciences, 17100 Science Drive, Bowie, Maryland 20715. 301-805-7479

Peter Athanas, Bradley Department of Electrical Engineering, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061. 703-231-7010

Duncan A. Buell, Center for Computing Sciences, 17100 Science Drive, Bowie, Maryland 20715. 301-805-7372

Maya Gokhale, David Sarnoff Research Center, CN 5300, Princeton, New Jersey 08543. 609-734-3119

Dzung T. Hoang, Department of Computer Science, Duke University, Durham, North Carolina 27706. 919-660-6598

Anil Jain, Department of Computer Science, Michigan State University, East Lansing, Michigan 48824. 517-353-5150

Walter J. Kleinfelder, Center for Computing Sciences, 17100 Science Drive, Bowie, Maryland 20715. 301-805-7355

Daniel V. Pryor, Center for Computing Sciences, 17100 Science Drive, Bowie, Maryland 20715. 301-805-7407

Nalini Ratha, Department of Computer Science, Michigan State University, East Lansing, Michigan 48824. c/o A. Jain 517-353-5150

Diane Rover, Department of Electrical Engineering, Michigan State University, East Lansing, Michigan 48824. 517-353-7735

Nabeel Shirazi, Bradley Department of Electrical Engineering, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061. c/o P. Athanas 703-231-7010

Mark R. Thistle, Center for Computing Sciences, 17100 Science Drive, Bowie, Maryland 20715. 301-805-7413

IEEE Computer Society Press Editorial Board

Advances in Computer Science and Engineering

The IEEE Computer Society Press Advances Board seeks manuscripts that describe new and significant advances in computer science and engineering. Although immediate application is not necessary, ultimate application to advanced computing systems is an important quality. Publications represent technically substantive and clear expositions of innovative ideas.

Additional Advances Board Titles

A Probabilistic Analysis of Test-Response Compaction
Slawomir Pilarski and Tiko Kameda

The Cache Coherence Problem in Shared-Memory Multiprocessors: Software Solutions
Igor Tartalja and Veljko Milutinović

The Cache Coherence Problem in Shared-Memory Multiprocessors: Hardware Solutions
Igor Tartalja and Veljko Milutinović

Advanced Multimicroprocessor Bus Architectures
Janusz Zalewski

IEEE Computer Society Press Publications

The world-renowned Computer Society Press publishes, promotes, and distributes a wide variety of authoritative computer science and engineering texts. These books are available in two formats: 100 percent original material by authors preeminent in their field who focus on relevant topics and cutting-edge research, and reprint collections consisting of carefully selected groups of previously published papers with accompanying original introductory and explanatory text.

Submission of proposals: For guidelines and information on CS Press books, send e-mail to csbooks@computer.org or write to the Acquisitions Editor, IEEE Computer Society Press, P.O. Box 3014, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720-1314. Telephone +1 714-821-8380. FAX +1 714-761-1784.

IEEE Computer Society Press Proceedings

The Computer Society Press also produces and actively promotes the proceedings of more than 130 acclaimed international conferences each year in multimedia formats that include hard and softcover books, CD-ROMs, videos, and on-line publications.

For information on CS Press proceedings, send e-mail to csbooks@computer.org or write to Proceedings, IEEE Computer Society Press, P.O. Box 3014, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720-1314. Telephone +1 714-821-8380. FAX +1 714-761-1784.

Additional information regarding the Computer Society, conferences and proceedings, CD-ROMs, videos, and books can also be accessed from our web site at www.computer.org.

3/21/96

Splash 2

FPGAs in a Custom Computing Machine

edited by Duncan A. Buell, Jeffrey M. Arnold, and Walte

Details the complete Splash 2 project–the hardware and software systems, their architecture and implementation, and the design process by which the architecture evolved from an earlier version machine. In addition to the description of the machine, this book explains why Splash 2 was engineered. It illustrates several applications in detail, allowing you to gain an understanding of the capabilities and the limitations of this kind of computing device.

The Splash 2 program is significant for two reasons. First, it is part of a complete computer system that achieves supercomputer like performance on a number of different applications. The second significant aspect is that this large system is capable of performing real computations on real problems. In order to understand what happens when the application programmer designs the processor architecture of the machine that executes his programs, it is necessary to see the system as a whole. This book looks in-depth at one of the handful of data points in the design space of this new kind of machine.

Contents:

- Custom Computing Machines: An Introduction
- The Architecture of Splash 2
- Hardware Implementation
- Splash 2: The Evolution of a New Architecture
- Software Architecture
- Software Implementation
- A Data Parallel Programming Model
- Searching Genetic Databases on Splash 2
- Text Searching on Splash 2
- Fingerprint Matching on Splash 2
- High-Speed Image Processing with Splash 2
- The Promise and the Problems
- An Example Application

Published by the IEEE Computer Society Press
10662 Los Vaqueros Circle
P.O. Box 3014
Los Alamitos, CA 90720-1314

ISBN 0-8186-7413-X
90000>
9 780818 674136

IEEE Computer Society Press Order Number BP07413
Library of Congress Number 95-47397
ISBN 0-8186-7413-X

Attachment 5C

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

Splash 2 : FPGAs in a custom computing machine

Author:	Duncan A Buell; Jeffrey M Arnold; Walter J Kleinfelder
Publisher:	Brussels : IEEE Computer Society Press, 1996.
Edition/Format:	Print book : English View all editions and formats
Rating:	(not yet rated) 0 with reviews - Be the first.

Get a Copy

Find a copy in the library

Find a copy in the library

Enter your location: mit Find libraries
Submit a complete postal address for best results.

Displaying libraries 1-6 out of 83 for all 10 editions (77 Massachusetts Ave, Cambridge, MA 02139, USA)

Show libraries holding just this edition

« First ‹ Prev 1 2 3 Next › Last »

	Library	Held formats	Distance	
1.	MIT Libraries **Massachusetts Institute of Technology Libraries** Cambridge, MA 02139 United States	Book	< 1 mile MAP IT	Library info Ask a librarian Add to favorites
2.	Boston University Libraries **Mugar Memorial Library** Boston, MA 02215 United States	Book	2 miles MAP IT	Library info Ask a librarian Add to favorites
3.	Harvard University Cambridge, MA 02138 United States	Book	2 miles MAP IT	Library info Add to favorites
4.	Worcester Polytechnic Institute **WPI; George C. Gordon Library** Worcester, MA 01609 United States	Book	37 miles MAP IT	Library info Ask a librarian Add to favorites
5.	Trinity College Library Hartford, CT 06106 United States	Book	92 miles MAP IT	Library info Search at this library Add to favorites
6.	University of Vermont **Bailey/Howe Library** Burlington, VT 05405 United States	Book	181 miles MAP IT	Library info Ask a librarian Add to favorites

« First ‹ Prev 1 2 3 Next › Last »

Details

Document Type:	Book
All Authors / Contributors:	Duncan A Buell; Jeffrey M Arnold; Walter J Kleinfelder Find more information about: Duncan A Buell Go
OCLC Number:	989612266
Description:	XIV, 205 pages
Responsibility:	ed. by Duncan A. Buell, Jeffrey M. Arnold and Walter J. Kleinfelder.

- Reviews

User-contributed reviews

Add a review and share your thoughts with other readers. Be the first.

- Tags

Add tags for "Splash 2 : FPGAs in a custom computing machine". Be the first.

+ Linked Data

Attachment 5D

Barton MIT Libraries' Catalog

MIT Libraries

Search Full Catalog:
- Basic
- Advanced

Search only for:
- Conferences
- E-resources
- Journals
- MIT Theses
- Reserves
- more...

- Your Account
- Help with Your Account
- Your Bookshelf
- Previous Searches

Ask Us! Other Catalogs Help

Full Record

Permalink for this record: http://library.mit.edu/item/000791622

Results List | Add to Bookshelf | Save/Email

Choose format: Standard | Citation | MARC tags

Record 1 out of 1

Title	Splash 2 : FPGAs in a custom computing machine / Duncan A. Buell, Jeffrey M. Arnold, Walter J. Kleinfelder, editors.
Shelf Access	Find it in the library/Request item
Shelf Location	Barker Library - Stacks | QA76.8.S65.B84 1996
Published	Los Alamitos, Calif. : IEEE Computer Society Press, c1996.
Description	xiv, 205 p. : ill. ; 26 cm.
Format	Book
Bibliography	Includes bibliographical references (p. 190-200) and index.
Subject	Splash 2 (Computer)
	Electronic digital computers -- Design and construction.
Other Author	Buell, Duncan A.
	Arnold, Jeffrey M.
	Kleinfelder, Walter J.
Other Title	Splash two.
ISBN	081867413X (paper)
Local System Number	000791622

Basic Search of Full Catalog

Search type:
- Keyword
- Title begins with...
- Title Keyword
- Author (last name first)
- Author Keyword
- Call Number begins with...
- ----- Scroll down for more choices -----

Search for:

Search

Attachment 5E

Full Record

Permalink for this record: http://library.mit.edu/item/000791622

Results List | Add to Bookshelf | Save/Email

Choose format: Standard | Citation | MARC tags

Record 1 out of 1	
FMT	BK
LDR	01000cam 2200301 a 45q0
003	MCM
005	20010609000235.0
008	961003s1996 caua b 001 0 eng
010	|a 95047397
020	|a 081867413X (paper)
035	|a MITb10791622
035	|a (OCoLC)33439519
040	|a DLC |c DLC |d C#P |d MYG
05000	|a QA76.8.S65 |b B84 1996
08200	|a 004.2/2 |2 20
24500	|a Splash 2 : |b FPGAs in a custom computing machine / |c Duncan A. Buell, Jeffrey M. Arnold, Walter J. Kleinfelder, editors.
2463	|a Splash two
260	|a Los Alamitos, Calif. : |b IEEE Computer Society Press, |c c1996.
300	|a xiv, 205 p. : |b ill. ; |c 26 cm.
504	|a Includes bibliographical references (p. 190-200) and index.
650 0	|a Splash 2 (Computer)
650 0	|a Electronic digital computers |x Design and construction.
7001	|a Buell, Duncan A.
7001	|a Arnold, Jeffrey M.
7001	|a Kleinfelder, Walter J.
CAT	|a CONV |b 00 |c 20010620 |l MIT01 |h 1544
CAT	|a lti0904 |b 00 |c 20090523 |l MIT01 |h 2240
049	|a MYGG
910	|a tn961003
949	|a e |p 39080013873036
PST0	|0 Z30 |1 000791622000010 |b ENG |c STACK |o BOOK |d 01 |y 00000 |f N |r MIT60-000797503 |n 0 |h QA76.8.S65.B84 1996 |a MCM |3 Book |4 Barker Library |5 Stacks |6 60 Day Loan |p Avail
LDR	nx 22 zn 4500
008	0106230u 0 4 uu 1
004	000791622
8520	|a MCM |b ENG |c STACK |h QA76.8.S65.B84 1996 |z
001	000791622
SFX01	|s 0-0-0-7-9-1-6-2-2 |l MIT01 |9 000 |z ~~~~~~~~~ |p Avail |f 000
SYS	000791622

Basic Search of Full Catalog

Search type: Search for:

Keyword
Title begins with...
Title Keyword
Author (last name first)
Author Keyword
Call Number begins with...
----- Scroll down for more choices -----

Search

Attachment 6A

IEEE TRANSACTIONS ON

VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS

A JOINT PUBLICATION OF THE IEEE CIRCUITS AND SYSTEMS SOCIETY
THE IEEE COMPUTER SOCIETY
THE IEEE SOLID-STATE CIRCUITS SOCIETY

JUNE 1997 VOLUME 5 NUMBER 2 IEVSE9 (ISSN 1063-8210)

PAPERS

Pipelined H-Trees for High-Speed Clocking of Large Integrated Systems in Presence of Process Variations .... M. Nekili, G. Bois, and Y. Savaria 161
Scheduling Tests for VLSI Systems Under Power Constraints .... R. M. Chou, K. K. Saluja, and V. D. Agrawal 175
Hierarchical Interconnection Structures for Field Programmable Gate Arrays .... Y.-T. Lai and P.-T. Wang 186
Unifiable Scheduling and Allocation for Minimizing System Cycle Time .... S. C.-Y. Huang and W. H. Wolf 197
VLSI Array Algorithms and Architectures for RSA Modular Multiplication .... Y.-J. Jeong and W. P. Burleson 211
An Architectural Co-Synthesis Algorithm for Distributed, Embedded Computing Systems .... W. H. Wolf 218

TRANSACTIONS BRIEFS

VLSI Compressor Design with Applications to Digital Neural Networks.................. D. Zhang and M. I. Elmasry 230
Diagnosis and Correction of Multiple Logic Design Errors in Digital Circuits .... P.-Y. (Emerald) Chung and I. N. Hajj 233
Design of a 32 b Monolithic Microprocessor Based on GaAs HMESFET Technology .... C.-K. V. Tien, K. Lewis, H. J. Greub, T. Tsen, and J. F. McDonald 238
Correction to "Control-Flow Versus Data-Flow-Based Scheduling: Combining Both Approaches in an Adaptive Scheduling System" .... R. A. Bergamaschi, S. Raje, I. Nair, and L. Trevillyan 243

CALLS FOR PAPERS

International Conference on Next Decades of High Technologies 244
Workshop on Enhancing Success-Oriented E.Q. and I.Q. for Students.. 245
International Symposium on Low Power Electronics and Design 246

INFORMATION FOR AUTHORS

The IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS is published quarterly. Contributed papers may be of a tutorial nature. Transactions Briefs include contributions of the following nature: corrections; rebuttals of previous briefs; comments on published results; discussion of experiences using published results; conjectures; posing of new problems; announcement of new results and other contributions written in the format of a correspondence. Research papers must be of original contributions and must not duplicate descriptions or derivations available elsewhere. The author should limit paper length whenever this can be done without impairing quality.

If a contribution is of sufficient general interest to the entire IEEE membership, it will be recommended for publication in the PROCEEDINGS OF THE IEEE. Submission of a manuscript manifests the fact that it has been neither coprighted, published, nor submitted for publication elsewhere, unless otherwise so stated by the author.

To avoid delay, please be guided by the following suggestions.

A. Process for Submission of a Technical Paper

1) Send to the Editor-in-Chief a postscript version of your manuscript to (http://microsys6.engr.utk.edu/~tvlsi), seven copies of y[illegible] manuscript, each copy complete with illustrations, abstract, and indexterms.
2) Enclose a signed copyright form with your manuscript. (You may reproduce the copyright form.)
3) Enclosed with each manuscript, on a separate page five to ten index terms (key phrases). These terms should be alphabetized, relative[illegible] independent (coordinate index terms), and as a group should optimally characterize the paper.
4) Enclose originals for the illustrations (including tables) in the style described below. Alternatively, good quality copies may be sent initially, with the originals ready to be sent immediately upon acceptance of the paper.
5) Enclose a separate page giving your preferred address for correspondence and return of page proofs.
6) Enclose a technical biography and photograph of each author for a paper (not a Transactions Brief), or be ready to supply these upon acceptance of a paper.
7) The referee process assumes the anonymity of the reviewers of your paper. It is also possible to provide a review in which the author's identity is also kept from the reviewers. Should you wish to take advantage of this provision, please make your desires explicit in this regard in your cover letter to the Editor-in-Chief. In this case, make sure that your name appears only on a removable cover page.
8) If the manuscript has been presented, published, or submitted for publication elsewhere, please so inform the Editor-in-Chief. Our primary objective is to publish technical material not available elsewhere, but on occasion we publish papers of unusual merit that have appeared or will appear before other audiences.

B. Style for Manuscript

1) The manuscript should be typewritten or printed double-spaced, one side only, using a font size of 11 points or larger. (Good office-duplicated copies are acceptable.)
2) Provide an informative 100-250 word abstract, at the head of the manuscript, to appear with the paper.
3) Provide a separate double-spaced sheet using all footnotes, beginning with "Manuscript received ______________" continuing with an "Acknowledgment of financial support" and "Affiliation of author."
4) References must appear as a separate bibliography at the end of the paper, with items referred to by numerals in square brackets, e.g., [12]. References should be complete and in IEEE style (see item 6).
   Style for papers: Author, first initials followed by last name, title, volume, inclusive page numbers, month, year.
   Style for books: Author, title, location of publisher, year, chapter, or page numbers (if desired).
5) Provide a separate sheet listing all figure captions, in proper IEEE style, e.g., "Fig. 6. The final 32-bit adder design."
6) For further information, see the booklet, *Information for IEEE Transactions and Journal Authors*, available from the IEEE Operations Center, Transactions/Journal Department, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331.

C. Style for Illustrations

1) Originals for illustrations should be sharp, noise-free, and of good contrast. We regret we cannot provide drafting or art service.
2) Line drawings should be in India ink or drafting cloth, paper, or board. Use only 8-1/2 in by 11-in size sheets, to simplify handling of the manuscript.
3) On graphs, show only the coordinate axes, or at most the major grid lines, to avoid a dense hard-to-read result.
4) All lettering should be large enough to permit legible reduction of the figure to column width, perhaps as much as 4:1. Typing on figures is not acceptable.
5) Photographs should be glossy prints, of good contrast and gradation, and any reasonable size.
6) Identify each original on the back, or at the bottom of front, and indicate the author's name.
7) Note B-5) above. Captions lettered on figures will be blocked out in reproduction, in favor of typeset captions.

D. Final Manuscripts in Electronic Form

If a manuscript is accepted for publication, the author will be asked to supply an electronic or magnetic form of the manuscript via email disk, or tape, and two matching hard copies. The IEEE can process most software, but not page layout programs. Do not send postscript files

E. Page Charges

After a manuscript has been accepted for publication, the author's company or institution will be approached with a request to pay a charge of $110 per page. Payment of page charges for this IEEE TRANSACTIONS is not obligatory; nor is it a prerequisite for publication. The author will receive 100 reprints (without covers) free only if the page charge is honored. Detailed instructions will accompany the page proofs.

F. Copyright

It is the policy of the IEEE to own the copyright to the technical contributions it publishes on behalf of the interests of the IEEE, its authors, and their employers, and to facilitate the appropriate reuse of this material by others. To comply with the U.S. Copyright Law, authors are required to sign an IEEE copyright transfer form before publication. This form, a copy of which appears in the March 1997 issue of this TRANSACTIONS returns to authors and their employers full rights to reuse their material for their purposes. Authors must submit a signed copy of this form with their manuscript.

IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS

The Circuits and Systems Society is an association of IEEE members with professional interest in the field of circuits and systems theory. All members of the IEEE are eligible for membership in the Society upon payment of the annual Society membership fee of $7.00. The Computer Society is an association of people with professional interest in the field of computers. All members of the IEEE are eligible for membership in the Computer Society upon payment of the annual Society membership fee of $29.00. For information on joining, write to the IEEE Computer Society, 1730 Massachusetts Avenue, NW, Washington, DC 20036-1903. The annual subscription fee for members of either society is $18.00. For information on joining, write to the IEEE at the address below. *Member copies of Transactions/Journals are for personal use only.*

IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS (ISSN 1063-8210) is published quarterly by The Institute of Electrical and Electronics Engineers, Inc. Responsibility for the contents rests upon the authors and not upon the IEEE, the Society/Council, or its members. **IEEE Corporate Office:** 345 East 47 Street, New York, NY 10017-2394. **IEEE Operations Center:** 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331. **NJ Telephone:** 908-981-0060. **Price/Publication Information:** Individual copies: IEEE Members $10.00 (first copy only), nonmembers $20.00 per copy. (Note: Add $4.00 postage and handling charge to any order from $1.00 to $50.00, including prepaid orders.) Member and nonmember subscription prices available upon request. Available in microfiche and microfilm. **Copyright and Reprint Permissions:** Abstracting is permitted with credit to the source. Libraries are permitted to photocopy for private use of patrons, provided the per-copy fee indicated in the code at the bottom of the first page is paid through the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923. For all other copying, reprint, or republication permission, write to Copyrights and Permissions Department, IEEE Publications Administration, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331. Copyright © 1997 by The Institute of Electrical and Electronics Engineers, Inc. All rights reserved. Periodicals Postage Paid at New York, NY and at additional mailing offices. **Postmaster:** Send address changes to IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, IEEE, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331. GST Registration No. 125634188. Printed in U.S.A.

Pipelined H-Trees for High-Speed Clocking of Large Integrated Systems in Presence of Process Variations

Mohamed Nekili, Guy Bois, and Yvon Savaria, *Member, IEEE*

***Abstract*—This paper addresses the problem of clocking large high-speed digital systems, as well as deterministic skew modeling, a related problem. A conventional method for clocking a large digital system is to use a set of metallic lines organized as a tree. This method is limited by the bandwidth of the clock network. Another limitation of existing solutions is that available skew models do not directly take into account process variations. In order to provide a reliable skew model, and to avoid the frequency limitation, we propose a novel approach that distributes the clock with an H-tree, whose branches are composed of minimum-sized inverters rather than metal. With such a structure, we obtain the highest clocking rate achievable with a given technology. Indeed, clock rates around 1 GHz are possible with a 1.2 μm CMOS technology. From the skew modeling standpoint, we derive an analytic expression of the skew between two leaves of the H-tree, which we consider to be the difference in root-to-leaf delay pairs. The skew upper bound obtained has an order of complexity which, with respect to the H-tree size D, is the same as the one that may be derived from the Fisher and Kung model for both side-to-side and neighbor-to-neighbor communications, i.e., a $\Omega(D^2)$, whereas, the Steiglitz and Kugelmass probabilistic model predicts $\Theta(D \times \sqrt{\mathrm{Log}\, D})$. In an H-tree implemented with metallic lines, the leaf-to-leaf skew is obviously bounded by the delay between the root and the leaves. However, with the logic based H-tree proposed in this paper, we arrive at a nonobvious result, which states that the leaf-to-leaf skew grows faster than the root-to-leaf delay in presence of a uniform transistor time constant gradient. This paper also proposes generalizations of the skew model to 1) the case of chips in a wafer subject to a smooth, but nonuniform gradient and 2) the case of H-tree configurations mixing logic and interconnections; in this respect, this paper covers the H-tree configurations based on the combination of logic and interconnections.**

***Index Terms*—H-tree, high-speed clocking, pipelining, process variations, skew.**

I. Introduction

THE evolution of VLSI chips toward larger die sizes and faster clock speeds makes clock design an increasingly important issue. A striking example of what can be accomplished with aggressive clock design is the DEC alpha chip [1], designed to operate at more than 200 MHz. At such speeds, clock skew becomes a very significant problem. Available literature dealing with skew [2]–[8], [10], [11] approaches the problem both from deterministic and probabilistic standpoints.

In the deterministic approaches, Friedmann and Powell [6] emphasize the use of a hierarchical clock distribution, while others [2], [3], [8], [11] suggest the length equalization of the different paths followed by the clock throughout the circuit. Shoji [5] suggests an approach that guarantees a symmetry between paths that contribute to propagate "0" and "1." This symmetry ensures proper operation despite some types of process variations [5]. Except for the work of Fisher and Kung [4], which provides bounds on skew, the other authors do not deal with the analytic modeling of system skew.

In the probabilistic approaches, Kugelmass and Steiglitz [7] consider the delay of a clock signal along a given path as a sum of delays along path segments, each of these segments behaving according to a probabilistic law. Then, by assuming independence between these delays, the total delay, as well as the skew, can then be described by a normal law. By assuming independence and the linearity of delay with line length, their approach becomes an oversimplification of the reality. Other authors [10] consider the skew as a dispersion in the physical parameters of a circuit (e.g., geometrical dimensions) and in the process (e.g., sensitivity to temperature).

The work that is most directly related to that presented in this paper is the work of Fisher and Kung [4]. These authors have developed two deterministic skew models (the difference model and the summation model), from which they determined bounds on skew. However, these models do not directly refer to a process variation model. The difference model tends to be unrealistically optimistic, whereas, under the summation model, Fisher and Kung reached a pessimistic result which states that, from a skew standpoint, synchronous systems are not feasible with large two-dimensional arrays.

In order to avoid the frequency limitation when using metallic lines, we propose a logic-based H-tree structure that provides the highest clocking rate achievable with a given technology in Section II. To provide a reliable skew model, Section III suggests a model based on delay differences combined with a model of electrical variations in the process parameters. Under this model, we derive an analytic expression of the skew between any leaf pair, which we consider to be the difference in root-to-leaf delay pairs. Even though the model of electrical variations described in this paper assumes

Manuscript received March 24, 1994; revised September 22, 1995. This work was supported by the INI (Institut National de formation en Informatique) in Algiers, CIDA (Canadian International Development Agency), and strategic grant and operating grants from the Natural Sciences and Engineering Research Council of Canada.

The authors are with the Department of Electrical Engineering, Ecole Polytechnique of Montréal, Station "Centre-Ville," Montréal, P.Q. H3C 3A7, Canada.

Publisher Item Identifier S 1063-8210(97)01949-5.

1063–8210/97$10.00 © 1997 IEEE

VLSI Array Algorithms and Architectures for RSA Modular Multiplication

Yong-Jin Jeong, *Member, IEEE,* and Wayne P. Burleson, *Member, IEEE*

***Abstract*—We present two novel iterative algorithms and their array structures for integer modular multiplication. The algorithms are designed for Rivest–Shamir–Adelman (RSA) cryptography and are based on the familiar iterative *Horner's* rule, but use precalculated complements of the modulus. The problem of deciding which multiples of the modulus to subtract in intermediate iteration stages has been simplified using simple look-up of precalculated complement numbers, thus allowing a finer-grain pipeline. Both algorithms use a carry save adder scheme with modulo reduction performed on each intermediate partial product which results in an output in carry-save format. Regularity and local connections make both algorithms suitable for high-performance array implementation in FPGA's or deep submicron VLSI. The processing nodes consist of just one or two full adders and a simple multiplexor. The stored complement numbers need to be precalculated only when the modulus is changed, thus not affecting the performance of the main computation. In both cases, there exists a bit-level systolic schedule, which means the array can be fully pipelined for high performance and can also easily be mapped to linear arrays for various space/time tradeoffs.**

***Index Terms*— Cryptography, modular multiplication, RSA, systolic arrays, VLSI.**

I. Introduction

Cryptography systems have been growing in importance recently as a method for improving data security. *Public key cryptography* (PKC) systems are generally preferred to traditional *secret key cryptography* systems like the *data encryption standard* due to the safety of key distribution [3]. The Rivest–Shamir–Adelman (RSA) [10] system is one of the most widely used public key cryptography systems, and its core arithmetic is modular multiplication over a positive integer. Modular multiplication is also a major computation of *residue number systems* [13] as well as other cryptography systems (e.g., *international data encryption algorithm* [8], [16], Diffie–Hellman key exchange [3]). In this paper, we develop an array modular multiplier with applications to, but not restricted to, RSA systems.

In RSA, the modulus is a product of two large prime numbers, usually more than 500 bits, and should be changeable for security reasons. But, since the modulus (or key) is not changed very often, we can use precomputation and look-up in our array modular multipliers. We are not aware of anyone who has utilized this special property of *multirate input data* in the RSA algorithm, that is, the *input message* changes rapidly while the *key* remains unchanged for a long period. In practice, the key is updated infrequently, for example, a few months, weeks, or days, depending on the security requirements. In order to satisfy the ever growing security requirements of high-speed communications, such as personal communication services and wireless local area networks, a *dedicated* VLSI hardware solution is needed because of 1) high throughput requirements, 2) low-power requirements, 3) a high-volume market, 4) the computation is poorly suited to microprocessors or DSP's, and 5) the problem size is expected to continue to grow rather than saturate.

Modular multiplication is generally considered a complicated arithmetic operation because of the inherent multiplication and division operations. There are two main approaches to computing modular multiplication: 1) perform the modulo operation *after* multiplication or 2) *during* multiplication. The modulo operation is accomplished by integer division in which only the remainder is needed for further computation. The first approach requires a $n \times n$ bit multiplier with a $2n$-bit register followed by a $2n \times n$ bit divider. In the second approach, the modulo operation occurs in each iteration step of integer multiplication. Therefore the first approach requires more hardware while the second requires more addition/subtraction computations due to $O(n)$ modulo reduction steps. In both cases, most previous research has focused on the fast calculation of a long carry chain. Redundant number systems and a higher radix carry-save form are some of the different number representations that have been used for this purpose [12], [14]. A carry prediction technique has also been used for fast calculation of modular multiplication [1].

Since PKC was introduced, many algorithms and hardware structures have been proposed for modular multiplication, and [4] contains a good review on this topic. Several array structures suited for VLSI implementation have been discussed in [4], [5], [14], and [15]. In [14], Vandemeulebroecke *et al.*, use a *modulo after multiplication* approach using a *signed digit* number representation. It consists of two arrays: one for multiplication and the other for integer division. In [5], Koc and Hung apply Blakley's algorithm [2] and use a sign-estimation method by looking at the five most significant bits in each iteration stage. Although they derive a bit-level systolic array structure, the latency and clock cycle are relatively long due to the control node which estimates the sign of the intermediate result in each stage. In [4] and [15], Eldridge and Walter use Montgomery's algorithm [9] which only works if

Manuscript received November 21, 1994; revised January 26, 1996. This work was supported in part by NSF Grant MIP-9108086.

Y. Jeong is with Samsung Electronics, Co., Seoul, Korea.

W. Burleson is with the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA 01003 USA.

Publisher Item Identifier S 1063-8210(97)01953-7.

1063–8210/97$10.00 © 1997 IEEE

the modulus is relatively prime to the radix, although this is always the case in RSA.

In this paper, we develop two new VLSI array architectures for modular multiplication. The idea is similar to Montgomery's algorithm in which he tries to make each partial product a multiple of the radix to simplify the multiplication by the radix (just by shifting) by only looking at the least significant bits (LSB), thus requiring a post-processing step to get the final answer. In our algorithms, we look at the most significant bits (MSB) to remove higher bit positions while keeping the correct answer in each partial product, keeping it within a certain range. Due to the simple translation of a modulo operation into an addition of a precalculated complement of the modulus, the *modulo during multiplication* approach is used with a carry-save adder structure. Instead we pay for multiplexors to choose the precalculated integer depending on the control which is generated in the leftmost node in each stage. Compared to previous works, we can obtain a higher clock frequency mainly due to the simplified modulo reduction operation. In Section II, we will explain our basic concept for the modulo reduction operation and then describe the two iterative algorithms. Array structures corresponding to these algorithms, analysis, and some modifications are also discussed in this section. Conclusions and discussion are in Section III.

II. Modular Multiplication Algorithm

In a modular multiplication, the n-bit modulus C is represented by a binary number system as $C = \sum_{i=0}^{n-1} c_i 2^i$ where $c_i \in GF(2)$. Obviously C is less than 2^n. We introduce K, which is called the *complement* of the modulus C, such that

$$K \equiv 2^n \bmod C. \quad (1)$$

In other words, any carry of weight 2^n can be replaced by an addition of K, which means that the end-around carry implies an extra addition. If K does not change frequently, we can precalculate multiples of K and store them in registers for use in the modulo reduction operation. Note that if the MSB of C is 1, K is equivalent to $-C$ in a 2's complement number system.

Now we describe the general modular multiplication algorithm using the *modulo during multiplication* approach. Given any two n-bit integers, A and B, and the n-bit modulus C, where $(C > A, B)$, the modular multiplication can be described by an iterative procedure using *Horner's rule*

$$\begin{aligned} AB \bmod C &= A \cdot \sum_{i=0}^{n-1} b_i 2^i \bmod C \\ &= ((\cdots(b_{n-1}A)2 + b_{n-2}A)2 \\ &\quad + \cdots + b_1 A)2 + b_0 A) \bmod C. \end{aligned} \quad (2)$$

We can describe (2) in a recursive form as follows:

$$\begin{aligned} P_0 &= 0 \\ P_i &= 2P_{i-1} + b_{n-1}A \bmod C \end{aligned} \quad (3)$$

and P_n is the final result. Using (1) and (3), we will derive two different bit-level array structures.

A. Using the CSA Scheme

The carry save addition (CSA) scheme is the most commonly used technique in integer multiplication to reduce the carry propagation penalty [6]. In the CSA scheme, a partial sum and a carry sequence are generated in the intermediate stages and the carry propagation occurs only at the last stage. The basic element of the CSA scheme is a full adder (FA) which is often called a (3, 2) counter. It accepts three inputs, referred to here as s_i, c_i, x_i with (associated weight 2^i), and produces two outputs, carry c_o (with weight 2^{i+1}) and sum s_o (with weight 2^i). The arithmetic operation of the (3, 2) counter can thus be described by the familiar expression:

$$2c_o + s_o = s_i + c_i + x_i \quad (4)$$

where "+" means an algebraic (not Boolean) addition.

Using the CSA scheme, we have a carry of weight 2^n in the leftmost node in each stage. As shown in (1), this carry can be replaced by the addition of the integer K for a modulo operation. The basic idea in our approach is that we handle the carries of weight 2^n and higher by using K wherever they appear, unless the basic CSA structure is broken. From (3), let us denote a partial product P_i as

$$P_i \equiv 2C_i + S_i \quad (0 \le i \le n) \quad (5)$$

then, the valid range of P_i is

$$0 \le P_i \le 3 \cdot 2^n - 3. \quad (6)$$

This means we allow P_i to be greater than modulus C at intermediate stages.

Before we begin the derivation of a recursive equation for modulo multiplication, we define a new variable K_h to handle multiple end-around carrys

$$K_h \stackrel{\text{def}}{=} h \cdot K \bmod C \quad (7)$$

where h is a positive integer $(1, 2, \cdots)$ and K is defined in (1). Then

$$2^{n+j} \bmod C = 2^j \cdot K \bmod C.$$

Carrys can also appear in a combined mode. As an example, suppose we have two carrys of weight 2^{n+1} and one carry of weight 2^n, then $(2^{n+1} + 2^{n+1} + 2^n) \bmod C = 5 \cdot 2^n \bmod C = 5 \cdot K \bmod C = K_5$.

Equation (3) contains two modulo reduction steps and can be written by introducing a new partial product term T_i, as

$$\begin{aligned} &\text{i)} \quad T_i = 2P_{i-1} \bmod C \\ &\text{ii)} \quad P_i = (T_i + b_{n-1}A) \bmod C. \end{aligned} \quad (8)$$

But step ii) cannot be implemented by the CSA scheme because it has four operands to be added. (Note that the modulo operation implies at least one extra addition of K.) This can be solved by dividing step ii) into two steps as

$$\begin{aligned} &\text{i)} \quad T_i = 2P_{i-1} \bmod C \\ &\text{ii-a)} \quad T_i^* = T_i + b_{n-1}A \\ &\text{ii-b)} \quad P_i = T_i^* \bmod C. \end{aligned} \quad (9)$$

In step i), $2P_{i-1}$ implies one 2^{n+1} term (c_{i-1}^{n-1}) and two 2^n terms (s_{i-1}^{n-1} and c_{i-1}^{n-2}), which can generate a maximum of $4 \cdot 2^n$.[1] In step ii-a), we do not perform the modulo operation because there are already three operands: two from T_i in carry save form, and one for A depending on b_{n-i}. Instead we want to pass through the MSB carry of T_i to step ii-b). So, in step ii-b), we will have at most $2 \cdot 2^n$ (one passed from T_i and another newly generated in T_i^*) as end-around carrys. In both the steps i) and ii-b), only one additional operand is allowed. That is why we precalculate the K_h's instead of adding K multiple times.

To explain the algorithm more formally, we define $\sigma(P_i)$ as follows:

$$\sigma(P_i) \stackrel{\text{def}}{=} P_i - h \cdot 2^n + K_h$$

where

$$h = f(x_1, x_2, x_3, \cdots, x_r) \tag{10}$$

and the function $f(\cdot)$ calculates the total magnitude of end-around carrys, and $x_1, x_2, \cdots, x_r$ are bit variables (always carrys and sums of the MSB position) which contribute to the translation of (1). Thus

$$f(x_1, x_2, \cdots, x_r) = \sum_{k=1}^{r} \alpha_k x_k \tag{11}$$

where $\alpha_k = 1$ if x_k has weight $2^n, \alpha_k = 2$ if the weight is $2^{n+1}, \alpha_k = 4$ if the weight is 2^{n+2}, and so on. In other words, $\sigma(P_i)$ replaces $h \cdot 2^n$ with K_h which is precalculated.

Using (10), we can rewrite algorithm (9) as follows:

$$\begin{aligned} &\text{i)} && T_i = \sigma(2P_{i-1}) \\ &\text{ii-a)} && T_i^* = T_i + b_{n-1} A \\ &\text{ii-b)} && P_i = \sigma(T_i^*). \end{aligned} \tag{12}$$

As we can see in Fig. 1, the function $f(\cdot)$ of the above algorithm is

$$\begin{aligned} \text{for step i)} \quad & f(\cdot) = 2c_{i-1}^{n-1} + s_{i-1}^{n-1} + c_{i-1}^{n-2} \\ \text{for step ii-b)} \quad & f(\cdot) = \gamma_i^{n-1} + \gamma_i^{*\,n-1} \end{aligned}$$

where $\gamma_i^{n-1}, \gamma_i^{*\,n-1}$ are the MSB carrys of T_i and T_i^*, respectively (both have the weight 2^n).

Now we will informally verify that the algorithm (12) satisfies the valid range of (6) for all P_i's ($i = 0, 1, \cdots, n$). Obviously $0 \le P_0 < 3 \cdot 2^n - 3$. Suppose $0 \le P_{i-1} < 3 \cdot 2^n - 3$, then

$$\begin{aligned} 0 &\le 2P_{i-1} \\ &< 6 \cdot 2^n - 6 \\ 0 &\le \sigma(2P_{i-1}) \\ &< 6 \cdot 2^n - 6 - 4 \cdot 2^n + 2^n \\ &= 3 \cdot 2^n - 6 \\ 0 &\le T_i^* \\ &< 4 \cdot 2^n - 6 \end{aligned}$$

[1] Subscript is for an iteration stage and superscript is for denoting bit positions, that is, $S_i = \sum_{j=0}^{n-1} s_i^j \cdot 2^j$. Also note that lower case letters are used for bit-level variables while upper case is for word-level variables.

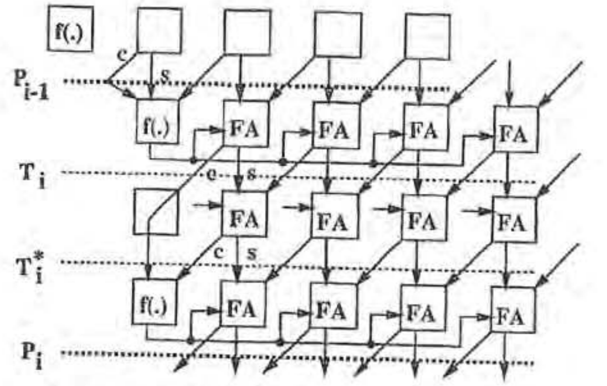


Fig. 1. An iteration stage of modular multiplication using the CSA scheme.

$$\begin{aligned} 0 &\le \sigma(T_i^*) \\ &< 4 \cdot 2^n - 6 - 2 \cdot 2^n + 2^n \\ &< 3 \cdot 2^n - 3 \end{aligned}$$

which assures $0 \le P_i < 3 \cdot 2^n - 3$. Therefore the algorithm (12) produces a final output P_n which is less than $3 \cdot 2^n - 3$. It can be directly fed into the next multiplication stage for further iteration if necessary (e.g., exponentiation).

Fig. 2(a) shows a single stage of the dependence graph (DG) which can be directly implemented as a parallel array multiplier. Fig. 2(b) describes the node functions. The nodes X_1, X_3 are control nodes which calculate the control value h of (10), and hence need simple encoders. The node X_2 is just a wire. Node type A is a FA with a 4-1 multiplexor and an AND gate. Node type B is just a FA with an AND gate and node type C is a FA with a 2-1 multiplexor and an AND gate. Note that node type B does not need a multiplexor and type C needs only K_1 and K_2 because the max value of h is two in node X_3. An AND gate is needed in type A and C to accept $K_0 = 0$ when the control value h is zero. There exists a systolic schedule which is not linear due to its skewed connection between the stages. Table I shows an example for our new bit-level modular multiplication algorithm using $n = 12$, with $A = 010001000100 (= 1092)$, $B = 010011001101 (= 1229)$, and $C = 100000101001 (= 2089)$. The K_h's are precalculated as $K_1 = K = 011111010111 (= 2007)$, $K_2 = 011110000101 (= 1925)$, $K_3 = 011100110011 (= 1843)$, $K_4 = 011011100001 (= 1761)$. The final output is

$$\begin{aligned} P_n &= 2(001100010101) + (110111001010) \\ &= 1001111110100 \\ &(= 5108) \end{aligned}$$

which equals 930 after modulo reduction to 2089.

By merging two nodes into one in each row as has been done in [5], one can modify the DG in Fig. 2 to derive a simpler DG. This is shown in Fig. 3. Node types AA, BB, CC are newly merged nodes which have two A, B, C type nodes, respectively. It now allows a linear systolic schedule and shows a better overview of the hardware array implementation. If the wordlength n is an even number, then all nodes except the control nodes will be merged nodes. Here we have the original nodes A, B, C in the LSB place because n is odd. From

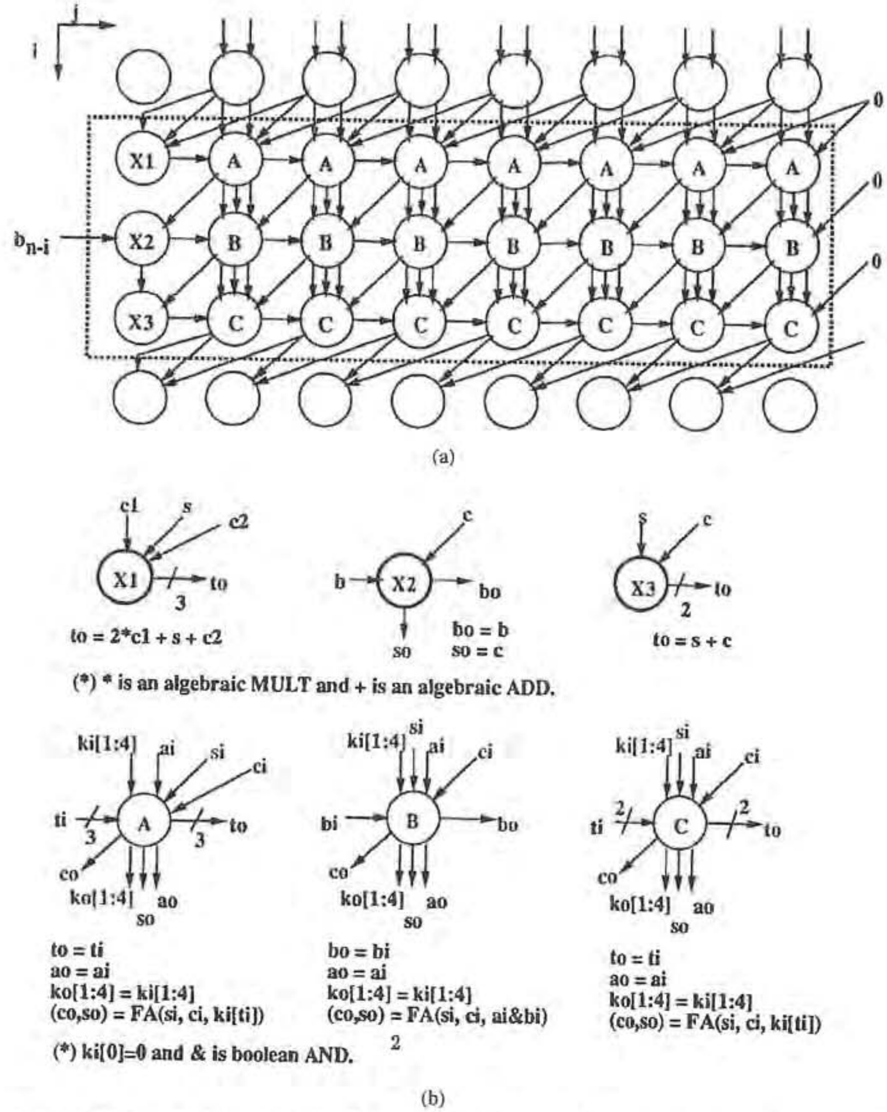


Fig. 2. Array structure for modular multiplication using CSA scheme ($n = 7$): (a) ith stage of DG for modular multiplication ($n = 7$) and (b) PE node description.

Figs. 2 or 3, we can obtain many different one-dimensional arrays (e.g., bit-serial modulo multiplier) depending on the mapping functions [7].

In our array, the control is generated in a single left-most node which has just four gates (two XOR, one AND, one NOR gate) or two gates (one XOR and one NOR gate). The simplicity of the control nodes gives a much faster clock cycle for the entire array. Thus, it is not the control node but the processing node which determines the clock cycle. Note that all signals in Fig. 2 except the carry (c_o) and sum (s_o) are *transmittent* signals, which means they are not modified while passing through the array, thus allowing for broadcasting.

Compared to [5], the dependency structure looks the same except for the control nodes due to the basic CSA scheme. However, the main difference is in the function of the control nodes. In [5], the control node (denoted as X^5, LU^5, LY^5) is made by merging five or six identical processing nodes and each one (X, L, Y, U) is fairly complex (roughly three XOR's and more than five AND or OR gates). Therefore, the resulting control nodes become five or six times larger, and this is the critical reason for the slow clock cycle time. It is also worthwhile to compare with [4] which claims to be the fastest structure. Our array gives a faster clock cycle (two XOR with one MUX versus five XOR) although it has a longer latency. There is also no restriction on choosing the modulus, thus allowing more general application. Furthermore, we do not need a post processing step because we keep the correct value in all iteration stages. Both methods require a precalculation.

B. Using a Modified CSA Scheme

Now we derive a far simpler array structure from the algorithm (3) by slightly modifying the basic element of the CSA scheme. To directly apply the CSA structure to the modular multiplication algorithm, we have to modify the basic element so that it can accept an additional operand which

TABLE I
EXAMPLE SHOWING COMPLETE FUNCTION OF FIG. 2 ($N = 12$)

stage		(carry,sum)	f(.)	b_{n-i}
0	P_0	00 00 00 00 00 00 00 00 00 00 00 00	-	-
1	T_1	00 00 00 00 00 00 00 00 00 00 00 00	f(.)=0	
	T_1^*	00 00 00 00 00 00 00 00 00 00 00 00		0
	P_1	00 00 00 00 00 00 00 00 00 00 00 00	f(.)=0	
2	T_2	00 00 00 00 00 00 00 00 00 00 00 00	f(.)=0	
	T_2^*	00 01 00 00 01 01 00 00 01 01 00 01		1
	P_2	00 01 00 00 01 01 00 00 01 01 00 01	f(.)=0	
3	T_3	01 00 00 01 01 00 00 01 01 00 01 00	f(.)=0	
	T_3^*	01 00 00 01 01 00 00 01 01 00 01 00		0
	P_3	01 00 00 01 01 00 00 01 01 00 01 00	f(.)=0	
4	T_4	00 01 10 10 01 01 01 10 00 10 01 01	f(.)=1	
	T_4^*	00 10 01 00 01 01 10 00 01 00 01 01		0
	P_4	01 00 01 00 01 10 00 00 01 00 01 01	f(.)=0	
5	T_5	00 10 01 11 01 01 00 10 00 10 10 01	f(.)=1	
	T_5^*	01 00 10 01 01 01 01 00 01 01 00 01		1
	P_5	01 01 00 01 01 01 01 00 01 01 00 01	f(.)=0	
6	T_6	01 01 10 10 10 10 00 10 01 01 10 01	f(.)=1	
	T_6^*	01 11 01 01 10 01 01 00 10 11 00 10		1
	P_6	10 01 01 10 00 01 01 01 01 01 01 00	f(.)=0	
7	T_7	01 11 01 01 10 01 01 01 01 10 00 01	f(.)=2	
	T_7^*	10 01 01 10 00 01 01 01 10 00 00 01		0
	P_7	00 10 11 01 01 10 01 11 00 01 01 10	f(.)=1	
8	T_8	01 10 10 11 01 11 01 01 01 11 01 01	f(.)=1	
	T_8^*	10 01 01 01 10 01 01 01 10 01 01 01		0
	P_8	00 10 10 11 01 10 01 11 00 10 10 10	f(.)=1	
9	T_9	01 10 10 11 01 11 01 10 01 10 01 01	f(.)=1	
	T_9^*	10 01 01 01 10 01 10 00 10 00 01 01		1
	P_9	00 10 10 11 01 11 00 10 00 01 10 10	f(.)=1	
10	T_{10}	01 10 10 11 10 10 00 01 10 10 01 01	f(.)=1	
	T_{10}^*	10 10 01 10 10 01 00 10 10 01 01 10		1
	P_{10}	01 01 11 10 01 10 01 10 00 10 11 01	f(.)=1	
11	T_{11}	10 11 01 11 01 11 00 10 01 10 10 01	f(.)=1	
	T_{11}^*	01 01 10 01 10 01 01 00 10 01 00 01		0
	P_{11}	01 11 01 11 01 10 01 10 00 10 01 10	f(.)=1	
12	T_{12}	01 11 10 11 01 10 00 01 00 11 00 01	f(.)=2	
	T_{12}^*	10 10 01 01 10 00 00 01 01 01 00 01		1
	P_{12}	01 01 10 11 01 01 00 10 01 10 01 10	f(.)=1	

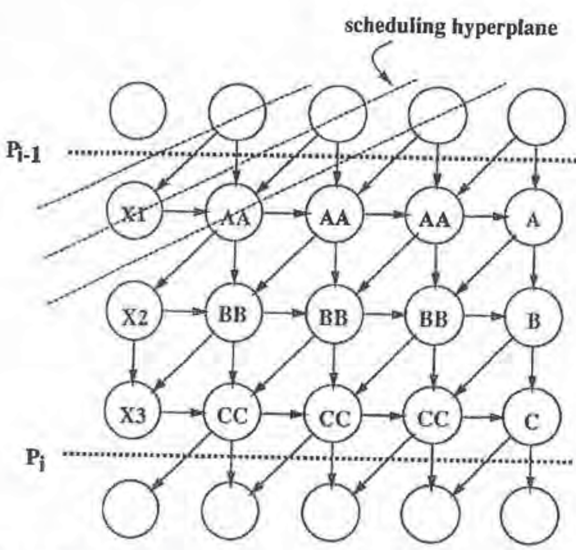


Fig. 3. Regularized DG of Fig. 2 by node merging.

arises from end-around carry terms in the MSB nodes. Now we extend the (3, 2) counter to generate a more general adder, which we call a *partially generalized counter* (PGC), as shown

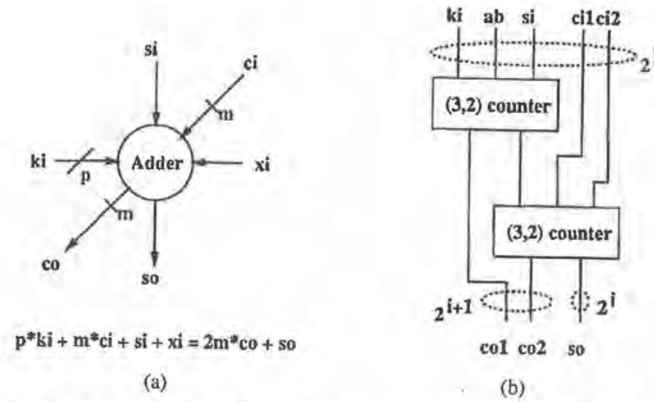


Fig. 4. A partially generalized counter (PGC) and a new adder logic: (a) a PGC element and (b) an adder derived from (a) with $m = 2$, $p = 1$.

in Fig. 4. In Fig. 4(a), s_i, c_i are a partial sum and a carry of the previous stage. Note that the carry signal is not a single bit, but m-bits. The variable x_i is a regular operand which appears in a typical array integer multiplication as $a_i \& b_j$(& is logical AND). An extra operand k_i is introduced with a p-bit signal for handling the end-around carry terms. So, the arithmetic operation of PGC can be described by

$$2m \cdot c_o + s_o = s_i + m \cdot c_i + x_i + p \cdot k_i \tag{13}$$

where "·" and "+" are algebraic multiplication and addition. If $m = 1$, the only possible value of p is zero, which makes it a typical FA. Therefore, we see that the m should be at least two to accept an additional operand resulting from end-around carrys. Furthermore, if $m = 2$ the only possible value of p is one, which results in five inputs and three outputs. This can be easily implemented using two FA's as shown in Fig. 4(b). We can also derive different adders by choosing different p and m. However, we want to minimize the number of signal lines to reduce the complexity of the node function.

To apply the newly derived adder in the CSA structure, we need a different CSA form (two carry terms) to express P_i. Let us denote

$$P_i \equiv 2(C_{i1} + C_{i2}) + S_i, \quad (0 \leq i \leq n) \tag{14}$$

then, the valid range of P_i is $0 \leq P_i \leq 5 \cdot 2^n - 5$. Following the same procedure as the CSA scheme, let us say, $C_{i1} = \sum_{j=0}^{n-1} c_{i1}^j 2^j$, $C_{i2} = \sum_{j=0}^{n-1} c_{i2}^j 2^j$, $S_i = \sum_{j=0}^{n-1} s_i^j 2^j$. Then, the control generating function $f(\cdot)$ is

$$f(\cdot) = 2(c_{(i-1)1}^{n-1} + c_{(i-1)2}^{n-1}) + (s_{i-1}^{n-1} + C_{(i-1)1}^{n-2} + c_{(i-1)2}^{n-2}). \tag{15}$$

Note that we need seven complement integers, $K_1, K_2, \cdots, K_7$. Fig. 5 shows an array structure for the modified CSA scheme with node functions. The node type D is a serial connection of two full adders with an 8-1 multiplexor. For the example shown in the previous section, additional K_h's should be precalculated as $K_5 = 011011100001 (= 1679)$, $K_6 = 011011100001 (= 1597)$, $K_7 = 011011100001 (= 1515)$.

The range of the final output P_n can be further reduced using two more basic CSA stages as shown in Fig. 6. The first extra CSA stage adds up to sequences of carrys and a

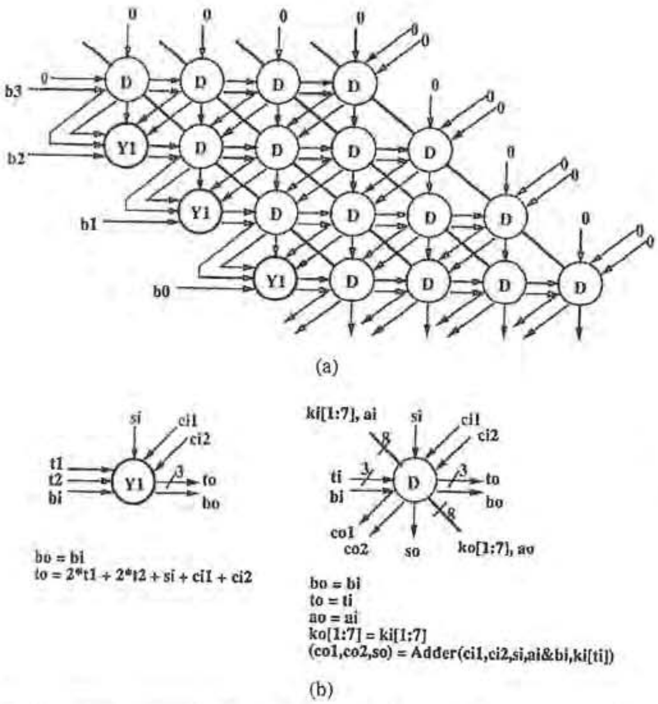


Fig. 5. Array structure for modular multiplication using modified CSA scheme: (a) DG for modular multiplication using CSA scheme ($n = 4$) and (b) PE descriptions.

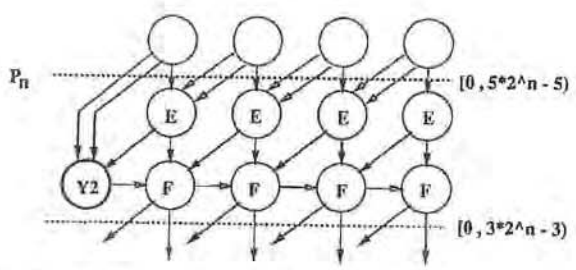


Fig. 6. Extra stages to reduce the range from $5 \cdot 2^n - 5$ to $3 \cdot 2^n - 3$.

sum of P_n except for the MSB carrys. The MSB carrys of P_n are compensated for in the second extra CSA stage. The range $5 \cdot 2^n - 5$ of P_n is reduced to $3 \cdot 2^n - 3$ by replacing maximum $3 \cdot 2^n$ with $K_h (K_h < 2^n)$. At the second CSA stage of Fig. 6, a 4-1 multiplexor is needed instead of an 8-1 multiplexor.

By running SIS [11], a multilevel logic minimization tool, with arbitrary encoding, the control note Y_1 had eight OR gates (including one, three-input OR), nine AND gates (including two, three-input AND), and three inverters, and the combinational logic depth was 12. This is reasonably close to the function of node D (2-bit FA with a multiplexor)[2] in both area and speed. Here again note that all signals are transmittent except carrys and a sum.

III. Conclusion and Discussion

We have shown two new array structures for modular multiplication. Both use the basic CSA structure with some modification and do not need any number translation. The complement and its multiples need be updated only when the modulus changes. In RSA applications, the key does not change very often therefore this preprocessing step is acceptable. The details of each processing node in the two array structures are summarized in Table II. The first array needs processing nodes of $n(X_1 + X_3) + n^2(A + B + C)$, and the second needs $(n-1)Y_1 + Y_2 + n^2 D + n(E + F)$. In each iteration stage, the second array has a smaller number of FA's but the first one has simpler encoders and a multiplexor which lead to a faster clock cycle. Both have a systolic schedule which means they can be fully bit-wise systolized for maximum throughput.

TABLE II
Summary for Logic of PE Nodes in Two Array Structures

array 1 (Fig.2)				
X1	X3	A	B	C
(5,3)encoder	(3,2)encoder	(4:1) mux FA AND gate	FA AND gate	(2:1) mux FA AND gate
array 2 (Fig.5 and Fig.6)				
Y1	Y2	D	E	F
(8,3) encoder	(4,2) encoder	(8:1) mux 2 FAs AND gate	FA	(4:1) mux FA

Our approach has advantages over other previous array structures. First, it is more general and has no restrictions in choosing the modulus, hence can be used in any application in which a larger number of computations are required for a relatively long-lasting modulus (RSA, key-exchange, special purpose DSP chip). Second, the algorithm and architecture is simple to understand and verify, hence is easy to modify for a hardware implementation, and does not use any special technique like sign estimation which may imply a significant degree of hardware and verification complexity. We could obtain a faster clock cycle by reducing the complexity of the control nodes in both area and speed. From an algorithmic point of view, the concerns put forward in most previous papers on deciding multiples of the modulus have been eliminated by multiplexing precalculated complement numbers. We have verified our array structures first by C programming and then by implementing two prototype VLSI designs which were verified for function and timing using logic and circuit simulation.

Finally, the RSA algorithm requires more than 500 bits for security reasons, which may make the array multiplier too large (~15 million gates including pipeline stages). To maintain high performance for larger bit-lengths, we need to map the original array onto smaller processor arrays in order to build large "virtual" modular multipliers on fixed sized arrays, called *partitioning* [7]. The regularity of our algorithm makes it easy to find an appropriate partitioning strategy. Because the control nodes which generate the multiplexor control signals are located in the MSB in each iteration, the LPGS (*locally parallel, globally sequential* [7]) scheme is an appropriate choice, needing some extra buffers outside the processor array to contain intermediate data for each block.

[2] We used the common mapping to two-input OR gates to measure the logic depth, hence the 2-bit FA has logic depth 9.

ACKNOWLEDGMENT

The authors would like to acknowledge the work of the referees whose reviews and comments were very meticulous and insightful.

REFERENCES

[1] E. B. Brickell, "A fast modular multiplication algorithm with application to two key cryptography," in *Advances in Cryptology, Proceedings of Crypto 82.* New York: Plenum, 1982, pp. 51–60.

[2] G. R. Blakley, "A computer alrogithm for the product AB modulo M," *IEEE Trans. Comput.*, vol. 32, pp. 497–500, 1983.

[3] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Inform. Theory*, vol. IT-22, pp. 644–654, Nov. 1976.

[4] S. E. Eldridge and D. Walter, "Hardware implementation of Montgomery's modular multiplication algorithm," *IEEE Trans. Comput.*, vol. 42, pp. 693–699, June 1993.

[5] C. K. Koc and C. Y. Hung, "Bi-level systolic arrays for modular multiplication," *J. VLSI Sig. Proc.*, vol. 3, pp. 215–223, 1991.

[6] I. Koren, *Computer Arithmetic Algorithms.* Englewood Cliffs, NJ: Prentice-Hall, 1993.

[7] S. Y. Kung, *VLSI Array Processors.* Englewood Cliffs, NJ: Prentice-Hall, 1988.

[8] X. Lai and J. L. Massey, "A proposal for a new block encryption standard," in *EUROCRYPT '90*, Aarhus, Denmark, May 1990.

[9] P. L. Montgomery, "Modular multiplication without trial division," *Math. Comp.*, vol. 44, pp. 519–521, 1985.

[10] R. L. Rivest, A. Shamir, and L. Adelman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, pp. 120–126, Feb. 1978.

[11] E. M. Sentovich, K. J. Singh, C. Moon, H. Savoy, and R. K. Brayton, "Sequential circuit design using synthesis and optimization," in *Proc. ICCD*, 1992.

[12] N. Takagi, "A radix-4 modular multiplication hardware algorithm for modular exponentiation," *IEEE Trans. Comput.*, vol. 41, pp. 949–956, Aug. 1992.

[13] F. J. Taylor, "Residue arithmetic: A tutorial with examples," *IEEE Comput. Mag.*, vol. 17, pp. 50–62, May 1984.

[14] A. Vandemeulebroecke *et al.*, "A new carry-free division algorithm and its application to a single chip 1024-b RSA processor," *IEEE J. Solid-State Circuits*, vol. 25, pp. 748–755, June 1990.

[15] C. D. Walter, "Systolic modular multiplication," *IEEE Trans. Comput.*, vol. 42, npp. 376–378, Mar. 1993.

[16] R. Zimmermann *et al.*, "A 177 Mb/s VLSI implementation of International data encryption algorithm," *IEEE J. Solid-State Circuits*, vol. 29, pp. 303–307, Mar. 1994.

Yong-Jin Jeong (M'96) received the B.S. degree in control and instrumentation engineering from the Seoul National University, Seoul, Korea, in 1983, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Massachusetts, Amherst, MA, in 1995.

He is now working for Samsung Electronic Co., Seoul, Korea, in the area of communication chipset design. From 1983 to 1989, he was a Member of the Research Staff at the Electronics and Telecommunications Research Institute (ETRI) in Daejun, Korea. He has also worked for the Swedish telecommunication company, LM Ericsson, Stockholm, Sweden, as a Training and Design Engineer. His research interests include computer arithmetic, parallel algorithm, and array synthesis, VLSI testing, logic synthesis, and wireless communication systems.

Wayne P. Burleson (M'89) received the B.S.E.E. and M.S.E.E. degrees from the Massachusetts Institute of Technology (M.I.T.), Cambridge, in 1983 and the Ph.D. degree in VLSI signal processing from the University of Colorado, Boulder, in 1989.

He has been exploring the area of VLSI signal processing for 13 years. His work has included research, development, teaching, and industrial work at a variety of levels, including development of algorithms, architectures, circuits, and computer-aided design (CAD) tools. He is currently an Assistant Professor of Electrical and Computer Engineering at the University of Massachusetts, Amherst. For four years, he worked as a custom DSP chip designer for VLSI Technology Inc., and Fairchild Semiconductor. He is currently being funded by the NSF, exploring advanced timing schemes for CMOS systems including circuit and system design, clocking methods, and verification techniques which are all being developed to meet the needs of modern highly pipelined architectures with tight latency requirements. He is also currently developing a CAD system for VLSI arrays with applications in digital signal processing (DSP) and communications. The ARRay ESTimator (ARREST) system allows a broad range of algorithm and architecture exploration, interface with simulation and estimation tools, and finally output to VERILOG for backend synthesis. In a more applied area, he is currently collaborating with the University of Massachusetts' researchers in real-time systems, computer vision and robotics, and the development and implementation of special-purpose coprocessors in advanced technologies.

Dr. Burleson is a member of the ACM and Sigma Xi.

Attachment 6B

VLSI Array Algorithms and Architectures for RSA Modular Multiplication

Yong-Jin Jeong, *Member, IEEE,* and Wayne P. Burleson, *Member, IEEE*

***Abstract*—We present two novel iterative algorithms and their array structures for integer modular multiplication. The algorithms are designed for Rivest–Shamir–Adelman (RSA) cryptography and are based on the familiar iterative *Horner's* rule, but use precalculated complements of the modulus. The problem of deciding which multiples of the modulus to subtract in intermediate iteration stages has been simplified using simple look-up of precalculated complement numbers, thus allowing a finer-grain pipeline. Both algorithms use a carry save adder scheme with modulo reduction performed on each intermediate partial product which results in an output in carry-save format. Regularity and local connections make both algorithms suitable for high-performance array implementation in FPGA's or deep submicron VLSI. The processing nodes consist of just one or two full adders and a simple multiplexor. The stored complement numbers need to be precalculated only when the modulus is changed, thus not affecting the performance of the main computation. In both cases, there exists a bit-level systolic schedule, which means the array can be fully pipelined for high performance and can also easily be mapped to linear arrays for various space/time tradeoffs.**

***Index Terms*— Cryptography, modular multiplication, RSA, systolic arrays, VLSI.**

I. Introduction

CRYPTOGRAPHY systems have been growing in importance recently as a method for improving data security. *Public key cryptography* (PKC) systems are generally preferred to traditional *secret key cryptography* systems like the *data encryption standard* due to the safety of key distribution [3]. The Rivest–Shamir–Adelman (RSA) [10] system is one of the most widely used public key cryptography systems, and its core arithmetic is modular multiplication over a positive integer. Modular multiplication is also a major computation of *residue number systems* [13] as well as other cryptography systems (e.g., *international data encryption algorithm* [8], [16], Diffie–Hellman key exchange [3]). In this paper, we develop an array modular multiplier with applications to, but not restricted to, RSA systems.

In RSA, the modulus is a product of two large prime numbers, usually more than 500 bits, and should be changeable for security reasons. But, since the modulus (or key) is not changed very often, we can use precomputation and look-up in our array modular multipliers. We are not aware of anyone who has utilized this special property of *multirate input data* in the RSA algorithm, that is, the *input message* changes rapidly while the *key* remains unchanged for a long period. In practice, the key is updated infrequently, for example, a few months, weeks, or days, depending on the security requirements. In order to satisfy the ever growing security requirements of high-speed communications, such as personal communication services and wireless local area networks, a *dedicated* VLSI hardware solution is needed because of 1) high throughput requirements, 2) low-power requirements, 3) a high-volume market, 4) the computation is poorly suited to microprocessors or DSP's, and 5) the problem size is expected to continue to grow rather than saturate.

Modular multiplication is generally considered a complicated arithmetic operation because of the inherent multiplication and division operations. There are two main approaches to computing modular multiplication: 1) perform the modulo operation *after* multiplication or 2) *during* multiplication. The modulo operation is accomplished by integer division in which only the remainder is needed for further computation. The first approach requires a $n \times n$ bit multiplier with a $2n$-bit register followed by a $2n \times n$ bit divider. In the second approach, the modulo operation occurs in each iteration step of integer multiplication. Therefore the first approach requires more hardware while the second requires more addition/subtraction computations due to $O(n)$ modulo reduction steps. In both cases, most previous research has focused on the fast calculation of a long carry chain. Redundant number systems and a higher radix carry-save form are some of the different number representations that have been used for this purpose [12], [14]. A carry prediction technique has also been used for fast calculation of modular multiplication [1].

Since PKC was introduced, many algorithms and hardware structures have been proposed for modular multiplication, and [4] contains a good review on this topic. Several array structures suited for VLSI implementation have been discussed in [4], [5], [14], and [15]. In [14], Vandemeulebroccke *et al.*, use a *modulo after multiplication* approach using a *signed digit* number representation. It consists of two arrays: one for multiplication and the other for integer division. In [5], Koc and Hung apply Blakley's algorithm [2] and use a sign-estimation method by looking at the five most significant bits in each iteration stage. Although they derive a bit-level systolic array structure, the latency and clock cycle are relatively long due to the control node which estimates the sign of the intermediate result in each stage. In [4] and [15], Eldridge and Walter use Montgomery's algorithm [9] which only works if

Manuscript received November 21, 1994; revised January 26, 1996. This work was supported in part by NSF Grant MIP-9108086.

Y. Jeong is with Samsung Electronics, Co., Seoul, Korea.

W. Burleson is with the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA 01003 USA.

Publisher Item Identifier S 1063-8210(97)01953-7.

the modulus is relatively prime to the radix, although this is always the case in RSA.

In this paper, we develop two new VLSI array architectures for modular multiplication. The idea is similar to Montgomery's algorithm in which he tries to make each partial product a multiple of the radix to simplify the multiplication by the radix (just by shifting) by only looking at the least significant bits (LSB), thus requiring a post-processing step to get the final answer. In our algorithms, we look at the most significant bits (MSB) to remove higher bit positions while keeping the correct answer in each partial product, keeping it within a certain range. Due to the simple translation of a modulo operation into an addition of a precalculated complement of the modulus, the *modulo during multiplication* approach is used with a carry-save adder structure. Instead we pay for multiplexors to choose the precalculated integer depending on the control which is generated in the leftmost node in each stage. Compared to previous works, we can obtain a higher clock frequency mainly due to the simplified modulo reduction operation. In Section II, we will explain our basic concept for the modulo reduction operation and then describe the two iterative algorithms. Array structures corresponding to these algorithms, analysis, and some modifications are also discussed in this section. Conclusions and discussion are in Section III.

II. Modular Multiplication Algorithm

In a modular multiplication, the n-bit modulus C is represented by a binary number system as $C = \sum_{i=0}^{n-1} c_i 2^i$ where $c_i \in GF(2)$. Obviously C is less than 2^n. We introduce K, which is called the *complement* of the modulus C, such that

$$K \equiv 2^n \bmod C. \tag{1}$$

In other words, any carry of weight 2^n can be replaced by an addition of K, which means that the end-around carry implies an extra addition. If K does not change frequently, we can precalculate multiples of K and store them in registers for use in the modulo reduction operation. Note that if the MSB of C is 1, K is equivalent to $-C$ in a 2's complement number system.

Now we describe the general modular multiplication algorithm using the *modulo during multiplication* approach. Given any two n-bit integers, A and B, and the n-bit modulus C, where $(C > A, B)$, the modular multiplication can be described by an iterative procedure using *Horner's rule*

$$\begin{aligned} AB \bmod C &= A \cdot \sum_{i=0}^{n-1} b_i 2^i \bmod C \\ &= ((\cdots(b_{n-1}A)2 + b_{n-2}A)2 \\ &\quad + \cdots + b_1 A)2 + b_0 A) \bmod C. \end{aligned} \tag{2}$$

We can describe (2) in a recursive form as follows:

$$\begin{aligned} P_0 &= 0 \\ P_i &= 2P_{i-1} + b_{n-1}A \bmod C \end{aligned} \tag{3}$$

and P_n is the final result. Using (1) and (3), we will derive two different bit-level array structures.

A. Using the CSA Scheme

The carry save addition (CSA) scheme is the most commonly used technique in integer multiplication to reduce the carry propagation penalty [6]. In the CSA scheme, a partial sum and a carry sequence are generated in the intermediate stages and the carry propagation occurs only at the last stage. The basic element of the CSA scheme is a full adder (FA) which is often called a (3, 2) counter. It accepts three inputs, referred to here as s_i, c_i, x_i with (associated weight 2^i), and produces two outputs, carry c_o (with weight 2^{i+1}) and sum s_o (with weight 2^i). The arithmetic operation of the (3, 2) counter can thus be described by the familiar expression:

$$2c_o + s_o = s_i + c_i + x_i \tag{4}$$

where "+" means an algebraic (not Boolean) addition.

Using the CSA scheme, we have a carry of weight 2^n in the leftmost node in each stage. As shown in (1), this carry can be replaced by the addition of the integer K for a modulo operation. The basic idea in our approach is that we handle the carries of weight 2^n and higher by using K wherever they appear, unless the basic CSA structure is broken. From (3), let us denote a partial product P_i as

$$P_i \equiv 2C_i + S_i \quad (0 \le i \le n) \tag{5}$$

then, the valid range of P_i is

$$0 \le P_i \le 3 \cdot 2^n - 3. \tag{6}$$

This means we allow P_i to be greater than modulus C at intermediate stages.

Before we begin the derivation of a recursive equation for modulo multiplication, we define a new variable K_h to handle multiple end-around carrys

$$K_h \stackrel{\text{def}}{=} h \cdot K \bmod C \tag{7}$$

where h is a positive integer $(1, 2, \cdots)$ and K is defined in (1). Then

$$2^{n+j} \bmod C = 2^j \cdot K \bmod C.$$

Carrys can also appear in a combined mode. As an example, suppose we have two carrys of weight 2^{n+1} and one carry of weight 2^n, then $(2^{n+1} + 2^{n+1} + 2^n) \bmod C = 5 \cdot 2^n \bmod C = 5 \cdot K \bmod C = K_5$.

Equation (3) contains two modulo reduction steps and can be written by introducing a new partial product term T_i, as

$$\begin{aligned} &\text{i)} \quad T_i = 2P_{i-1} \bmod C \\ &\text{ii)} \quad P_i = (T_i + b_{n-1}A) \bmod C. \end{aligned} \tag{8}$$

But step ii) cannot be implemented by the CSA scheme because it has four operands to be added. (Note that the modulo operation implies at least one extra addition of K.) This can be solved by dividing step ii) into two steps as

$$\begin{aligned} &\text{i)} \quad T_i = 2P_{i-1} \bmod C \\ &\text{ii-a)} \quad T_i^* = T_i + b_{n-1}A \\ &\text{ii-b)} \quad P_i = T_i^* \bmod C. \end{aligned} \tag{9}$$

In step i), $2P_{i-1}$ implies one 2^{n+1} term (c_{i-1}^{n-1}) and two 2^n terms (s_{i-1}^{n-1} and c_{i-1}^{n-2}), which can generate a maximum of $4 \cdot 2^n$.[1] In step ii-a), we do not perform the modulo operation because there are already three operands: two from T_i in carry save form, and one for A depending on b_{n-i}. Instead we want to pass through the MSB carry of T_i to step ii-b). So, in step ii-b), we will have at most $2 \cdot 2^n$ (one passed from T_i and another newly generated in T_i^*) as end-around carrys. In both the steps i) and ii-b), only one additional operand is allowed. That is why we precalculate the K_h's instead of adding K multiple times.

To explain the algorithm more formally, we define $\sigma(P_i)$ as follows:

$$\sigma(P_i) \stackrel{\text{def}}{=} P_i - h \cdot 2^n + K_h$$

where

$$h = f(x_1, x_2, x_3, \cdots, x_r) \tag{10}$$

and the function $f(\cdot)$ calculates the total magnitude of end-around carrys, and $x_1, x_2, \cdots, x_r$ are bit variables (always carrys and sums of the MSB position) which contribute to the translation of (1). Thus

$$f(x_1, x_2, \cdots, x_r) = \sum_{k=1}^{r} \alpha_k x_k \tag{11}$$

where $\alpha_k = 1$ if x_k has weight $2^n, \alpha_k = 2$ if the weight is $2^{n+1}, \alpha_k = 4$ if the weight is 2^{n+2}, and so on. In other words, $\sigma(P_i)$ replaces $h \cdot 2^n$ with K_h which is precalculated.

Using (10), we can rewrite algorithm (9) as follows:

$$\begin{array}{ll} \text{i)} & T_i = \sigma(2P_{i-1}) \\ \text{ii-a)} & T_i^* = T_i + b_{n-1} A \\ \text{ii-b)} & P_i = \sigma(T_i^*). \end{array} \tag{12}$$

As we can see in Fig. 1, the function $f(\cdot)$ of the above algorithm is

$$\begin{array}{ll} \text{for step i)} & f(\cdot) = 2c_{i-1}^{n-1} + s_{i-1}^{n-1} + c_{i-1}^{n-2} \\ \text{for step ii-b)} & f(\cdot) = \gamma_i^{n-1} + \gamma_i^{*\,n-1} \end{array}$$

where $\gamma_i^{n-1}, \gamma_i^{*\,n-1}$ are the MSB carrys of T_i and T_i^*, respectively (both have the weight 2^n).

Now we will informally verify that the algorithm (12) satisfies the valid range of (6) for all P_i's ($i = 0, 1, \cdots, n$). Obviously $0 \le P_0 < 3 \cdot 2^n - 3$. Suppose $0 \le P_{i-1} < 3 \cdot 2^n - 3$, then

$$\begin{aligned} 0 &\le 2P_{i-1} \\ &< 6 \cdot 2^n - 6 \\ 0 &\le \sigma(2P_{i-1}) \\ &< 6 \cdot 2^n - 6 - 4 \cdot 2^n + 2^n \\ &= 3 \cdot 2^n - 6 \\ 0 &\le T_i^* \\ &< 4 \cdot 2^n - 6 \end{aligned}$$

[1] Subscript is for an iteration stage and superscript is for denoting bit positions, that is, $S_i = \sum_{j=0}^{n-1} s_i^j \cdot 2^j$. Also note that lower case letters are used for bit-level variables while upper case is for word-level variables.

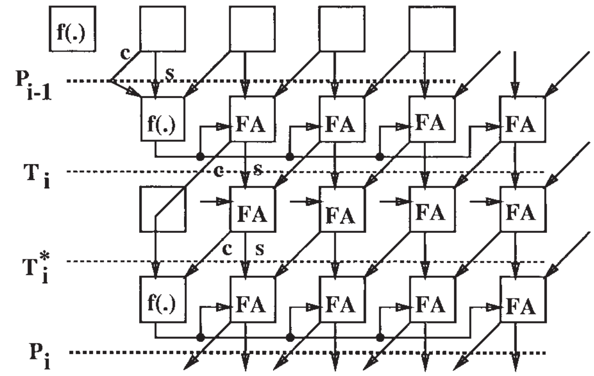


Fig. 1. An iteration stage of modular multiplication using the CSA scheme.

$$\begin{aligned} 0 &\le \sigma(T_i^*) \\ &< 4 \cdot 2^n - 6 - 2 \cdot 2^n + 2^n \\ &< 3 \cdot 2^n - 3 \end{aligned}$$

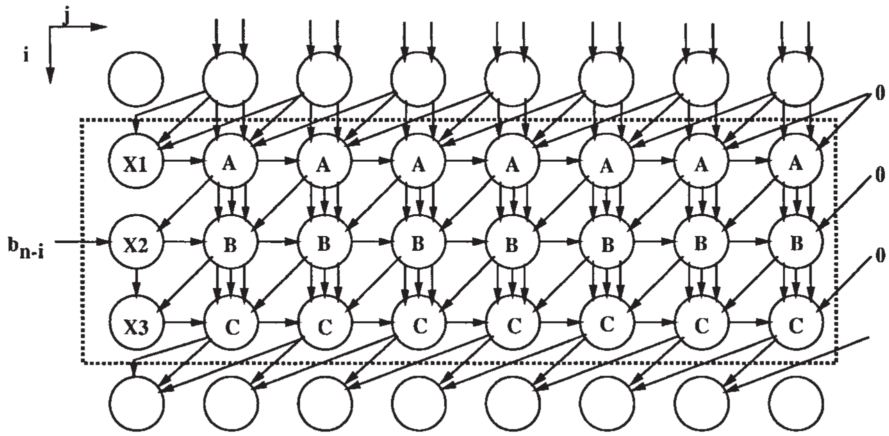
which assures $0 \le P_i < 3 \cdot 2^n - 3$. Therefore the algorithm (12) produces a final output P_n which is less than $3 \cdot 2^n - 3$. It can be directly fed into the next multiplication stage for further iteration if necessary (e.g., exponentiation).

Fig. 2(a) shows a single stage of the dependence graph (DG) which can be directly implemented as a parallel array multiplier. Fig. 2(b) describes the node functions. The nodes X_1, X_3 are control nodes which calculate the control value h of (10), and hence need simple encoders. The node X_2 is just a wire. Node type A is a FA with a 4-1 multiplexor and an AND gate. Node type B is just a FA with an AND gate and node type C is a FA with a 2-1 multiplexor and an AND gate. Note that node type B does not need a multiplexor and type C needs only K_1 and K_2 because the max value of h is two in node X_3. An AND gate is needed in type A and C to accept $K_0 = 0$ when the control value h is zero. There exists a systolic schedule which is not linear due to its skewed connection between the stages. Table I shows an example for our new bit-level modular multiplication algorithm using $n = 12$, with $A = 010001000100 (= 1092)$, $B = 010011001101 (= 1229)$, and $C = 100000101001 (= 2089)$. The K_h's are precalculated as $K_1 = K = 011111010111 (= 2007)$, $K_2 = 011110000101 (= 1925)$, $K_3 = 011100110011 (= 1843)$, $K_4 = 011011100001 (= 1761)$. The final output is

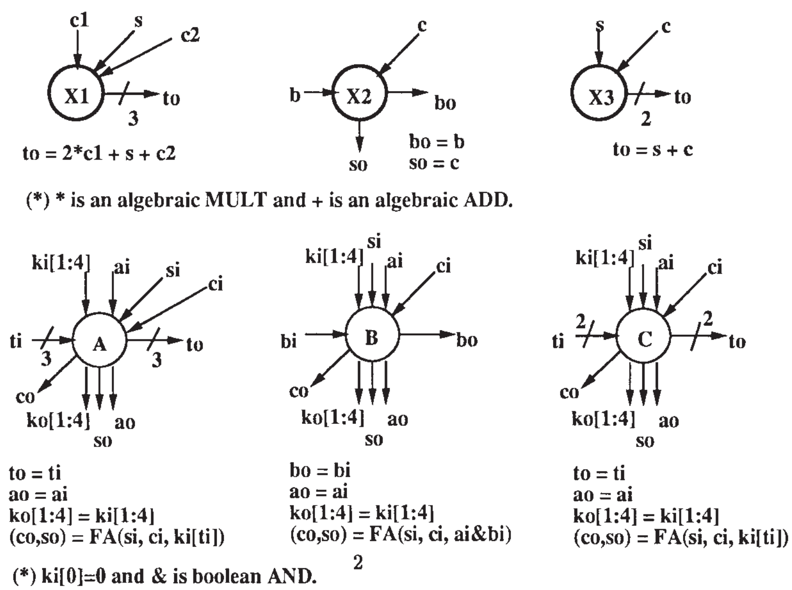
$$\begin{aligned} P_n &= 2(001100010101) + (110111001010) \\ &= 1001111110100 \\ &(= 5108) \end{aligned}$$

which equals 930 after modulo reduction to 2089.

By merging two nodes into one in each row as has been done in [5], one can modify the DG in Fig. 2 to derive a simpler DG. This is shown in Fig. 3. Node types AA, BB, CC are newly merged nodes which have two A, B, C type nodes, respectively. It now allows a linear systolic schedule and shows a better overview of the hardware array implementation. If the wordlength n is an even number, then all nodes except the control nodes will be merged nodes. Here we have the original nodes A, B, C in the LSB place because n is odd. From



(a)



(b)

Fig. 2. Array structure for modular multiplication using CSA scheme ($n = 7$): (a) ith stage of DG for modular multiplication ($n = 7$) and (b) PE node description.

Figs. 2 or 3, we can obtain many different one-dimensional arrays (e.g., bit-serial modulo multiplier) depending on the mapping functions [7].

In our array, the control is generated in a single left-most node which has just four gates (two XOR, one AND, one NOR gate) or two gates (one XOR and one NOR gate). The simplicity of the control nodes gives a much faster clock cycle for the entire array. Thus, it is not the control node but the processing node which determines the clock cycle. Note that all signals in Fig. 2 except the carry (c_o) and sum (s_o) are *transmittent* signals, which means they are not modified while passing through the array, thus allowing for broadcasting.

Compared to [5], the dependency structure looks the same except for the control nodes due to the basic CSA scheme. However, the main difference is in the function of the control nodes. In [5], the control node (denoted as X^5, LU^5, LY^5) is made by merging five or six identical processing nodes and each one (X, L, Y, U) is fairly complex (roughly three XOR's and more than five AND or OR gates). Therefore, the resulting control nodes become five or six times larger, and this is the critical reason for the slow clock cycle time. It is also worthwhile to compare with [4] which claims to be the fastest structure. Our array gives a faster clock cycle (two XOR with one MUX versus five XOR) although it has a longer latency. There is also no restriction on choosing the modulus, thus allowing more general application. Furthermore, we do not need a post processing step because we keep the correct value in all iteration stages. Both methods require a precalculation.

B. Using a Modified CSA Scheme

Now we derive a far simpler array structure from the algorithm (3) by slightly modifying the basic element of the CSA scheme. To directly apply the CSA structure to the modular multiplication algorithm, we have to modify the basic element so that it can accept an additional operand which

TABLE I
EXAMPLE SHOWING COMPLETE FUNCTION OF FIG. 2 ($N = 12$)

stage		(carry,sum)	f(.)	b_{n-i}
0	P_0	00 00 00 00 00 00 00 00 00 00 00 00	-	-
1	T_1	00 00 00 00 00 00 00 00 00 00 00 00	f(.)=0	
	T_1^*	00 00 00 00 00 00 00 00 00 00 00 00		0
	P_1	00 00 00 00 00 00 00 00 00 00 00 00	f(.)=0	
2	T_2	00 00 00 00 00 00 00 00 00 00 00 00	f(.)=0	
	T_2^*	00 01 00 00 01 01 00 00 01 01 00 01		1
	P_2	00 01 00 00 01 01 00 00 01 01 00 01	f(.)=0	
3	T_3	01 00 00 01 01 00 00 01 01 00 01 00	f(.)=0	
	T_3^*	01 00 00 01 01 00 00 01 01 00 01 00		0
	P_3	01 00 00 01 01 00 00 01 01 00 01 00	f(.)=0	
4	T_4	00 01 10 10 01 01 01 10 00 10 01 01	f(.)=1	
	T_4^*	00 10 01 00 01 01 10 00 01 00 01 01		0
	P_4	01 00 01 00 01 10 00 00 01 00 01 01	f(.)=0	
5	T_5	00 10 01 11 01 01 00 10 00 10 10 01	f(.)=1	
	T_5^*	01 00 10 01 01 01 01 00 01 01 00 01		1
	P_5	01 01 00 01 01 01 01 00 01 01 00 01	f(.)=0	
6	T_6	01 01 10 10 10 10 00 10 01 01 10 01	f(.)=1	
	T_6^*	01 11 01 01 10 01 01 00 10 11 00 10		1
	P_6	10 01 01 10 00 01 01 01 01 01 01 00	f(.)=0	
7	T_7	01 11 01 01 10 01 01 01 01 10 00 01	f(.)=2	
	T_7^*	10 01 01 10 00 01 01 01 10 00 00 01		0
	P_7	00 10 11 01 01 10 01 11 00 01 01 10	f(.)=1	
8	T_8	01 10 10 11 01 11 01 01 01 11 01 01	f(.)=1	
	T_8^*	10 01 01 01 10 01 01 01 10 01 01 01		0
	P_8	00 10 10 11 01 10 01 11 00 10 10 10	f(.)=1	
9	T_9	01 10 10 11 01 11 01 10 01 10 01 01	f(.)=1	
	T_9^*	10 01 01 01 10 01 10 00 10 00 01 01		1
	P_9	00 10 10 11 01 11 00 10 00 01 10 10	f(.)=1	
10	T_{10}	01 10 10 11 10 10 00 01 10 10 01 01	f(.)=1	
	T_{10}^*	10 10 01 10 10 01 00 10 10 01 01 10		1
	P_{10}	01 01 11 10 01 10 01 10 00 10 11 01	f(.)=1	
11	T_{11}	10 11 01 11 01 11 00 10 01 10 10 01	f(.)=1	
	T_{11}^*	01 01 10 01 10 01 01 00 10 01 00 01		0
	P_{11}	01 11 01 11 01 10 01 10 00 10 01 10	f(.)=1	
12	T_{12}	01 11 10 11 01 10 00 01 00 11 00 01	f(.)=2	
	T_{12}^*	10 10 01 01 10 00 00 01 01 01 00 01		1
	P_{12}	01 01 10 11 01 01 00 10 01 10 01 10	f(.)=1	

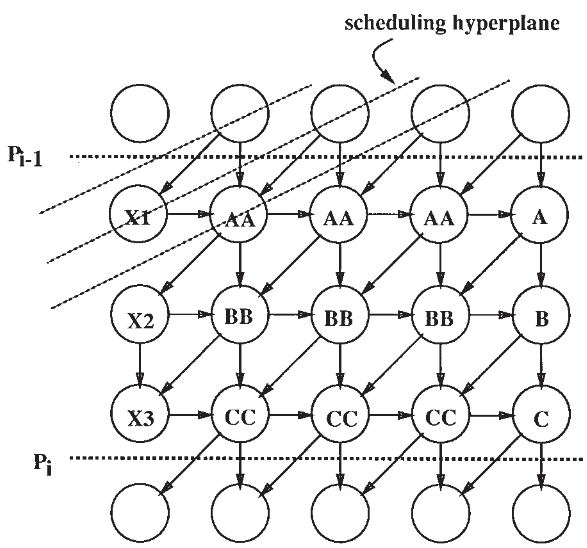


Fig. 3. Regularized DG of Fig. 2 by node merging.

arises from end-around carry terms in the MSB nodes. Now we extend the $(3, 2)$ counter to generate a more general adder, which we call a *partially generalized counter* (PGC), as shown

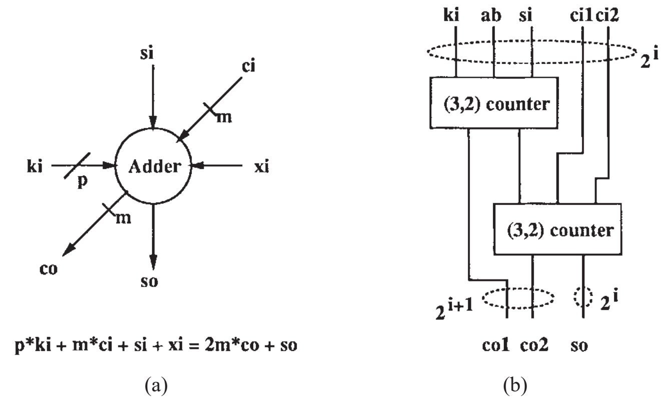


Fig. 4. A partially generalized counter (PGC) and a new adder logic: (a) a PGC element and (b) an adder derived from (a) with $m = 2$, $p = 1$.

in Fig. 4. In Fig. 4(a), s_i, c_i are a partial sum and a carry of the previous stage. Note that the carry signal is not a single bit, but m-bits. The variable x_i is a regular operand which appears in a typical array integer multiplication as $a_i \& b_j$(& is logical AND). An extra operand k_i is introduced with a p-bit signal for handling the end-around carry terms. So, the arithmetic operation of PGC can be described by

$$2m \cdot c_o + s_o = s_i + m \cdot c_i + x_i + p \cdot k_i \tag{13}$$

where "$\cdot$" and "$+$" are algebraic multiplication and addition. If $m = 1$, the only possible value of p is zero, which makes it a typical FA. Therefore, we see that the m should be at least two to accept an additional operand resulting from end-around carrys. Furthermore, if $m = 2$ the only possible value of p is one, which results in five inputs and three outputs. This can be easily implemented using two FA's as shown in Fig. 4(b). We can also derive different adders by choosing different p and m. However, we want to minimize the number of signal lines to reduce the complexity of the node function.

To apply the newly derived adder in the CSA structure, we need a different CSA form (two carry terms) to express P_i. Let us denote

$$P_i \equiv 2(C_{i1} + C_{i2}) + S_i, \quad (0 \le i \le n) \tag{14}$$

then, the valid range of P_i is $0 \le P_i \le 5 \cdot 2^n - 5$. Following the same procedure as the CSA scheme, let us say, $C_{i1} = \sum_{j=0}^{n-1} c_{i1}^j 2^j$, $C_{i2} = \sum_{j=0}^{n-1} c_{i2}^j 2^j$, $S_i = \sum_{j=0}^{n-1} s_i^j 2^j$. Then, the control generating function $f(\cdot)$ is

$$f(\cdot) = 2(c_{(i-1)1}^{n-1} + c_{(i-1)2}^{n-1}) + (s_{i-1}^{n-1} + C_{(i-1)1}^{n-2} + c_{(i-1)2}^{n-2}). \tag{15}$$

Note that we need seven complement integers, $K_1, K_2, \cdots, K_7$. Fig. 5 shows an array structure for the modified CSA scheme with node functions. The node type D is a serial connection of two full adders with an 8-1 multiplexor. For the example shown in the previous section, additional K_h's should be precalculated as $K_5 = 011011100001 (= 1679)$, $K_6 = 011011100001 (= 1597)$, $K_7 = 011011100001 (= 1515)$.

The range of the final output P_n can be further reduced using two more basic CSA stages as shown in Fig. 6. The first extra CSA stage adds up to sequences of carrys and a

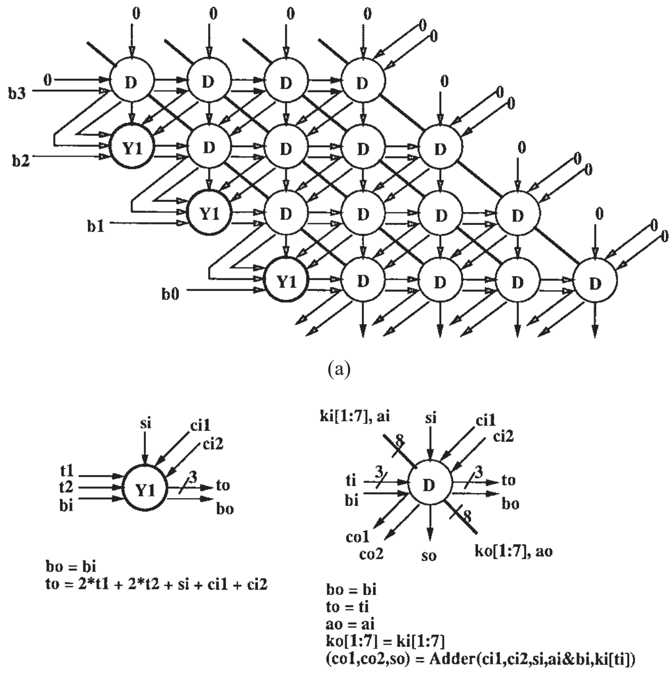


Fig. 5. Array structure for modular multiplication using modified CSA scheme: (a) DG for modular multiplication using CSA scheme ($n = 4$) and (b) PE descriptions.

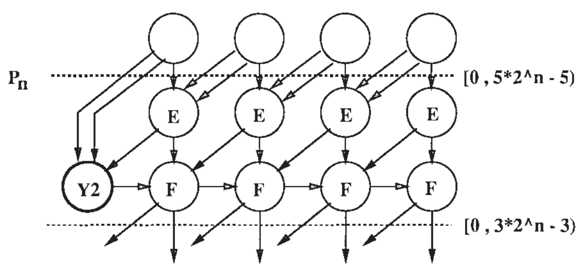


Fig. 6. Extra stages to reduce the range from $5 \cdot 2^n - 5$ to $3 \cdot 2^n - 3$.

sum of $P_n$except for the MSB carrys. The MSB carrys of P_n are compensated for in the second extra CSA stage. The range $5 \cdot 2^n - 5$ of P_n is reduced to $3 \cdot 2^n - 3$ by replacing maximum $3 \cdot 2^n$ with $K_h (K_h < 2^n)$. At the second CSA stage of Fig. 6, a 4-1 multiplexor is needed instead of an 8-1 multiplexor.

By running SIS [11], a multilevel logic minimization tool, with arbitrary encoding, the control note Y_1 had eight OR gates (including one, three-input OR), nine AND gates (including two, three-input AND), and three inverters, and the combinational logic depth was 12. This is reasonably close to the function of node D (2-bit FA with a multiplexor)[2] in both area and speed. Here again note that all signals are transmittent except carrys and a sum.

III. Conclusion and Discussion

We have shown two new array structures for modular multiplication. Both use the basic CSA structure with some

[2] We used the common mapping to two-input OR gates to measure the logic depth, hence the 2-bit FA has logic depth 9.

TABLE II
Summary for Logic of PE Nodes in Two Array Structures

array 1 (Fig.2)				
X1	X3	A	B	C
(5,3)encoder	(3,2)encoder	(4:1) mux FA AND gate	FA AND gate	(2:1) mux FA AND gate
array 2 (Fig.5 and Fig.6)				
Y1	Y2	D	E	F
(8,3) encoder	(4,2) encoder	(8:1) mux 2 FAs AND gate	FA	(4:1) mux FA

modification and do not need any number translation. The complement and its multiples need be updated only when the modulus changes. In RSA applications, the key does not change very often therefore this preprocessing step is acceptable. The details of each processing node in the two array structures are summarized in Table II. The first array needs processing nodes of $n(\mathrm{X}_1 + \mathrm{X}_3) + n^2(\mathrm{A} + \mathrm{B} + \mathrm{C})$, and the second needs $(n-1)\mathrm{Y}_1 + \mathrm{Y}_2 + n^2\mathrm{D} + n(\mathrm{E} + \mathrm{F})$. In each iteration stage, the second array has a smaller number of FA's but the first one has simpler encoders and a multiplexor which lead to a faster clock cycle. Both have a systolic schedule which means they can be fully bit-wise systolized for maximum throughput.

Our approach has advantages over other previous array structures. First, it is more general and has no restrictions in choosing the modulus, hence can be used in any application in which a larger number of computations are required for a relatively long-lasting modulus (RSA, key-exchange, special purpose DSP chip). Second, the algorithm and architecture is simple to understand and verify, hence is easy to modify for a hardware implementation, and does not use any special technique like sign estimation which may imply a significant degree of hardware and verification complexity. We could obtain a faster clock cycle by reducing the complexity of the control nodes in both area and speed. From an algorithmic point of view, the concerns put forward in most previous papers on deciding multiples of the modulus have been eliminated by multiplexing precalculated complement numbers. We have verified our array structures first by C programming and then by implementing two prototype VLSI designs which were verified for function and timing using logic and circuit simulation.

Finally, the RSA algorithm requires more than 500 bits for security reasons, which may make the array multiplier too large (~15 million gates including pipeline stages). To maintain high performance for larger bit-lengths, we need to map the original array onto smaller processor arrays in order to build large "virtual" modular multipliers on fixed sized arrays, called *partitioning* [7]. The regularity of our algorithm makes it easy to find an appropriate partitioning strategy. Because the control nodes which generate the multiplexor control signals are located in the MSB in each iteration, the LPGS (*locally parallel, globally sequential* [7]) scheme is an appropriate choice, needing some extra buffers outside the processor array to contain intermediate data for each block.

ACKNOWLEDGMENT

The authors would like to acknowledge the work of the referees whose reviews and comments were very meticulous and insightful.

REFERENCES

[1] E. B. Brickell, "A fast modular multiplication algorithm with application to two key cryptography," in *Advances in Cryptology, Proceedings of Crypto 82.* New York: Plenum, 1982, pp. 51–60.

[2] G. R. Blakley, "A computer alrogithm for the product AB modulo M," *IEEE Trans. Comput.,*vol. 32, pp. 497–500, 1983.

[3] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Inform. Theory,* vol. IT-22, pp. 644–654, Nov. 1976.

[4] S. E. Eldridge and D. Walter, "Hardware implementation of Montgomery's modular multiplication algorithm," *IEEE Trans. Comput.,* vol. 42, pp. 693–699, June 1993.

[5] C. K. Koc and C. Y. Hung, "Bi-level systolic arrays for modular multiplication," *J. VLSI Sig. Proc.,* vol. 3, pp. 215–223, 1991.

[6] I. Koren, *Computer Arithmetic Algorithms.* Englewood Cliffs, NJ: Prentice-Hall, 1993.

[7] S. Y. Kung, *VLSI Array Processors.* Englewood Cliffs, NJ: Prentice-Hall, 1988.

[8] X. Lai and J. L. Massey, "A proposal for a new block encryption standard," in *EUROCRYPT '90,*Aarhus, Denmark, May 1990.

[9] P. L. Montgomery, "Modular multiplication without trial division," *Math. Comp.,* vol. 44, pp. 519–521, 1985.

[10] R. L. Rivest, A. Shamir, and L. Adelman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM,* vol. 21, pp. 120–126, Feb. 1978.

[11] E. M. Sentovich, K. J. Singh, C. Moon, H. Savoy, and R. K. Brayton, "Sequential circuit design using synthesis and optimization," in *Proc. ICCD,* 1992.

[12] N. Takagi, "A radix-4 modular multiplication hardware algorithm for modular exponentiation," *IEEE Trans. Comput.,* vol. 41, pp. 949–956, Aug. 1992.

[13] F. J. Taylor, "Residue arithmetic: A tutorial with examples," *IEEE Comput. Mag.,* vol. 17, pp. 50–62, May 1984.

[14] A. Vandemeulebroecke *et al.,* "A new carry-free division algorithm and its application to a single chip 1024-b RSA processor," *IEEE J. Solid-State Circuits,* vol. 25, pp. 748–755, June 1990.

[15] C. D. Walter, "Systolic modular multiplication," *IEEE Trans. Comput.,* vol. 42, npp. 376–378, Mar. 1993.

[16] R. Zimmermann *et al.,* "A 177 Mb/s VLSI implementation of International data encryption algorithm," *IEEE J. Solid-State Circuits,* vol. 29, pp. 303–307, Mar. 1994.

Yong-Jin Jeong (M'96) received the B.S. degree in control and instrumentation engineering from the Seoul National University, Seoul, Korea, in 1983, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Massachusetts, Amherst, MA, in 1995.

He is now working for Samsung Electronic Co., Seoul, Korea, in the area of communication chipset design. From 1983 to 1989, he was a Member of the Research Staff at the Electronics and Telecommunications Research Institute (ETRI) in Daejun, Korea. He has also worked for the Swedish telecommunication company, LM Ericsson, Stockholm, Sweden, as a Training and Design Engineer. His research interests include computer arithmetic, parallel algorithm, and array synthesis, VLSI testing, logic synthesis, and wireless communication systems.

Wayne P. Burleson (M'89) received the B.S.E.E. and M.S.E.E. degrees from the Massachusetts Institute of Technology (M.I.T.), Cambridge, in 1983 and the Ph.D. degree in VLSI signal processing from the University of Colorado, Boulder, in 1989.

He has been exploring the area of VLSI signal processing for 13 years. His work has included research, development, teaching, and industrial work at a variety of levels, including development of algorithms, architectures, circuits, and computer-aided design (CAD) tools. He is currently an Assistant Professor of Electrical and Computer Engineering at the University of Massachusetts, Amherst. For four years, he worked as a custom DSP chip designer for VLSI Technology Inc., and Fairchild Semiconductor. He is currently being funded by the NSF, exploring advanced timing schemes for CMOS systems including circuit and system design, clocking methods, and verification techniques which are all being developed to meet the needs of modern highly pipelined architectures with tight latency requirements. He is also currently developing a CAD system for VLSI arrays with applications in digital signal processing (DSP) and communications. The ARRay ESTimator (ARREST) system allows a broad range of algorithm and architecture exploration, interface with simulation and estimation tools, and finally output to VERILOG for backend synthesis. In a more applied area, he is currently collaborating with the University of Massachusetts' researchers in real-time systems, computer vision and robotics, and the development and implementation of special-purpose coprocessors in advanced technologies.

Dr. Burleson is a member of the ACM and Sigma Xi.

Attachment 6C

OCLC WorldCat®

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

IEEE Transactions on Very Large Scale Integration (VLSI) Systems : a joint publication of the IEEE Circuits and Systems Society, the IEEE Computer Society, the IEEE Solid-State Circuits Council.

Get a Copy

Find a copy in the library

Author:	Institute of Electrical and Electronics Engineers.; IEEE Circuits and Systems Society.; Institute of Electrical and Electronics Engineers. Computer Society.; Institute of Electrical and Electronics Engineers. Solid-State Circuits Council.
Publisher:	New York, N.Y. : IEEE, 1993-.
Edition/Format:	Journal, magazine : English View all editions and formats
Rating:	(not yet rated) 0 with reviews - Be the first.

Search this publication for other articles with the following words:

Search

Find a copy online

Links to this item

ieeexplore.ieee.org

Find a copy in the library

Enter your location: library of congress Find libraries

Submit a complete postal address for best results.

Displaying libraries 1-6 out of 530 for all 23 editions (101 Independence Ave SE, Washington, DC 20540, USA)

Show libraries holding **just this edition** or narrow results by **format**

« First ‹ Prev 1 2 3 Next › Last »

	Library	Held formats	Distance	
1.	**Library of Congress** Washington, DC 20540 United States	Journal / Magazine / Newspaper	< 1 mile MAP IT	Library info Ask a librarian Add to favorites
2.	**Federal Communications Commission** Washington, DC 20554 United States	Journal / Magazine / Newspaper	1 mile MAP IT	Library info Add to favorites
3.	**NATIONAL TRANS LIBRARY** Washington, DC 20590 United States	Journal / Magazine / Newspaper	1 mile MAP IT	Library info Add to favorites

4.	**George Washington University** Washington, DC 20052 United States	Journal / Magazine / Newspaper	2 miles MAP IT	Library info Ask a librarian Add to favorites
5.	**Howard University** Washington, DC 20059 United States	Journal / Magazine / Newspaper	2 miles MAP IT	Library info Ask a librarian Add to favorites
6.	**Research Center, National Academies of Sciences, Engineering, and Medicine** Washington, DC 20001 United States	Journal / Magazine / Newspaper	2 miles MAP IT	Library info Add to favorites

« First ‹ Prev 1 2 3 Next › Last »

Details

Material Type:	Internet resource
Document Type:	Journal / Magazine / Newspaper, Internet Resource
All Authors / Contributors:	Institute of Electrical and Electronics Engineers.; IEEE Circuits and Systems Society.; Institute of Electrical and Electronics Engineers. Computer Society.; Institute of Electrical and Electronics Engineers. Solid-State Circuits Council.
ISSN:	1063-8210
OCLC Number:	750041125
Notes:	Tyt. okł.
Description:	28 cm.
Other Titles:	IEEE transactions on very large scale integration (VLSI) systems Very Large Scale Integration (VLSI) Systems

Reviews

User-contributed reviews

Add a review and share your thoughts with other readers. Be the first.

Tags

Add tags for "IEEE Transactions on Very Large Scale Integration (VLSI) Systems : a joint publication of the IEEE Circuits and Systems Society, the IEEE Computer Society, the IEEE Solid-State Circuits Council.". Be the first.

Linked Data

Attachment 6D

LIBRARY | CATALOG

1 of 1

PERIODICAL OR NEWSPAPER

IEEE transactions on very large scale integration (VLSI) systems

Full Record MARC Tags

Main title

IEEE transactions on very large scale integration (VLSI) systems.

Published/Created

New York, NY : Institute of Electrical and Electronics Engineers, c1993-

Request this Item LC Find It

More Information

LCCN Permalink	https://lccn.loc.gov/93641510
Publication history	Vol. 1, no. 1 (Mar. 1993)-
Description	v. : ill. ; 28 cm.
Current frequency	Bimonthly, <June 2002->
Former frequency	Quarterly, 1993-
ISSN	1063-8210
Linking ISSN	1063-8210

Request this Item LC Find It

Institute of Electrical and Electronics Engineers transactions on very large scale integration (VLSI) systems

Portion of title	Very large scale integration (VLSI) systems
Serial key title	IEEE transactions on very large scale integration (VLSI) systems (Print)
Abbreviated title	IEEE trans. very large scale integr. (VLSI) syst. (Print)
Related names	Institute of Electrical and Electronics Engineers. IEEE Circuits and Systems Society. IEEE Computer Society. IEEE Solid-State Circuits Council.
LC Subjects	Integrated circuits--Very large scale integration--Design and construction--Periodicals.
Other Subjects	Integrated circuits--Very large scale integration--Design and construction. VLSI Zeitschrift Online-Ressource
Form/Genre	Periodicals.
Browse by shelf order	TK7874
Notes	Title from cover. A joint publication of: IEEE Circuits and Systems Society, IEEE Computer Society, and IEEE Solid-State Circuits Council.
Reproduction no./Source	IEEE Service Center, 445 Hoes La., POB 1331, Piscataway, NJ 08855-1331
Additional formats	Also issued online. IEEE transactions on very large scale integration (VLSI) systems (Online) 1557-9999 (DLC) 2005215205 (OCoLC)44617287
LCCN	93641510
Invalid LCCN	sn 92000187
CODEN	IEVSE9
Dewey class no.	621.39/5
National bib agency no.	009845046
Other system no.	(OCoLC)ocm26142392

Request this Item

LC Find It

Type of material	Periodical or Newspaper

Item Availability

CALL NUMBER	TK7874 .I3273 Set 1
Request in	Jefferson or Adams Building Reading Rooms
Status	c.1 v. 23, no. 1-6 2015 Jan-June In Process 05-09-2016
Older receipts	v.1-v.14:no.5 (1993-2006:May), v.14:no.7-v.16:no.4 (2006:July-2008:Apr.), v.16:no.6-v.24:no.12 (2008:June-2016:Dec.)

CALL NUMBER	TK7874 .I3273 Copy 1 Unbound issues
Request in	Newspaper & Current Periodical Reading Room (Madison LM133)
Latest receipts	v. 26, no. 8 (2018 Aug.) v. 26, no. 7 (2018 July) v. 26, no. 6 (2018 June) v. 26, no. 5 (2018 May) v. 26, no. 4 (2018 Apr.) v. 26, no. 3 (2018 Mar.) v. 26, no. 2 (2018 Feb.) v. 26, no. 1 (2018 Jan.) v. 25, no. 12 (2017 Dec.) v. 25, no. 11 (2017 Nov.) v. 25, no. 10 (2017 Oct.) v. 25, no. 9 (2017 Sept.) v. 25, no. 8 (2017 Aug.) v. 25, no. 7 (2017 July) v. 25, no. 6 (2017 June) v. 25, no. 5 (2017 May) v. 25, no. 4 (2017 Apr.) v. 25, no. 3 (2017 Mar.) v. 25, no. 2 (2017 Feb.) v. 25, no. 1 (2017 Jan.)

Request this Item

LC Find It

Attachment 6E

LIBRARY CATALOG

PERIODICAL OR NEWSPAPER

IEEE transactions on very large scale integration (VLSI) systems

Full Record MARC Tags

Main title

IEEE transactions on very large scale integration (VLSI) systems.

Published/Created

New York, NY : Institute of Electrical and Electronics Engineers, c1993-

Request this Item LC Find It

More Information

LCCN Permalink	https://lccn.loc.gov/93641510
Publication history	Vol. 1, no. 1 (Mar. 1993)-
Description	v. : ill. ; 28 cm.
Current frequency	Bimonthly, <June 2002->
Former frequency	Quarterly, 1993-
ISSN	1063-8210
Linking ISSN	1063-8210

Request this Item LC Find It

Institute of Electrical and Electronics Engineers transactions on very large scale integration (VLSI) systems

Portion of title	Very large scale integration (VLSI) systems
Serial key title	IEEE transactions on very large scale integration (VLSI) systems (Print)
Abbreviated title	IEEE trans. very large scale integr. (VLSI) syst. (Print)
Related names	Institute of Electrical and Electronics Engineers. IEEE Circuits and Systems Society. IEEE Computer Society. IEEE Solid-State Circuits Council.
LC Subjects	Integrated circuits--Very large scale integration--Design and construction--Periodicals.
Other Subjects	Integrated circuits--Very large scale integration--Design and construction. VLSI Zeitschrift Online-Ressource
Form/Genre	Periodicals.
Browse by shelf order	TK7874
Notes	Title from cover. A joint publication of: IEEE Circuits and Systems Society, IEEE Computer Society, and IEEE Solid-State Circuits Council.
Reproduction no./Source	IEEE Service Center, 445 Hoes La., POB 1331, Piscataway, NJ 08855-1331
Additional formats	Also issued online. IEEE transactions on very large scale integration (VLSI) systems (Online) 1557-9999 (DLC) 2005215205 (OCoLC)44617287
LCCN	93641510
Invalid LCCN	sn 92000187
CODEN	IEVSE9
Dewey class no.	621.39/5
National bib agency no.	009845046
Other system no.	(OCoLC)ocm26142392

Request this Item

LC Find It

Type of material	Periodical or Newspaper

Item Availability

CALL NUMBER	TK7874 .I3273 Set 1
Request in	Jefferson or Adams Building Reading Rooms
Status	c.1 v. 23, no. 1-6 2015 Jan-June In Process 05-09-2016
Older receipts	v.1-v.14:no.5 (1993-2006:May), v.14:no.7-v.16:no.4 (2006:July-2008:Apr.), v.16:no.6-v.24:no.12 (2008:June-2016:Dec.)

CALL NUMBER	TK7874 .I3273 Copy 1 Unbound issues
Request in	Newspaper & Current Periodical Reading Room (Madison LM133)
Latest receipts	v. 26, no. 8 (2018 Aug.) v. 26, no. 7 (2018 July) v. 26, no. 6 (2018 June) v. 26, no. 5 (2018 May) v. 26, no. 4 (2018 Apr.) v. 26, no. 3 (2018 Mar.) v. 26, no. 2 (2018 Feb.) v. 26, no. 1 (2018 Jan.) v. 25, no. 12 (2017 Dec.) v. 25, no. 11 (2017 Nov.) v. 25, no. 10 (2017 Oct.) v. 25, no. 9 (2017 Sept.) v. 25, no. 8 (2017 Aug.) v. 25, no. 7 (2017 July) v. 25, no. 6 (2017 June) v. 25, no. 5 (2017 May) v. 25, no. 4 (2017 Apr.) v. 25, no. 3 (2017 Mar.) v. 25, no. 2 (2017 Feb.) v. 25, no. 1 (2017 Jan.)

Request this Item | LC Find It

Attachment 6F

RING-PLANARIZED CYLINDRICAL ARRAYS WITH APPLICATION TO MODULAR MULTIPLICATION*

William L. Freking and Keshab K. Parhi
Dept. of ECE, University of Minnesota,
200 Union St. SE, Minneapolis, MN 55455
{freking, parhi}@ece.umn.edu

Abstract

Cylindrical arrays have been shown useful for VLSI implementation of a variety of problems including matrix-matrix multiplication and algebraic path determination. However, spiral feedback paths limit their scalability due to performance degradation in interconnect-delay dominant environments. A recently proposed feedback-pipelining technique can efficiently address this problem when signal paths are non-diametric in the projection direction. However, this method may incur excessive penalties when the latter condition does not hold. In this paper, a new class of cylindrical array is proposed, the *ring-planarized cylindrical array*, which overcomes the barrier to efficient, fully-pipelined arrays projected in directions having diametric signal paths. In contrast to standard cylindrical arrays, processors from each cylinder row are distributed along planar ring structures rather than lines. This construction inherently constrains maximum signal path length to a constant, permitting efficient scalability. Application to the cryptographically relevant modular multiplication problem is demonstrated.

1 INTRODUCTION

Systolic arrays remain an essential architectural methodology due to inherent properties of modularity, regularity, local interconnection, and high degree of pipelining [1]. Moreover, as interconnect delay grows increasingly dominant as deep-submicron technology progresses to smaller dimensions, systolic techniques become even more relevant in the quest to attain the highest levels of performance.

Cylindrical arrays belong to the subset of array architectures which exhibit spiral interconnections, and have been utilized in problems such as matrix-matrix multiplication [2] and the algebraic path problem [3]. Although these arrays exhibit some interesting properties, they have been underutilized, due to the fact that the spiral feedback paths are not strictly local. Therefore, the interconnect delay of such paths can become a dominating factor as the array size increases.

*This research was supported by DARPA under grant number DA/DABT63-96-C-0050.

In [4], a folded array methodology and two alternatives were presented to address performance scalability in the modular multiplication problem. The rudimentary embodiment of the former technique can be effectively described in terms of mapping to a standard cylindrical array. However, it was further demonstrated that by applying proper temporal transformations to the array (in the form of rescheduling or cutset pipelining/retiming), the spiral-feedback problem may be addressed architecturally by rendering such signal paths as transmittent, nearest-neighbor interconnections. Although this method permits architectural scaling without interconnect-related clock-rate penalty, it will be demonstrated in this paper that excessive delay counts may be incurred when diametric signal paths exist in the projection direction.

To address this problem, this paper introduces a new cylindrical array paradigm, the *ring-planarized cylindrical array* (RPCA). In contrast to the planarization which yields standard 2-D cylindrical arrays, the new method distributes processors from each cylinder row along planar ring structures rather than lines. RPCA structure inherently limits maximum path length to a constant, eliminating array-size dependence. Modular multiplication for cryptographic applications serves as the context in which the RPCA methodology is applied in this paper. Paper organization is as follows. Section 2 provides background on the modular multiplication problem. Properties of standard cylindrical arrays are discussed in section 3, followed by the introduction of the RPCA technique. A comparison of relevant quantities is provided in section 4, followed by conclusions in section 5.

2 BACKGROUND: MODULAR MULTIPLICATION

A fundamental constituent of modern cryptography is the public-key class of cryptosystems, which enable secure transmission of information over public channels without requiring exchange of secret parameters between communicating parties. Modular exponentiation is the basis of many such methods including the popular RSA technique [5], and the modular multiplication operation is elemental to modular exponentiation algorithms. Efficient hardware implementations of modular multiplication are therefore essential to many modern cryptography applications.

Given positive integers A, B, and N, we define the modular multiplication computation as $AB \bmod N = AB - \lfloor \frac{AB}{N} \rfloor N$, i.e., determine the remainder of the product of A and B with respect to the modulus N. Large problem sizes in cryptographic applications (e.g. typical word lengths of 1024 bits or greater for RSA) necessitate efficient iterative modular multiplication algorithms, and many are available [6] [7].

Although the architectural techniques to follow in the next section may be applied generally to many of the available modular multiplication algorithms, we will focus attention on an example which may be deemed as typical in many respects. We choose a useful binary LSD-first algorithm first derived in [8] by algebraic manipulation of the Montgomery method [6], wherein the quotient evaluation step is rendered trivial. This algorithm also corresponds to a binary LSD-first form of the general-radix IRA algorithms described in [7]. The algorithm may be specified as:

0-7803-6488-0/00/$10.00 © 2000 IEEE

Algorithm 1 *Binary IRA Modular Multiplication Algorithm*

Inputs: N, $0 < A = \sum_{i=0}^{n} a_i 2^i < 2N$ (where $n = \lceil \log_2 N \rceil$), $0 < B < 2N$, $\hat{N} = \frac{1+N}{2} = \left|2^{-1}\right|_N$ (N odd)

Output: $0 \leq S_{n+1} < 2N$

$S_{-1} = 0$, $a_{n+1} = 0$, and let m_i denote $|S_{i-1}|_2$

for $i = 0$ to $n + 1$

$S_i = \frac{S_{i-1} - |S_{i-1}|_2}{2} + a_i B + m_i \cdot \hat{N}$

where $|x|_y$ denotes $x \bmod y$, $S_{n+1} = \left|AB \cdot 2^{-n-2}\right|_N + \epsilon N$ and $\epsilon \in \{0, 1\}$. Note that the least significant bit of the partial result of the previous iteration, S_{i-1}, directly selects the modular correction value, which is a member of $\left\{0, \hat{N}\right\}$. Moreover the first term on the right-hand-side of the expression for S_i consists of S_{i-1} truncated by one bit. Finally, note the presence of a weighting factor in the final result related to the total number of iterations executed. This property is common to any LSD-first modular multiplication algorithm, and is addressed by operand prescaling in the manner prescribed by Montgomery [6].

We now rewrite the core of algorithm 1 in terms of bit-wise computations in algorithm 2 below, having the goal of deriving bit-level systolic modular multiplication arrays [9] [10] [11]. Note that within algorithm 2, assuming a single carry would be invalid (i.e., a 4-operand addition cannot be represented by single sum and carry signals). However, assuming two carries of equal weight reveals no contradiction. Furthermore, the computation is cast to reflect carry-ripple rather than carry-save dependencies to avoid redundant representation of the final result, S_{n+1}. However, to account for a consequential single possible carry at most significant bit positions, an additional j-loop iteration is added so that the $S_{i,n+1}$ output may pass this output to iteration $i + 1$.

Algorithm 2 *Bit-wise Modular Multiplication*

$S_{-1,j} = 0$ for all j, $S_{i,n+2} = 0$ for all i, and $a_{n+1}, b_{n+1}, \hat{N}_n, \hat{N}_{n+1} = 0$

for $i = 0$ to $n + 1$

for $j = 0$ to $n + 1$

let $m_i = S_{i-1,0}$

$S_{i,j} + 2c_{i,j}^{(1)} + 2c_{i,j}^{(2)} = S_{i-1,j+1} + a_i b_j + m_i \hat{N}_j + c_{i,j-1}^{(1)} + c_{i,j-1}^{(2)}$

Figure 1(a) depicts a dependence graph (DG) based on the above bit-wise algorithm for a very small problem size (i.e., $n = 5$). Also shown is the cell I/O description and a valid linear schedule corresponding to $s^T = [1\ 2]$ (using a $[j\ i]$ convention). Mapping via a [0 1] projection results in the linear array in figure 1(b). Such an array achieves 100% utilization when two independent data streams are interleaved. These data sets are distinguished in the figure by bracketed superscripts. A new pair of data sets may be introduced roughly every $2n$ clock cycles, resulting in throughput proportional to $\frac{2}{2n} = \frac{1}{n}$.

An efficient array is also obtained by projecting in the [1 0] direction, as shown in figure 1(c). One advantage of this projection is single-ported output, i.e., the output is available serially at the bottom-most cell position. Another advantage was demonstrated in [12], where favorable word-length scalability properties were demonstrated with a shorter length, Montgomery carry-save array. Finally, it can be observed that a [1 0] projected array can accept a single new data set every $n + 2$ clock cycles, and dependent computations must be separated by at least $2(n+2)$ clock cycles. Neglecting initial latency, such an array outputs one result roughly every n clock cycles, resulting in throughput proportional to $\frac{1}{n}$.

3 CYLINDRICAL MODULAR MULTIPLICATION ARRAYS

3.1 Standard Cylindrical Arrays and Feedback Pipelining

Having examined two linear systolic arrays for modular multiplication, we now consider the question of how to achieve structures which enable a performance/area trade-off for array sizes in excess of a linear array but smaller than a full 2-D array. Three such approaches were introduced in [4]. The method of present interest involved applying the folding transformation [13] to a full 2-D array in the [0 1] projection direction. Figure 2 depicts a portion of a full 2-D array having delay assignments consistent with the scheduling shown in figure 1(a). In order to obtain an architecture having k rows (where $\frac{n+1}{2} \geq k > 1$), we fold a possibly extended 2-D array by a factor of $\lceil \frac{n+1}{k} \rceil$ as displayed in figure 3(a) for $k = 3$.

Close examination reveals that such folded arrays are cylindrical in structure, and are homologous to cylindrical arrays proposed for problems such as matrix-matrix multiplication [2] and the algebraic path problem [3]. In the case currently under consideration, such arrays are advantageous in that they can simultaneously interleave $2k$ independent data streams, with new sets of inputs or outputs arriving roughly every $2(n+1)+1$ clock cycles. Throughput is therefore roughly proportional to $\frac{k}{n}$ if a fixed clock rate is assumed, a fact which demonstrates the performance scalability available through this design approach. However, the fixed clock rate assumption may not remain valid as k is increased in interconnect delay dominant environments due to unmitigated feedback paths. This is evident in the small k factor example of figure 3(a), where two inter-cell traversals are required (from row 3 to 2, and 2 to 1) as opposed to the limit of one such traversal for an array having strictly nearest neighbor communications.

To remedy this problem, an efficient feedback pipelining method [4] may be employed. Applying cutset pipelining in the original 2-D array along horizontal, periodic feed-forward cutsets having period k, the corresponding folded array will exhibit pipelined feedback paths having an identical pipelining factor. For example, pipelining by one level along the cutsets indicated in figure 2 followed by folding such that $k = 3$ yields the array in figure 3(b). It is clear that each feedback path now contains at least two delay elements, which corresponds directly to the number of inter-cell traversals. This fact allows the construction of an architecture having strictly nearest neighbor communications as shown in figure 4, where the feedback paths are rendered as pass-through signals in the second cell row. Extension of the feedback pipelining procedure for larger k is straightforward and is displayed in the array of figure 3(c). Note that feedback pipelining effectively increases the interleave factor from $2k$ to $3k-2$ in order to obtain full hardware utilization and throughput proportional to $\frac{k}{n}$. Pipelining overhead incurred solely from feedback pipelining amounts to roughly $3kn$.

Having demonstrated the method for the [0 1] projection direction, we now pursue application of the methodology for the [1 0] projection. Folding of an unmodified full 2-D array in the [1 0] direction with $k=3$ results in the array shown in figure 5(a). Attempting to achieve feedback pipelining by the previous temporal transformation strategy proves fruitless, since vertical cutsets within the modular multiplication array are not feed-forward. Instead, all delays within the folded array must be scaled by a factor of two, resulting in the array of figure 5(b). For general k, all delay elements in the architecture are scaled by a factor $(k-1)\,D$ in order to fully pipeline the feedback paths, as shown in figure 5(c). Observe that this solution is not nearly as efficient as the [0 1] projection case, due to diametric signal paths along the projection direction. Pipelining overhead is substantially greater, amounting to $9k^2n$, and displays a quadratic rather than linear dependence on k. Similarly, the required interleave factor for full hardware utilization and throughput is elevated to $k\,(k-1)$.

4 Ring-Planarized Cylindrical Arrays

Since significant penalties are associated with feedback pipelining of folded arrays projected in directions having diametric feedback, the solution may be considered untenable – especially for all but the smallest k values. However, rather than abandoning such projectiôns in this problem and others wherein the above techniques apply, an alternative solution is now derived based on the cylindrical nature of such arrays.

Although a cylindrical array exhibits strictly nearest neighbor communications in its three-dimensional embodiment as depicted conceptually in figure 6(a), non-ideal, spiral feedback paths are introduced when the array is planarized in standard fashion into a practical two-dimensional structure as in figure 6(b). Procedurally, this planarization may be envisioned as unrolling the cylinder structure into the plane, such that processors on each cylinder row are mapped onto a corresponding line.

As seen in the previous subsection, attempts to lessen the impact of feedback path interconnect delay in such architectures through feedback pipelining is inefficient when diametric signal paths exist along the projection direction. However, as it will now be shown, an alternative solution exists which avoids the significant penalties involved with the standard cylindrical array manifestation. The new approach consists of planarizing the three-dimensional cylindrical array in a manner such that processors from each cylinder row are distributed along planar ring structures rather than on lines as in the standard procedure. Figure 6(c) illustrates the resultant two-dimensional structure, which we denote the *ring-planarized cylindrical array* (RPCA). Notice that although more signal paths exceed nearest-neighbor length in this rudimentary form of the RPCA, all lengths are bounded according to constants rather than exhibiting any dependence on the circumference of the cylinder (i.e., k in the previous development). This fact indicates that the new structure may be scaled without encountering additional overhead due to increasing signal path lengths, as will be demonstrated further below.

Utilizing the RPCA paradigm, we now derive modular multiplication arrays for the [1 0] projection direction. Noting that the maximum signal path length amounts to three inter-cell traversals, we pipeline the pre-folded, full 2-D array along horizontal cutsets by two levels, resulting in the RPCA of figure 7(a).

Although this solution is straightforwardly derived from the RPCA definition, it is not optimal for the particular problem under consideration. An alternative which yields less overall pipelining overhead may be derived by observing that the number of delays associated with each vertical dependency (corresponding to $\{b, N\}$ variable sets) must be identical to the sum of the delays encountered on each associated horizontal and diagonal member (corresponding to $\{a, m\}$ variable sets and S variables, respectively). Therefore, vertical paths always contain more delays than diagonals. Exploitation of this property is achieved through counter-clockwise circular shifting by one position of each successive ring as displayed in the modified RPCA of figure 7(b). Although the longest path still occurs on diagonal dependencies, signal assignment has necessarily been altered, i.e., diagonal and vertical paths now represent $\{b, N\}$ variable sets and S variables, respectively. Thus, vertical paths may contain fewer delays than diagonals, permitting a solution with less overhead. Namely, imposition of only one level of pipelining (as opposed to two) at horizontal cutsets in the pre-folded 2-D array achieves full pipelining on the longest path.

Figure 8 renders the modified RPCA in a strictly nearest-neighbor communication architectural form. Diagrams of signal propagation for upper and lower cells within the ring structures are displayed, with gray accents indicating pass-through signals. Note an interleave factor of k achieves full throughput, and new data sets are introduced every $n + k + 2$ time steps. In the latter figure, k additional time steps are present so that all a variables may be fed from the right hand side of the array structure, rather than requiring internal porting. Dependent computations must be separated by $3\,(n + k + 2)$ time steps. Finally, note again that such an array may be easily scaled to other k values (preferably even) without incurring additional overhead.

5 COMPARISON

A brief comparison of some distinguishing features of the various array structures is performed in this section. For each architecture, Table 1 displays the total number of required delay elements, the wire density of internal cells stated in terms of inputs, and the number of inter-cell traversals exhibited by the longest signal path in the architecture. As concluded earlier, it is obvious that the [1 0] projected feedback-pipelined standard cylindrical array exhibits a burdensome, quadratically increasing delay count with respect to k. However, the [0 1] projected counterpart

0-7803-6488-0/00/$10.00 © 2000 IEEE

Array Type	# Delays	Input Wires/Cell (int.)	Longest Path
Std. Cyl. [0 1]	$9k\,(n+2)$	10	$k-1$
Std. Cyl. [1 0]	$9k\,(n+2)$	12	$k-1$
Std. F.P. Cyl. [0 1]	$(12k-6)\,(n+2)$	10	1
Std. F.P. Cyl. [1 0]	$9k\,(k-1)\,(n+2)$	12	1
mod. RPCA [1 0]	$12k\,(n+2)$	10	1

Table 1: Comparison of properties of standard cylindrical arrays with and without feedback pipelining and the modified RPCA architecture

and the modified RPCA in the [1 0] direction both exhibit a relatively minor overhead of approximately 33% over the non-pipelined feedback arrays. Furthermore, we note that the modified RPCA [1 0] achieves the same wire density of 10 input wires per internal cell as the [0 1] projected cylindrical arrays, as compared to 12 for the [1 0] projected counterparts. Wire density is improved in the modified RPCA over the latter arrays since the former avoids feedback of the multiplicity of horizontal dependencies encountered in the modular multiplication problem.

The above comparison demonstrates that the modified RPCA [1 0] architecture is competitive with the standard feedback pipelined [0 1] cylindrical array. This is significant since the design difficulties arising from diametric signals in the projection direction have been overcome, allowing potential advantages of arrays projected in such a manner to be realized. For the modular multiplication problem, such unique advantages include single ported output, simple word length scaling, and easy partitioning (along horizontal, feed-forward cutsets) for flexible multi-chip or block layout implementation.

6 CONCLUSIONS

In this paper, a new architectural paradigm has been introduced, the ring-planarized cylindrical array. In contrast to standard cylindrical arrays, RPCAs eliminate spiral feedback paths which can prove problematic in interconnect delay dominant environments. Furthermore, the RPCA technique is clearly superior to feedback-pipelined standard cylindrical arrays when diametric signal dependencies exist in the projection direction. Within the context of modular multiplication, the new technique is significant since it permits architectural scalability beyond linear arrays for the horizontal projection. Such arrays have unique advantages of single-ported output, word-length scalability, and simple partitioning.

References

[1] S. Y. Kung, VLSI array processors, Prentice-Hall, 1988.

[2] W. A. Porter, J. L. Aravena, "Cylindrical arrays for matrix multiplication," in Proc. of the Twenty-Fourth Annual Allerton Conference on Communication, Control, and Computing, pp.595-602, 1986.

[3] P. Y. Chang and J. C. Tsay, "A family of efficient regular arrays for algebraic path problem," *IEEE Trans. Comput.*, vol. 43, no. 7, pp. 769-77, 1994.

[4] W. L. Freking and K. K. Parhi, "Performance-scalable array architectures for modular multiplication," in Proc. IEEE Int. Conf. on Application-Specific Systems, Architectures and Processors, pp. 149-160, 2000.

[5] R. Rivest, A. Shamir, L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Comm. ACM*, vol. 21, no. 2, pp. 120-126, 1978.

[6] P. L. Montgomery, "Modular multiplication without trial division," *Math. Comp.*, vol. 44, no. 170, pp. 519-521, 1985.

[7] W. L. Freking and K. K. Parhi, "A unified method for iterative computation of modular multiplication and reduction operations," in Proc. IEEE Int. Conf. on Computer Design, pp.80-7, 1999.

[8] P. Kornerup, "A systolic, linear-array multiplier for a class of right-shift algorithms," *IEEE Trans. Comput.*, vol. 43, no. 8, pp. 892-898, 1994.

[9] C. D. Walter, "Systolic modular multiplication," *IEEE Trans. Comput.*, vol.42, no.3, pp.376-8, 1993.

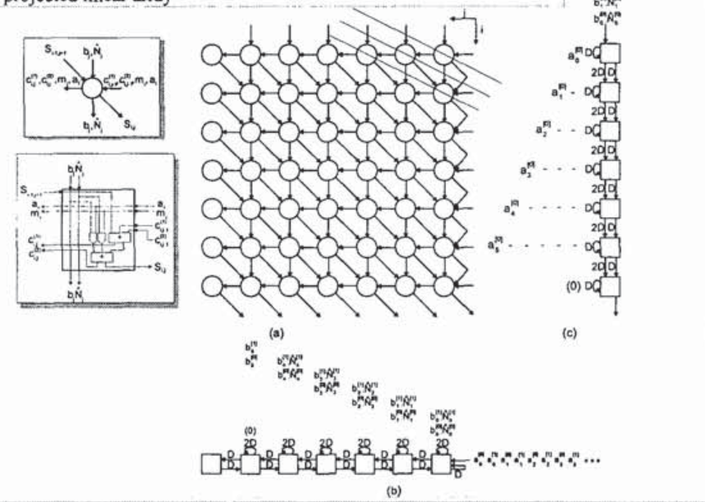
[10] Y. J. Jeong and W. P. Burleson, "VLSI array algorithms and architectures for RSA modular multiplication," *IEEE Trans. VLSI Syst.*, vol. 5, no. 2, pp. 211-217, 1997.

[11] W. C. Tsai, C. B. Shung, S. J. Wang, "Two systolic architectures for modular multiplication," *IEEE Trans. on VLSI Syst.*, vol. 8, no. 1, pp. 103-7, 2000.

[12] A. F. Tenca and C. K. Koc, "A scalable architecture for Montgomery multiplication," Cryptographic Hardware and Embedded Systems, LNCS No. 1717, pp. 94-108, 1999.

[13] K. K. Parhi, "High-level algorithm and architecture transformations for DSP synthesis," Journal of VLSI Signal Processing, v. 9, no. 1-2, pp.121-43, 1995.

Figure 1: Diagrams illustrating (a) the modular multiplication dependence graph and cell descriptions, (b) the [0 1] projected linear array, and (c) the [1 0] projected linear array



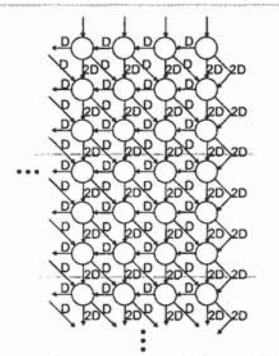


Figure 2: A full 2-D array having delay assignment in accordance with the scheduling of figure 1. Horizontal cutsets for a feedback pipelined k=3 case are illustrated

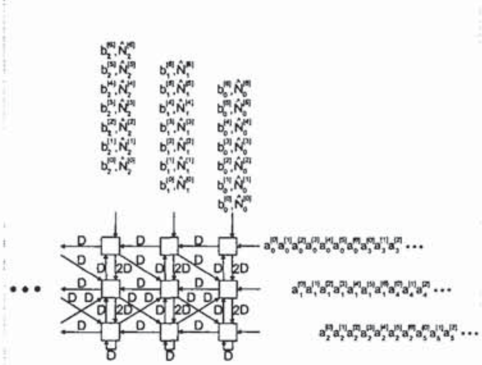


Figure 4: A nearest-neighbor communication, feedback-pipelined [0 1] projected array for k=3

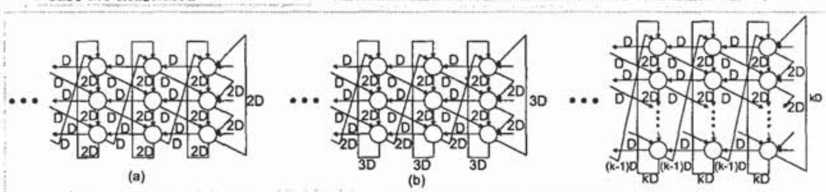
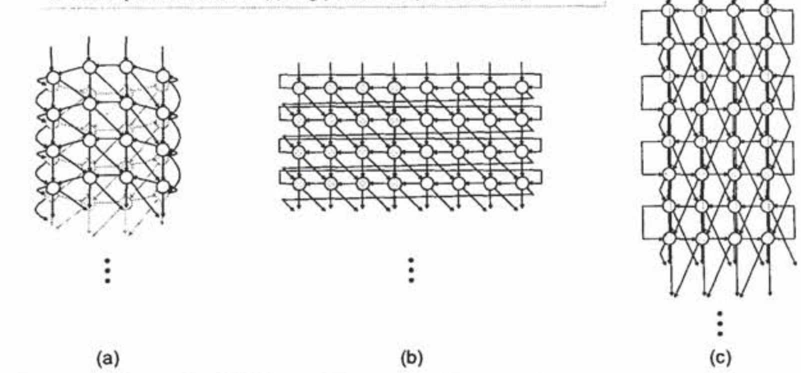


Figure 3: Folded [0 1] projected arrays for (a) k=3, (b) k=3 with feedback pipelining, and (c) feedback pipelined, general k

Figure 6: Diagrams depicting (a) three-dimensional cylindrical array, (b) standard planarization, and (c) ring-planarized cylindrical array



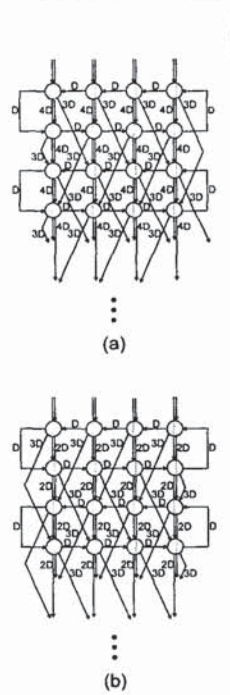


Figure 7: Diagrams of (a) straightforward application of RPCA technique to modular multiplication and (b) successively circularly-shifted RPCA

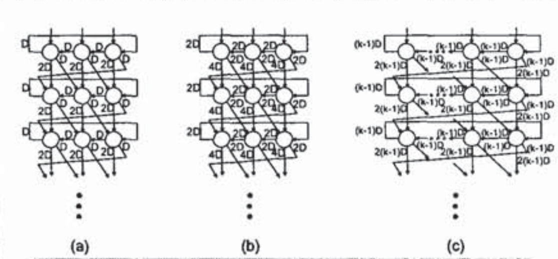


Figure 5: Diagrams for folded arrays in the [1 0] projection for (a) k=3, (b) k=3 with feedback pipelining, and (c) feedback pipelined, general k

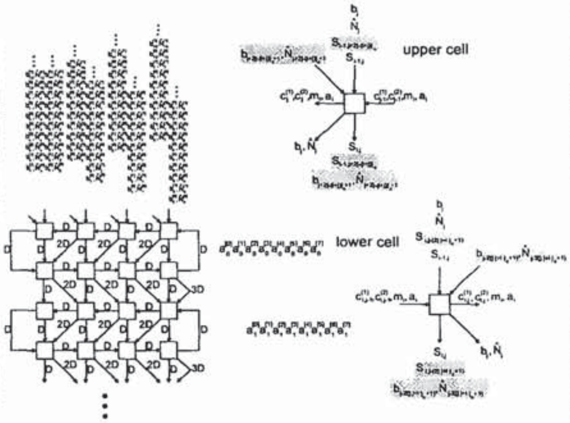


Figure 8: Illustration of a nearest neighbor communication, modified RPCA array in the [1 0] projection. Gray highlights indicate pass-through signals