

INVESTIGATION OF A METHOD TO IDENTIFY ENERGY
EFFICIENT ORGANIZATION OF COMPRESSION UNIT IN
PARALLEL TREE MULTIPLIERS

M. S. RASHID

JANUARY 2018

INVESTIGATION OF A METHOD TO IDENTIFY ENERGY
EFFICIENT ORGANIZATION OF COMPRESSION UNIT IN
PARALLEL TREE MULTIPLIERS

SUSTAINABLE ENVIRONMENT AND ENERGY SYSTEMS

MIDDLE EAST TECHNICAL UNIVERSITY,

NORTHERN CYPRUS CAMPUS

BY

M. S. RASHID

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR

THE DEGREE OF MASTER OF SCIENCE

IN

SUSTAINABLE ENVIRONMENT AND ENERGY SYSTEMS PROGRAM

JANUARY 2018

Approval of the Board of Graduate Programs



Prof. Dr. Gürkan Karakaş
Chairperson

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.








Assist. Prof. Dr. Carter Mandrik
Program Coordinator

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.



Assoc. Prof. Dr. Ali Muhtaroglu
Thesis Supervisor

Examining Committee Members

Assoc. Prof. Dr. Fadi Al-Turjman	Computer Engineering, METU, NCC.	
Assoc. Prof. Dr. Ali Muhtaroglu	Electrical & Electronics Engineering, METU, NCC.	
Prof. Dr. Dr. Sadık Ülker	Electrical & Electronics, Engineering, EUL, TRNC.	
Assoc. Prof. Dr. Cüneyt F. Bazlamaççi	Electrical & Electronics, Engineering, METU Ankara.	
Assoc. Prof. Dr. Muhammad Salamah	Computer Engineering Department, EMU, TRNC.	

ETHICAL DECLARATION

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: M. S. Rashid

Signature: *M. S. Rashid*

Email: saleh.rashid@metu.edu.tr

saleh.uet@gmail.com

ABSTRACT

INVESTIGATION OF A METHOD TO IDENTIFY ENERGY EFFICIENT ORGANIZATION OF COMPRESSION UNIT IN PARALLEL TREE MULTIPLIERS

M. S. Rashid

M.Sc., Sustainable Environment, and Energy Systems

Supervisor: Assoc. Prof. Dr. Ali Muhtaroglu

January 2018, 120 pages

In this study, high-performance integer multipliers extensively used in digital signal processing are investigated in the context of energy-aware system organization. Among different functional blocks in an integer multiplier, the compression unit is primarily targeted for optimization as the largest section of the multiplier with significant energy consumption. Five different realizations of the most popular Wallace tree are investigated in this work, which consists of Dual Pass Logic (DPL) circuit implementations based on (3,2) counters, 4:2 compressors, 5:2 compressors, 6:2 compressors, and hybrid of (7,3) and (3,2) counters. Multiplier energy consumption highly depends on the type, efficiency, and a number of counters and compressors in the compression block, as well as connectivity and placement of these blocks. Therefore, the study investigates these aspects separately. Firstly, individual compressor and counter modules are implemented using Dual Pass Logic (DPL) circuit style in 180nm process technology to be available in a common library. A new metric is proposed for the energy efficiency of the individual compressor or counter modules and is quantified through circuit simulations. Each Wallace tree organization to be compared is designed in various sizes using the library. Theoretical worst delay path in each organization is verified through simulations using a large vector set. Then, a new method named System Compression Energy Efficiency Score (SCEES) is proposed, to quantitatively predict the ranking of different multiplier organizations in terms of energy efficiency. Extensive circuit simulations are performed to verify the effectiveness of the proposed method and metric, and the output data is post-processed using MATLAB. The analysis results in a refined energy aware methodology to select the most appropriate compression blocks to achieve energy efficiency for a particular multiplier size without going through the cumbersome steps in custom system design and simulations.

Keywords: Compression circuits, VLSI, energy, delay, integer multiplication.

ÖZ

PARALEL AĞAÇ ÇARPANLARINDA ENERJİ VERİMLİLİĞİ YÜKSEK SIKIŞTIRMA ÜNİTELERİNİN BELİRLENMESİNE YÖNELİK YÖNTEM GELİŞTİRİLMESİ

M.S.Rashid

Master, Sürdürülebilir Çevre ve Enerji Sistemleri

Tez Yöneticisi : Doç. Dr. Ali Muhtaroglu

Ocak 2018, 120 sayfa

Bu çalışmada, dijital sinyal işlemede yaygın olarak kullanılan yüksek performanslı tamsayı çarpanları, enerji farkındalıklı sistem organizasyonu bağlamında incelenmiştir. Bir tamsayı çarpanı içindeki farklı işlevsel bloklar arasında enerji harcaması en yüksek parça olan sıkıştırma birimi, optimizasyon için öncelikli olarak hedeflenir. En popüler Wallace ağacının (3,2) sayaçları, 4:2 sıkıştırıcıları, 5:2 sıkıştırıcıları, 6:2 sıkıştırıcıları ve (7,3) ve (3,2) sayaçlarının hibrit yapısını temel alan beş farklı uygulaması, Çift Geçiş Mantiği (DPL) devre tasarım tekniği kullanarak incelenmiştir. Çarpan enerji tüketimi, sıkıştırma ünitesindeki sayaçların ve sıkıştırıcıların tür, verimlilik ve sayısına, ayrıca bu blokların yerleşim ve bağlantısına göre değişir. Bu nedenle, çalışma bu faktörleri ayrı ayrı incelemiştir. Öncelikle, sıkıştırıcı ve sayaç modülleri 180nm transistör teknolojisinde Dual Pass Logic (DPL) devre stili kullanılarak tasarlanmış ve ortak bir kütüphanede toplanmıştır. Sıkıştırıcı ve sayaç enerji verimliliği için yeni bir metrik önerilmiş ve devre simülasyonları ile nicelleştirilmiştir. Karşılaştırılacak her Wallace ağacı organizasyonu ortak kütüphaneyi kullanarak çeşitli boyutlarda tasarlanmıştır. Her organizasyonda en kötü teorik gecikme yolu, büyük bir vektör seti kullanarak simülasyonlar yoluyla doğrulanmıştır. Ardından, farklı çarpan organizasyonlarının sıralamasını enerji verimliliği açısından nicel olarak tahmin etmek için Sistem Sıkıştırma Enerji Verimliliği Skoru (SCEES) adlı yeni bir metrik önerilmiştir. Önerilen yöntemin ve metriğin etkinliğini doğrulamak için kapsamlı devre simülasyonları yapılmış ve çıktı verileri MATLAB kullanılarak işlenmiştir. Analiz, özel sistem tasarımı ve simülasyonundaki zor ve uzun adımları atmadan belirli bir çarpan boyutu için enerji verimliliği yüksek en uygun sıkıştırma bloklarını seçmek için yeni bir yöntemi ortaya koymaktadır.

Anahtar Kelimeler: Sayıcı ve sıkıştırma devreleri, VLSI, güç, performans, gecikme, tamsayı çarpımı.

DEDICATION

To my Family

ACKNOWLEDGEMENTS

I would like to thank Dr. Ali Muhtaroglu for his great contribution in this work. It was really a pleasure for me to work with him. Also I am grateful to him for accepting me to work under his supervision at the time when I was novice in research.

I also like to thank Pradeep Jayaweera and Hamed Osoli for helping me with managing the Cadence environment. I would also like to acknowledge and thank Arsalan Tariq, Hassan Ali, Fahad Haneef, Hamayun Bhai, Bushra Fatima, Obaid Ullah and his wife, Tariq Rahim and Ali Hamza, Ahmed Rasheed, Harron ur Rasheed, Zakaria Qadir, Yashfeen Zahid and Sana Khan for their support

This study was supported by Middle East Technical University Northern Cyprus Campus, Scientific Research Project Grant No: FEN-14-D-5.

TABLE OF CONTENTS

ETHICAL DECLARATION	iii
ABSTRACT	v
ÖZ.....	vi
DEDICATION	vii
ACKNOWLEDGEMENTS	viii
TABLE OF CONTENTS	ix
TABLE OF FIGURES	xii
LIST OF TABLES	xv
GLOSSARY	xvi
1. CHAPTER 1: INTRODUCTION	1
1.1 GREEN COMPUTING	1
1.1.1 PROCESSOR THERMAL EFFECTS	3
1.2 CHOICE OF DESIGN PARAMETERS	4
1.2.1 STATIC OR DYNAMIC LOGIC	6
1.2.2 PASS GATE AND COMPLEMENTARY STATIC LOGIC STYLES	7
1.2.3 VOLTAGE	8
1.2.4 ASPECT RATIO	9
1.2.5 PIPELINED OR SINGLE CYCLE MACHINE	9
1.2.6 COMBINATIONAL CIRCUITS OR LUT BASED MULTIPLIER	9
1.3 LEVELS OF ENERGY OPTIMIZATION	10
1.3.1 ARCHITECTURE LEVEL OPTIMIZATION	11
1.3.2 MULTIPLICATION IN GENERAL	11
1.4 TARGETED ENERGY EFFICIENT VLSI MULTIPLIER ARCHITECTURES	12
1.5 ORGANIZATION OF CURRENT STUDY	14
2. CHAPTER 2: BACKGROUND RESEARCH	15
2.1 MULTIPLIERS	15
2.1.1 LITERATURE REVIEW	17

2.1.2	COMPONENTS OF AN INTEGER MULTIPLIER.....	23
2.1.3	BOOTH ENCODING.....	23
2.1.4	COMPRESSION STAGE.....	24
2.1.5	FINAL ADDITION	25
2.2	COUNTER AND COMPRESSOR CIRCUITS.....	28
2.2.1	DEFINITIONS.....	29
2.2.2	SATURATED SCI COUNTERS	31
2.2.3	SATURATED COMPRESSORS	39
2.3	SPECIAL COMPRESSION CIRCUITS	43
2.4	CHAPTER SUMMARY	44
3.	CHAPTER 3: MULTIPLIERS UNDER STUDY.....	45
3.1	WALLACE MULTIPLIER.....	45
3.1.1	WALLACE MULTIPLIER BASED ON (3,2) COUNTERS.....	45
3.1.2	WALLACE MULTIPLIERS BASED ON COMPRESSORS	50
3.2	HYBRID WALLACE MULTIPLIER	55
3.3	ABACUS MULTIPLIER.....	57
3.4	CONFIGURATION OF MULTIPLIER ARCHITECTURES	60
3.5	CHAPTER SUMMARY	63
4.	CHAPTER 4: ANALYSIS OF COMPRESSION BLOCKS FOR ENERGY AWARE MULTIPLIERS.....	66
4.1	INTRODUCTION.....	66
4.2	CHARACTERIZING COMPRESSION CIRCUITS	68
4.3	THEORETICAL COMPARISON OF COMPRESSION CIRCUITS.....	70
4.4	SIMULATION COMPARISON OF COMPRESSION CIRCUITS	74
4.4.1	EXISTING AND PROPOSED METRIC FOR EVALUATION OF COMPRESSION CIRCUITS	74
4.5	CHAPTER SUMMARY	76
5.	CHAPTER 5: ANALYSIS OF MULTIPLIERS FOR ENERGY AWARE APPLICATIONS	79
5.1	EVALUATION METHOD OF MULTIPLIER ARCHITECTURES	79

5.2	WORST DELAY IN MULTIPLIERS	81
5.2.1	WORST DELAY ANALYSIS IN LITERATURE.....	82
5.2.2	ESTIMATED WORST CASE.....	83
5.3	LEAKAGE AND POWER ESTIMATION	85
5.4	SIMULATION RESULTS.....	86
5.4.1	8x8 MULTIPLIER	86
5.4.2	10x10 MULTIPLIER	87
5.4.3	12x12 MULTIPLIER	89
5.4.4	16x16 MULTIPLIER	89
5.5	COMPRESSION EFFICIENCY SCORE OF MULTIPLIERS	92
5.6	CHAPTER SUMMARY	95
6.	CHAPTER 6: CONCLUSION AND FUTURE WORK.....	99
6.1	SUMMARY AND CONCLUSION.....	99
6.2	FUTURE WORK	102
6.2.1	OPTIMIZATION OF ABACUS ARCHITECTURE	102
6.2.2	INACCURATE ABACUS.....	106
6.2.3	MINIMIZING SIMULATION TIME FOR LARGE MULTIPLIERS...	107
6.2.4	MODELLING OF POWER AND DELAY	109
6.2.5	CIRCUIT LEVEL OPTIMIZATION: GATE RESIZING.....	109
7.	REFERENCES	110
8.	APPENDIX A.....	118
9.	LIST OF PUBLICATIONS	124

TABLE OF FIGURES

Figure 1.1. The growth rate of smartphone and PC users	3
Figure 1.2. Power Management software by NSF [8].	5
Figure 1.3. Dynamic Logic (left) and Static Logic (right).....	6
Figure 1.4. Two-input multiplexer with static logic design styles: Conventional CMOS (left), CMOS with pass gates (middle), Dual Pass Logic (right) [12].	7
Figure 1.5. Domains of energy optimization in computer architecture [5].	10
Figure 1.6. Power Density (left) [28] and temperature Map for single core processor (right). ...	11
Figure 1.7. Typical M x N multiplication [24].	12
Figure 1.8. Leakage power projected w.r.t die size by ITRS (http://public.itrs.net).....	13
Figure 2.1. Classification of Multipliers	16
Figure 2.2. Parallel Multiplication flow.....	23
Figure 2.3. Typical structure of RCA	26
Figure 2.4. Mux gate for implementation of CLA carry.....	28
Figure 2.5. Compressor (left) and Counter (right) block	29
Figure 2.6. Classification of counter and compressor circuits	31
Figure 2.7. (3,2) the counter circuit or full Adder [15]	32
Figure 2.8. (7,3) counter circuit based on full adder [51]	32
Figure 2.9. (15,4) counter circuit based on full Adder [109]	33
Figure 2.10. (2,3,3) counter circuit [114].....	34
Figure 2.11. (5,5,4) counter symbol (left) and circuit (right) [51]	34
Figure 2.12. (2,2) counter (right), symbol (left).....	35
Figure 2.13. Un-Saturated Counters	38
Figure 2.14. 4:2 compressor based on (3,2) counter [51] (bottom) and its symbol (top)	39
Figure 2.15. Optimized 4:2 compressor(right) [15], symbol used in multiplier (left)	40
Figure 2.16. 5:2 compressor [51].....	41
Figure 2.17. Optimized 5:2 compressor [124]	41
Figure 2.18. 6:2 compressors	42
Figure 2.19. Optimized 6:2 compressor.....	42
Figure 2.20. Optimized 6:2 compressor [51]	42

Figure 3.1. 8x8 Wallace Multiplier based on (3,2) counters [128]	47
Figure 3.2. 10x10 Wallace Multiplier based on (3,2) counters	48
Figure 3.3. 12x12 Wallace Multiplier based on (3,2) counters [30].	49
Figure 3.4. 16x16 Wallace Multiplier based on (3,2) counters [43]	49
Figure 3.5. 8x8 Wallace Multiplier based on 4:2 compressors [131]	50
Figure 3.6. 10x10 Wallace Multiplier based on 4:2 compressors	51
Figure 3.7. 12x12 Wallace Multiplier based on 4:2 compressors	51
Figure 3.8. 16x16 Wallace Multiplier based on 4:2 compressors	51
Figure 3.9. 8x8 Wallace Multiplier based on 5:2 compressors	52
Figure 3.10. 10x10 Wallace Multiplier based on 5:2 compressors	52
Figure 3.11. 12x12 Wallace Multiplier based on 5:2 compressors	53
Figure 3.12. 16x16 Wallace Multiplier based on 5:2 compressors	53
Figure 3.13. 8x8 Wallace Multiplier based on 6:2 compressors	53
Figure 3.14. 10x10 Wallace Multiplier based on 6:2 compressors	54
Figure 3.15. 12x12 Wallace Multiplier based on 6:2 compressors	54
Figure 3.16. 16x16 Wallace Multiplier based on 6:2 compressors	54
Figure 3.17. 8x8 hybrid Wallace Multiplier [43]	55
Figure 3.18. 10x10 hybrid Wallace Multiplier	56
Figure 3.19. 12x12 hybrid Wallace Multiplier	56
Figure 3.20. 16x16 hybrid Wallace Multiplier [43]	57
Figure 3.21. 8x8 ABACUS Multiplier [24]	58
Figure 3.22. 10x10 ABACUS Multiplier	59
Figure 3.23. 12x12 ABACUS Multiplier	59
Figure 3.24. 16x16 ABACUS Multiplier	59
Figure 3.25. Trends of logic blocks for different sizes of multipliers	65
Figure 4.1. AND implementation in DPL (left) and symbol (right).	67
Figure 4.2. MUX implementation in DPL: symbol (left) and gate level implementation (right)	67
Figure 4.3. XOR implementation in DPL: symbol (left) and gate level implementation (right).	67
Figure 4.4. 'C _R ' and 'c _R ' of counter and compressor circuits	73
Figure 4.5. Compression ratio and Normalized compression ratio w.r.t worst gate delays and total number of gate delays, for saturated counters	73

Figure 4.6. Test bench for compression circuit.....	74
Figure 4.7. C_R and normalized C_R w.r.t simulated PDP of single column input saturated and unsaturated counters.....	76
Figure 4.8. XOR/MUX gates and AND-gates in single column input counters.....	77
Figure 4.9. CR and normalized CR w.r.t simulated PDP of single column input saturated and unsaturated counters.....	77
Figure 5.1. Test bench for multipliers.....	80
Figure 5.2. Possible worst-case path only in compression stage	83
Figure 5.3. Worst case delay input vectors, 8x8 bits multiplier as an example	84
Figure 5.4. A sampling of input current when $\alpha = 0$	86
Figure 5.5 Comparison of PDP and SCEES for all given sizes	96
Figure 5.6. PDP and EDP in 8-bit multiplier	96
Figure 5.7 PDP and EDP in 10-bit multiplier	97
Figure 5.8. PDP and EDP in 12-bit multiplier	97
Figure 5.9. PDP and EDP in 16-bit multiplier	98
Figure 5.10. Trends in PDP for different multipliers	98
Figure 6.1. Proposed 8x8 Hybrid ABACUS Multiplier (Red rectangles depict 4:2 compressor circuits).....	103
Figure 6.2. Proposed 10x10 Hybrid ABACUS Multiplier.....	103
Figure 6.3. Proposed 12x12 Hybrid ABACUS Multiplier.....	104
Figure 6.4. Proposed 16x16 Hybrid ABACUS Multiplier_Version-I.....	104
Figure 6.5. Proposed 16x16 Hybrid ABACUS Multiplier Version-II	105
Figure 6.6. Proposed 16x16 Hybrid ABACUS Multiplier Version-III.....	105
Figure 6.7. Trends of number of stages of Multiplier Architecture	107

LIST OF TABLES

Table 1. Literature review of Multiplication Architectures	18
Table 2. Configuration of 8x8 multiplier	60
Table 3. Configuration of 10x10 multiplier	61
Table 4. Configuration of 12x12 multiplier	62
Table 5. Configuration of 16x16 multiplier	63
Table 6. Abbreviation of multiplier architecture.....	64
Table 7. Configuration of counter and compressors with their ‘C _R ’ and ‘c _R ’	71
Table 8. Correction and characterization of compression circuits with new metric: C _R /PDP	78
Table 9. Evaluation Criteria for Multipliers in Literature	81
Table 10. Worst case distribution in multiplier architecture	84
Table 11. Simulation results for 8-bit multiplier architectures	87
Table 12. Contribution of each block in worst case for 8-bit multipliers	87
Table 13. Simulation results for 10-bit multiplier architectures	87
Table 14. Contribution of each block in worst case for 10-bit multipliers Pico seconds (ps)	88
Table 15. Simulation results for 12-bit multiplier architectures	89
Table 16. Contribution of each block in worst case for 12-bit multipliers Pico seconds (ps)	89
Table 17. Simulation results for 16-bit multiplier architectures	90
Table 18. Contribution of each block in worst case for 16-bit.....	90
Table 19. Fan-out of counter and compressor blocks	94
Table 20. Multiplier Configuration, PDP, and SCEES	94

GLOSSARY

ALU:	Arithmetic Logic Unit
CPU:	Central Processing Unit
ARC1:	(3,2) counter based Wallace Multiplier
ARC2:	4:2 compressor based Wallace Multiplier
ARC3:	5:2 compressor based Wallace Multiplier
ARC4:	6:2 compressor based Wallace Multiplier
ARC5:	Hybrid Wallace Multiplier
ARC6:	ABACUS Multiplier
C_{dyn} :	Dynamic Capacitance
C_{Stat} :	Static Capacitance
CLK:	Clock
CMOS:	Complementary Metal-Oxide Semiconductor
C_{out} :	Carry-Out
C_{in} :	Carry-In
CLA:	Carry Look-ahead Adder
CISC:	Complex instruction set computing
CSA:	Carry-Save Adder
CPL:	Complementary Pass Transistor Logic
DFT:	Design-for-Test
DSP:	Digital Signal Processor
DTL:	Differential Threshold Logic
DPL	Dual Pass Logic
ES:	Error Significance
ER:	Error Distance
EEMPC	Embedded Microprocessor Benchmark Consortium
EPA	Environmental Protection Agency
EDM:	Elmore Delay Model
EDP:	Energy-Delay Product
f:	Femto (10e-15)
fJ	Femto Joule
F_{out}	Output node of circuits
f_{opt}	Operating Frequency
FA:	Full Adder
FPGA:	Field Programmer gate array
GPU:	Graphical Processing Units

HA:	Half Adder
ICT	Information and Communication Technology
IC:	Integrated Circuit
I_{stat} :	Static Current
IoT:	Internet of Things
ITRS:	International Technology Roadmap for Semiconductors
J:	Joule
LSB:	Least significant bit
LUT:	Look-up Table
MUX:	Multiplexer
MAC:	Multiplication and Accumulator
MSB:	Most Significant Bit
MCI:	Multiple Column Input
MED:	Mean Error Distance
n:	Number of Inputs/Nano ($10e-9$) (Depending on Use)
N:	Overall Number of Stages
NSF:	National Science Foundation (non-full swing block)
p:	Pico ($10e-12$)
P:	Power
P_{av} :	Total Average Power
PCB:	Printed Circuit Board
PDP:	Power-Delay Product
PDA:	Power Delay Area
PTL	Pass Transistor Logic
P_{dyn} :	Dynamic Power
PPG:	Partial Product Generator
P_{short} :	Short Circuit Power
P_{stat} :	Static Power
RCA:	Ripple Carry Adder
RISC:	Reduced instruction set computing
SCI:	Single column input
SC:	Short Circuit
SWaP:	Space, Wattage, and Performance
SPEC:	Standard Performance Evaluation Corporation
SCEES:	System Compression Energy Efficiency Score
SNR:	Signal to Noise
TG:	Transmission gate

TTL:	Transistor, Transistor Logic
V_{th} :	Threshold Voltage
V_{DD}	Supply/Source Voltage
VLSI	Very Large Scale Integration
WGD:	Worst Gate Delay
WTM:	Wallace Tree Multiplier
XOR:	Exclusive-OR
yJs	Yecto ($10e-24$) Joule second
α :	Activity Factor
μ :	Micro ($10e-6$)

CHAPTER 1

INTRODUCTION

Though high-speed computation is important for electronic system performance, it is also an integral component of systems with long battery life. Portability, on the other hand, restricts system size and inhibits the use of bulky batteries. Green computing is another trend that drives the electronic design nowadays, caused by the desire to reduce greenhouse gas emissions due to fossil fuel consumption. Thus, efficient algorithms are required for electronics to use limited energy efficiently. This chapter provides an overview of the efforts toward green computing, and associated power consumption issues. Subsequently, sustainable practices are reviewed to position the significance of parallel multipliers in the sphere of green computing. Next, the available choices for multiplier VLSI circuits and motivations for focusing the work on a particular portion of this domain are covered. Finally, the organization of this study concludes the chapter.

1.1 GREEN COMPUTING

Power efficient computing dates back to 1991 when the environmental protection agency (EPA) introduced the first programme named '*Sustainable light*'. Later in 1992, '*Energy star program*' on energy specification for monitors and computers magnified the notion of green computing. More recently, technical innovations around cloud computing, growth of power density in processors and related new cooling requirements in both devices or data centers, and cost/kWh, and new environmentally motivated regulations to clamp down the power consumption of electronics have prioritized sustainability in the global Information and Communications Technology (ICT) industry. Data centers, for instance, represents a clear example of energy demanding ICT. Google data centers alone, including the one in Council Bluffs, Iowa, which spans to 115,000 ft², generates half of the energy expenditure and carbon footprint of Google

[1], and consume 4,402,836 MWh in 2014 [2]. Six thousand data centers in the US consume 6 BkWh of electricity, which cost 4.5 Billion USD in 2006 and has had a drastic growth rate of 12% per year [3].

The digital economy overall consumes a tenth of the world's electricity i.e. 1500 TWh of power/year [4], in addition, computers in the US alone consume 20 GWh per year only [5]. ICT companies are driven towards sustainable practices fueled either by internal factors such as cost, trademark, or external factors such as governmental laws, awareness of green products and environmental concern of power consumption from traditional sources. A lot of effort has been spent so far to support sustainable green computing. Such efforts include the 'Energy Star' programmes, 'SPECPower', 'JouleSort' and 'Carbon-Footprint' standards. Intel's 'Thermal Design Power' or TDP and AMD's 'Average CPU power' or ACP for processors, 3DMark2006 Score/Watts for graphics processing units (GPU), Space, wattage and performance (SWaP), Embedded Microprocessor Benchmark Consortium (EEMBC) are other examples. The agencies and institutes listed below are working in different parts of the world to put into force the laws of energy efficiency in the sector:

- US: (<http://www.epa.gov/>)
- EU:(<https://ec.europa.eu/easme/en/intelligent-energy-europe>)
- UK: (<http://efficient-products.defra.gov.uk/cms/market-transformation-programme/>)
- Japan:(http://www.eccj.or.jp/top_runner/index.html)
- World-wide: The Standard Performance Evaluation Corporation (**SPEC**), (<http://www.spec.org/>)
- Green Grid Association specifically for green ICT, members include Cisco, Digital Reality, Intel, Schneider Electric (<http://www.thegreengrid.org/>)
- TPC-Energy, specifically for Datacenters. (http://www.tpc.org/tpc_energy/default.asp)

Rebound effect, which exists in other technological spheres, is essentially absent in ICT. For example, fuel limits the excessive use of cars, but in the computer, there is virtually no layover for users [6]. Energy consumed by communication gadgets is quite small, but their number is increasing exponentially i.e. smartphone users increased from 42.8 million in 2008 to nearly 1 billion in 2014 [6], as depicted in Figure 1.1. If performance per watt-hour remains constant in the next few years, energy costs will be significantly more than hardware costs in digital gadgets [7], and it is just the beginning to endorse the impact of ICT on the global economy, electricity consumption, and carbon footprint.

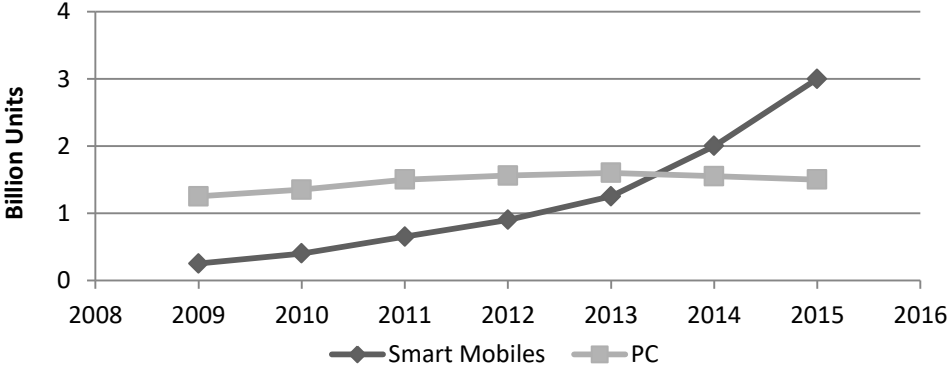


Figure 1.1. The growth rate of smartphone and PC users

Along with metrics discuss before, National Science Foundation (NSF) encourage component-level power efficiency and power management of microchips, which significantly reduce climatic impact and bring energy saving to users [8]. Energy efficient microchips and processors are also significant because of their reliability and durability, whereas both reliability and durability are associated with the heat generated by consumed power.

1.1.1 PROCESSOR THERMAL EFFECTS

IC and processor industry innovations are considerably faster than other technologies. In a comparison to Intel’s C4004 processor released in 1971 with the state of the art 14nm technology processor; performance is now 3500 times higher. Energy efficiency is improved to 90,000 times and cost per transistor falls up to 60,000 times. If automotive

technology had improved with the same rate, then cars would have gone up to 300 thousand mph, 2 million miles per gallon with 4 cents only (www.intel.com). Improvement in technology decreases per device power consumption but increases power density, which in return cause high temperatures in processors. Reliability can be ensured by maintaining the optimum temperature. However, heat does not drain in processors core properly to a regrettable extent. and heat exhaust consumption sometimes takes over processor's consumption. Not only those high cooling costs endured, extensive heat also causes chips to fracture. An *Uptime Institute* study, a preeminent global authority on data center standards (www.uptimeinstitute.com), suggests that for every 10°C increase in temperature, the probability of a fault in data centers increase up to 50%, while the same fault percentage applies to computers with the 20°C increase in temperature. Energy inefficient processors restrict the effective use of cores and result in short lifespan, therefore, *Apple* in 2006 switched to cooler *Intel* chips in 'Power Mac G5' from *IBM* 'PowerPC' chips, (www.technologyreview.com). *Intel* on the other hand, in 2004 also publicly declared its 'Thermal wall' on its microprocessor (www.intel.com). Also because of heat issues in processors, *Intel* scrapped its multimillion projects: Tejas and Jayhawk processors, so as to meet customer requirement of cooler and efficient chips [9]. Heat has become a basic design consideration in modern day computers, therefore, processor companies introduce dedicated power management chips in systems. Other quick fixes are third-party software i.e. 'speed-step' by *Intel* and 'PowerNow' by *AMD*. NSF funded company *Miserware* delivers similar software, 'Granola' (Figure 1.2) which can reduce power usage by 2%–18% [8]. Optimum temperature with high performance requires efficient energy-aware architecture, which is only possible by intelligent selection of design parameters.

1.2 CHOICE OF DESIGN PARAMETERS

Evaluation of arithmetic circuits is possible through FPGA synthesis or customized circuit design simulations. In the latter case, circuits can be tested directly after fabrication to analyze real-world behavior for example over voltage and temperature.

Fabrication of integrated circuits typically take several months; we have therefore chosen customized design simulation runs (without going through fabrication) to effectively compare multiplication architectures and compression circuits. Customized circuits provide better optimization of all performance metrics compared to programmable logic such as FPGAs since FPGAs have plenty of redundant circuits in return for providing much shorter design and test cycle. Well-known performance metrics of VLSI circuits include area, cost, power, delay, power delay product (PDP), energy-delay product (EDP), throughput, and uncommon but important: power delay area (PDA). It is nearly impossible to optimize for one design constraint without changing other factors. Each metric has its own advantages and disadvantages. For instance, **architectures** can be either asynchronous or synchronously clocked, RISC or CISC, pipelined or a single cycle. **Circuits** can conversely be realized by either static or dynamic logic, pass-gate or complementary CMOS style. A simple method discussed by [10] to select an architecture and circuit topology is to reduce V_{DD} to meet the throughput requirement with minimum power dissipation, and implement low power circuits. The detailed analysis of low power circuits is out of the scope of this study, but few design choices are discussed next.



Figure 1.2. Power Management software by NSF [8].

1.2.1 STATIC OR DYNAMIC LOGIC

Dynamic logic requires clock pulse with cyclic operation of pre-charge and evaluate phases (Figure 1.3). This significantly reduces the number of devices compared to static CMOS from ‘ $2N$ ’ to ‘ $N+2$ ’, for N -input logic function. In addition, dynamic logic has smaller short-circuited current (if clock skew is not significant), dynamic current (as a lesser number of devices being used or switched) and delay. Also, it has a smaller percentage of glitches compared to static CMOS in which glitches are responsible for 20% - 70% of total power dissipation [11]. Glitches or false transitions of a logic level in both static and dynamic logic depends on the local interconnects the skew delays, and depth of critical paths.

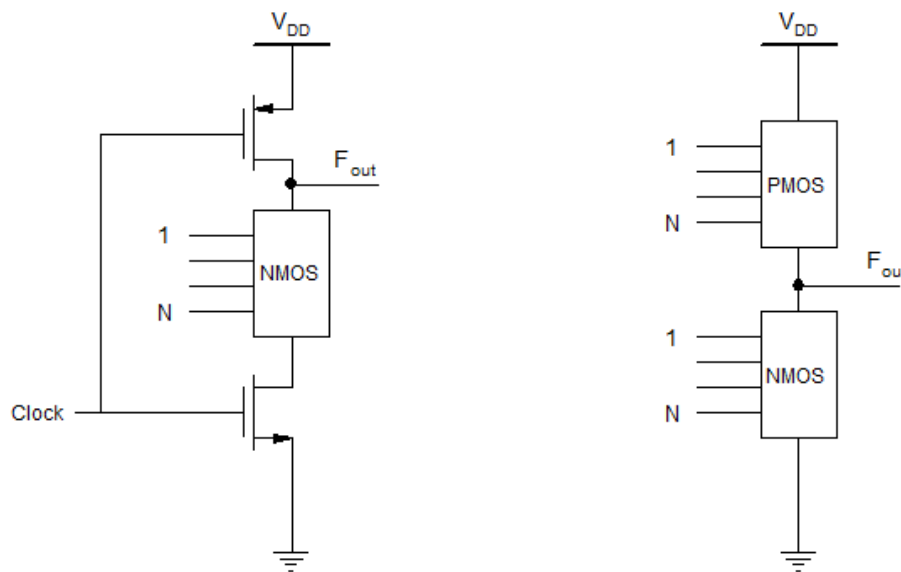


Figure 1.3. Dynamic Logic (left) and Static Logic (right)

On the other hand, the dynamic logic circuit requires a clock tree, distributed all over the circuit and pre-charge for every node (even if it discharges again in evaluate phase) which drastically increases the power dissipation. Static CMOS circuit gives better results for low power logic as it does not require a clock tree and devices to maintain the charge at ‘ F_{out} ’ node (Figure 1.3). We have therefore chosen static logic to prevent complexities and power overhead of clock tree.

1.2.2 PASS GATE AND COMPLEMENTARY STATIC LOGIC STYLES

Logic style strongly influences delay, size, power dissipation and wiring complexity. Ideally, logic realization methodology must be; easy to scale-up, robust and compatible [12]. Each logic style has its own advantages and disadvantages, as demonstrated by two-input multiplexer example in Figure 1.4. Conventional CMOS on left (Figure 1.4) has a large number of MOSFETs which gives full swing output, while pass gates at the middle of Figure 1.4 provide energy efficient solution with degradation of signal quality. Dual pass logic based circuit, on right in Figure 1.4, yields both implementation benefits i.e. the low power of pass logic and full swing of complementary logic.

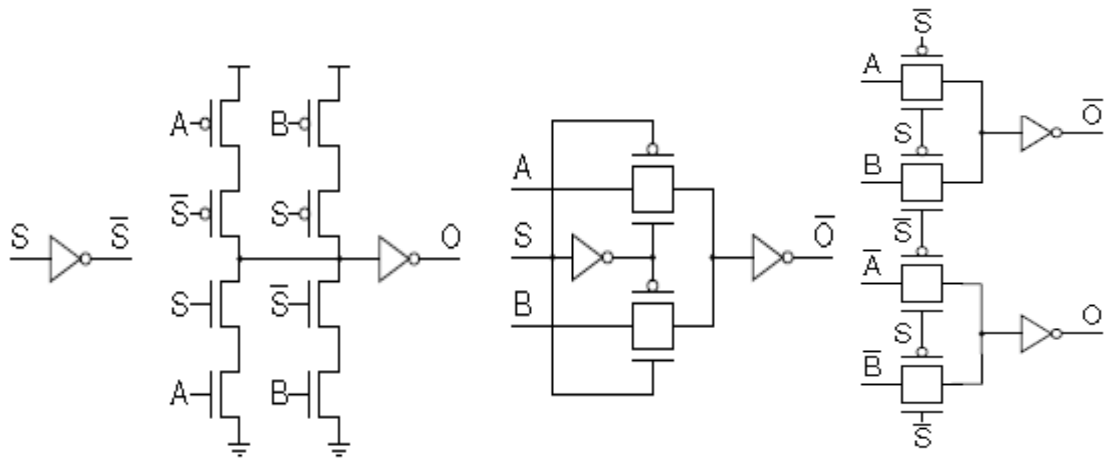


Figure 1.4. Two-input multiplexer with static logic design styles: Conventional CMOS (left), CMOS with pass gates (middle), Dual Pass Logic (right) [12].

Multiplication architectures exist in literature and have been realized by diverse logic styles for example; [13] used pseud-NMOS, [14] used complementary pass logic or CPL, [15] used DPL, [16] used domino and pass transistor logic, [17] used transmission gate based circuits and [18] used static CMOS. Rather few logic styles have been proposed by using multiplication circuits as vehicle i.e. CPL [14], while some compared different logic styles using multipliers i.e. [19]. Rather multiplier's silicon-based results often used as benchmarks for signifying high-speed technologies, for instance, si-bipolar, Josephson junction devices and GaAs [14].

Complementary CMOS logic style has pull-up and pull-down networks which result in full swing output voltage and robust to voltage scaling. Also, regularity, less diffuse capacitance and dynamic power, the generality of an input function and ease in modeling make it suitable for almost all functions except complex gates like XOR and multiplexers [20]. Authors [21] demonstrated that although pass logic family is sensitive to voltage scaling, it is better for complex gates like XOR used in adders and multiplication. Therefore, DPL introduced by M. Suzuki et al. [22] has been used in this study for circuit realization, among other pass logic styles, as it gives full swing output and has small PDP compared to other i.e. CPL, DPL, and C²PL etc. Inverted signals in DPL structure do not require inverters as in PTL, although they have been used only to restore signals at outputs of each compression blocks. In addition, complementary CMOS is not area efficient, because of bulky PMOSs. A more detailed discussion on circuit implementation will be presented later in Chapter 4, before presenting simulations testbench.

1.2.3 VOLTAGE

The optimal voltage level is crucial in VLSI circuits since it effects by order of magnitude on power and power-delay product, whereas delay is inversely proportional to voltage [10] as illustrated by (1.2) to (1.4):

$$P_{\text{dyn}} = \alpha (C_{\text{dyn}} \cdot F_{\text{opt}} \cdot V_{\text{DD}}^2) \quad (1.1)$$

$$\text{Power delay product} \propto V_{\text{DD}}^2 \quad (1.2)$$

$$\text{delay} \propto \frac{1}{V_{\text{DD}}} \quad (1.3)$$

There is a trade-off in speed and reliability by scaling voltage. High voltages can decrease latency but cause a hot spot in processors due to excessive heat and energy. In DPL, voltage scaling is inversely related to a number of restoring inverters. Restoring inverters deployed at every output of compression circuits to prevent major design flaw when $V_{\text{DD}} \approx$ threshold drop in the worst path. Authors in [23] performed detailed

analysis on voltage scaling w.r.t frequency and found that ultralow voltage (less than 1 V) can be achieved for the maximum rated frequency of 100MHz in 180nm technology. However, the conclusion is based on single small adder circuit, which cannot be generalized to large multiplication circuits; therefore, we have used standard voltage of 1.8 V for 180nm technology in our circuits.

1.2.4 ASPECT RATIO

Gate sizing or variation of W/L (aspect) ratio is another key parameter to optimize power in VLSI circuits. Optimization by tuning aspect ratio [20] is adequate for small compression units, but for multiplication architectures, aspect ratio optimization requires detailed analysis and automation which is out of the scope for this study. In 180nm technology, for equal rise and fall time, the aspect ratio of 2.5/1 has been used, calculated by a previous study [24] of our group on integer multipliers.

1.2.5 PIPELINED OR SINGLE CYCLE MACHINE

Classic single cycle multipliers can be pipelined, to increase operating frequency, by isolating compression and final addition into two cycles (commonly known as two-cycle multipliers). This approach is out of the scope of this study and perhaps inefficient because of imbalance delay distribution of multipliers [25]. The pipeline of each stage of compression affects the regularity of pipelining [13], as worst delay varies w.r.t. the depth of multiplier. Pipelining also makes hardware implementation more complex, therefore, single latency parallel multipliers have been used which typically preferred for arithmetic and logic unit (ALU) in processors.

1.2.6 COMBINATIONAL CIRCUITS OR LUT BASED MULTIPLIER

Parallel bits multipliers can realize by either combinational circuits or Look-Up Tables (LUT). VLSI combination circuits based implemented are more realistic in nature since it gives real-time analyses of arithmetic operation, whereas FPGA LUT based implementation shows constant delay for certain complex combinations. The overlap of two i.e. LUT based implementation of the multiplier in VLSI [26] is not that effective

since it requires an enormously on-die area for LUT. Also, it decreases the reliability of arithmetic operation [26]. Therefore, in this study, traditional combinational gate based multipliers have been implemented.

1.3 LEVELS OF ENERGY OPTIMIZATION

Energy can be optimized in hardware systems or through software, in the design phase or during operation. Methods of power optimization vary in different domains of ICT i.e. circuit, architecture, and systems. Energy efficiency was traditionally termed as system level optimization, but research is now shifting towards the algorithms and circuits. Optimization methodologies for the three major domains; circuits, architecture, and system, are well documented by [5] and summarized in Figure 1.5. The method in each domain also varies with respect to the metrics used to evaluate [5]. This study, in fact, is the overlap of architecture and circuits domains since it focuses on energy efficient multiplication architectures and compression circuits in particulars.

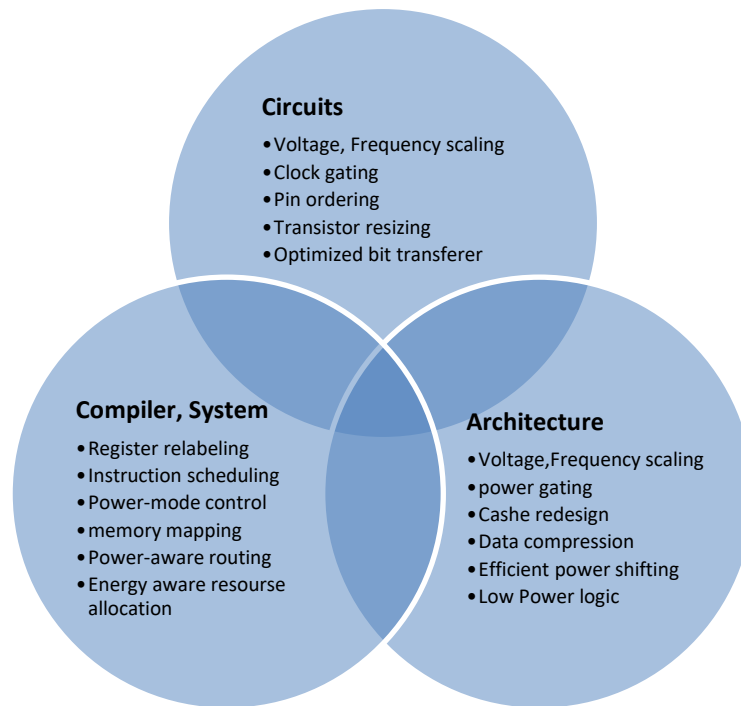


Figure 1.5. Domains of energy optimization in computer architecture [5].

1.3.1 ARCHITECTURE LEVEL OPTIMIZATION

This work essentially intended to investigate parallel bits multiplication architectures and compression circuits. Parallel bits binary multipliers are extensively used in modern 3D graphic engines, pixel-processing shades, and virtually all signal-processing applications. Efficient architectures of multipliers can also help to considerably reduce energy consumption in computing environs [27].

Heat concerns mentioned before are strongly influenced by the multiplication architectures in arithmetic and logic unit (ALU) in processors. Hot spots in the heat map of single core processor, as depicted in Figure 1.6 (right), can be observed particularly on ALU. Additionally given in [28], the power density map of Pentium® III as depicted in Figure 1.6 (left), also shows the power peaks near ALU. Generally, in ALU and particularly in graphical processing units (GPU), the most time and power consuming process is multiplication, as enormous energy is required to compress and present a vast amount of data.

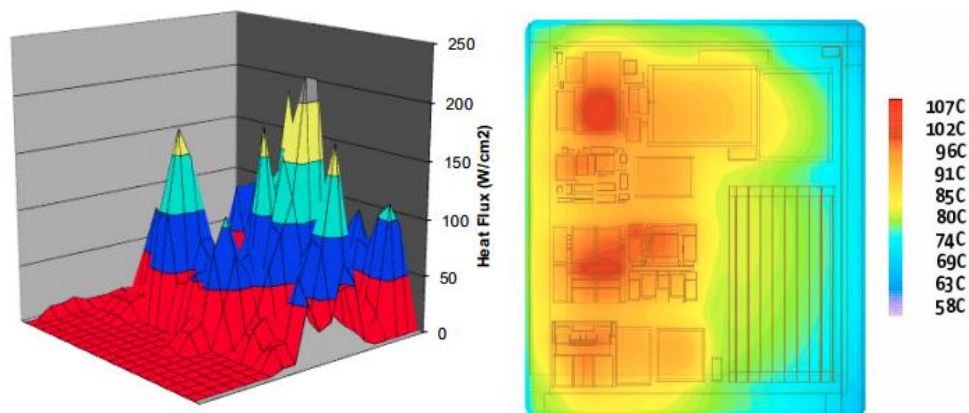


Figure 1.6. Power Density (left) [28] and temperature Map for single core processor (right).

1.3.2 MULTIPLICATION IN GENERAL

A simple $M \times N$ parallel bit multiplier as depicted in Figure 1.7 consists of M -bits for *multiplier* and N -bits for *multiplicand* and $M \times N$ bits of *partial products* and $N+M$ -bits

respectively. SC is governed by partially ON/OFF devices, which can be avoided by having full swing voltages at gates. SC power component can be tune to zero by resizing MOSFETs for the homogenous rise and fall timing. SC component is not critical for ultralow applications whereas, for 0.8V or above, SC power is expected to be 10~30% of overall power [10]. Lastly, the **static component** or leakage power, which is essentially controlled by the technology, can be minimized by power gating circuitry. A back-of-the-envelope calculation of static power is hard, but it largely depends on substrate injection and subthreshold effects. Leakage was not a major problem for older technologies i.e. 180nm or above, but with ever-shrinking MOSFET gate size, it perhaps dominates dynamic powers [10]. Percentage of **Leakage power** to the total power, according to the International Technology Roadmap for Semiconductors (www.itrs2.net) report, increases linearly with shrinking technology (**Error! Reference source not found.**). Leakage current is more significant than dynamic power in some application like sensor networks or more recent Internet of Things (IoT). As in IoT applications, numerous sensors nodes have been deployed and only a few of them occasionally communicates.

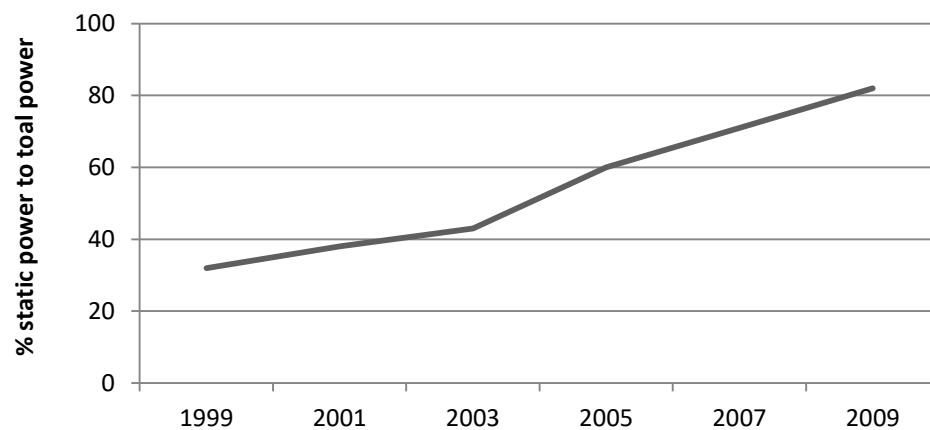


Figure 1.8. Leakage power projected w.r.t die size by ITRS (<http://public.itrs.net>.)

In conclusion, the low power VLSI circuit design is possible by optimizing: the number of transistors to reduce switched capacitance, having better architectures for smaller activity, improved layout methodologies for optimized capacitance, using small

size transistor for minimum node capacitance especially in critical path and by scaling down the voltage and frequency [12].

1.5 ORGANIZATION OF CURRENT STUDY

Either efficient architectures or optimized compression blocks can obtain energy-efficient multipliers. This study investigates both the individual compression blocks and multiplication architectures. Existing literature on multipliers and compression circuits is thoroughly discussed in Chapter 2. In Chapter 3, different multiplication architectures under study have been discussed in detail, which includes conventional Wallace tree based on (3,2) counter, 4:2 compressor, 5:2 compressor, 6:2 compressor and hybrid Wallace and ABACUS. The recently proposed hybrid Wallace is essentially composed of (7,3) and (2,3,3) counters. In this chapter each afore mentioned multiplier configuration have been realized in four different sizes (8-bit, 10-bit, 12-bit and 16-bit) and compared w.r.t the number of logic gates, the total number of MOSFETs and size of the final adder.

In Chapter 4, counters and compressors circuits discussed in Chapter 3 have been compared through exhaustive simulation runs. A new metric to evaluate compression circuits has been proposed in this chapter and compared with existing metric. Revisions of new evaluation metric have been summarized in the form of a table in conclusion section of this chapter. In Chapter 5, simulations are extended to multiplier architectures. A comprehensive discussion on simulation results and trends on worst delay, average current, PDP and EDP have been presented with theoretical back-ups. A new systematic evaluation method of multipliers has been proposed to observe the most efficient implementation for a particular size of the multiplier.

Chapter 6 summarizes this thesis study with concluding remarks and the future directions based on observations in Chapter 5. Appendix A contains signals output profile of multiplier architectures, which depicts the activity in compression stage.

CHAPTER 2

BACKGROUND RESEARCH

In Section 2.1 of this chapter, multipliers have been discussed in general and parallel bit binary multipliers in particular. A detailed comprehensive table summarizes the selected articles on parallel bit multipliers. Subsequently, three major components of the multiplier as mentioned in Chapter 1 (i.e. booth encoding, compression stage, and final addition) have been discussed in detail. Lastly, final adders exist in literature, and implementations of CLA and RCA have been demonstrated. Section 2.2 is dedicated to compression circuits used in compression stage of parallel bit binary multipliers. Each counter and compressors used in this study has been included in this section with a widespread discussion on their circuit realization. Compression ratio, worst-case path of every output interpreted through graphs and characteristic equations have been presented for extensive comparison of each counter and compressor circuits.

2.1 MULTIPLIERS

ALU is a key factor that limits the performance of modern processors. Inside ALU, multiplication is the most energy consuming process that drastically affect the performance [33]. In addition, the multipliers cover the largest silicon area of ALU in data paths [34]. The old binary multipliers; multiplied in parallel and accumulated in series (MAC), which later transformed into parallel multiplication and parallel addition. Parallel multipliers are the oldest among all modern multipliers. On one hand, parallel bit multipliers are fast, they consume a lot of energy and area, so series multiplication was introduced. Whereas on other hand, series multipliers are power efficient and consume smaller area on the die, but they require many clock pulses (equal to input bits) to accomplish multiplication, which causes very long computation latency.

Several types of integer multipliers, other than parallel binary multipliers mentioned above, exist in the literature. Finite field **GF(2^m) multipliers**, for instance, are used for public-key cryptosystems. These types of multipliers based on 2^m number system, are attractive for their simplicity and error detection circuits [35]. An array type **imprecise multiplier** [36] is also common for image/video processing and neural network applications. A larger imprecise multiplier array based on simplified inaccurate (2x2) multiplier block, was also proposed by [37]. In [38], an approximate signed multiplier has been proposed for use in Arithmetic Value Data Speculation (AVDS).

A **truncated multiplier** is also common with a correction circuit for accurate results [39]. Authors in [40] proposed truncated multiplier by cutting LSB in the partial products (and thus removing some adders in the compression stage) to decrease switching average current and area. Variable correction for the truncated multiplier is also proposed by [41]. Figure 2.1 summarizes the types of multipliers briefly discussed above.

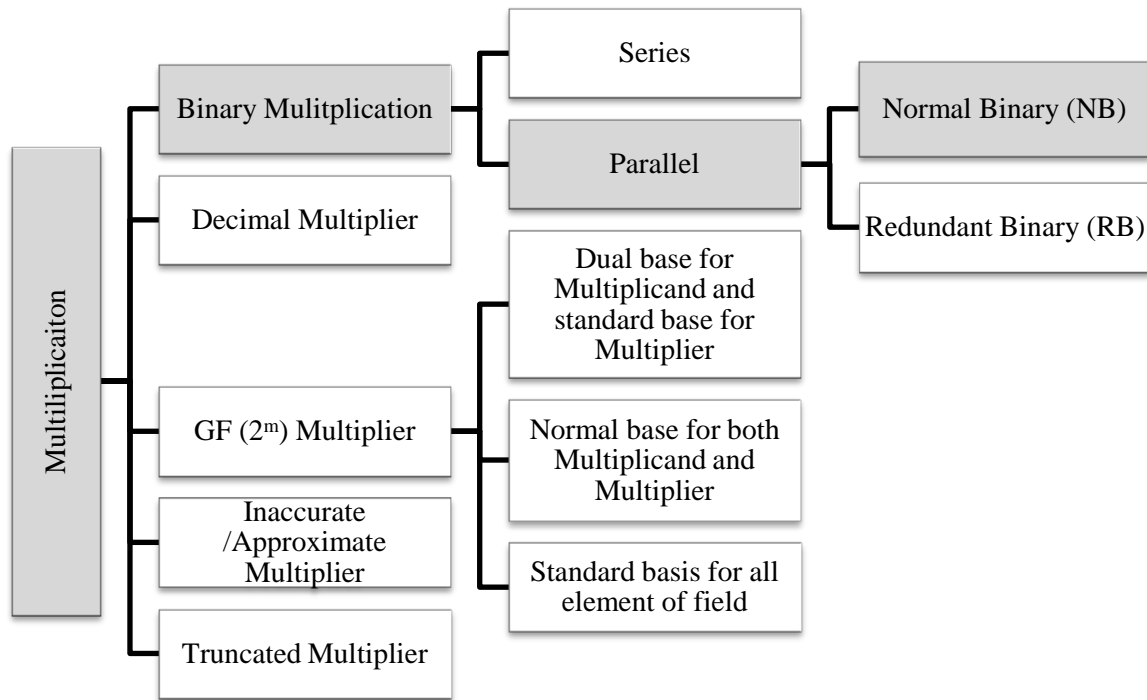


Figure 2.1. Classification of Multipliers

2.1.1 LITERATURE REVIEW

Table 1 summarizes and classifies the gigantic literature on parallel bit multipliers. Each study has been categorized into the year of publication, types of multiplier structure, objective of study, the methodology used to achieve the desired improvement, size of multiplier and most importantly evaluation criteria. Multiplier structure, sub-categorizes into booth encoding, compression algorithm, and type of final addition.

It is depicted in the review Table 1 that major contributions on parallel bit multipliers focused essentially on delay with specified targets. For instance, commonly cited [15] targeted speed of multipliers by optimizing 4:2 compressor circuit in compression stage (using parallelism). In another well-known work [42], authors fully utilize CPL by making the hybrid structure of non-full swing (NFS) and full-swing (FS) 4:2 compressor blocks, again to optimize the delay only. The most recent work on parallel bit multipliers [43] proposed an energy efficient hybrid Wallace structure, with results simulated on the FPGA. Therefore, a generic study that investigates parallel multipliers based on not only conventional (3,2) counters or 4:2 compressor, but also wide compressors i.e. 5:2 and 6:2 compressors, was missing. This study fills this gap by providing a comprehensive analysis through simulation of parallel integer multiplication architectures realized in customized VLSI circuits, not the typical FPGA.

In addition to multiplication architectures, this study also investigates compression circuits used in compression stage of multiplier. Classical work on compression circuits focused only on circuit level optimization i.e. transistor resizing and low power logic etc. For example, authors in [20] optimized 4:2 and 5:2 compressor, which was widely cited afterward. The major contribution by this study, therefore, is to compare and revise evaluation metrics for wide variety of compression circuits. The proposed evaluation metric based on simulated PDP, instead of worst gate delay, suggests that wide compression circuits are not efficient compared to smaller compression circuits. . The detailed discussion on correction made by new evaluation method will be discussed in Chapter 4.

Table 1. Literature review of Multiplication Architectures

Size of Multiplier under test	Process Parameters (Simulation or Fabrication)	Evaluation Criteria	Optimization approach to achieve the goal	Objective	Multiplier Structure			Year	#
					Final addition	Comp. algorithm	Booth Encoding		
(17x17)	RTL with 4 layers interconnects (Fabrication)	Discussed only worst case vector	Interconnect optimization, Carry anticipated to 3 bits for more propagation paths with sum skipped to one row to increase vertical flow.	Delay	NA	Array	No	1971	[44]*
(8x8)	2.7 V GaAs fabrication process (Fabrication)	One input tied to all 1's and other sweeps for all combinations	State of the art GaAs fabrication of large-scale circuits with reliable signal processing	Delay and Power	No (But forecast results)	Array	No (But forecast results)	1982	[45]*
(16x16)	2.7um n-E/D MOS process (Fabrication)	NA	Redundant binary tree compression algorithm	Delay	RB to NB converter	RBA tree	NB to RB converter	1987	[46]*
(16x16)	3.3v 600nm CMOS LSI (Fabrication)	1200 random vectors including series of worst vectors.	Introduce highly regular modified array compression algorithm for regular structure	Delay and Power	Hybrid CLA and CSA	Array	Yes	1987	[36, p. 16]*
(16x16)	3.3V 500nm CMOS (Fabrication)	10,000 test vectors	Implement recursive algorithm using Pseudo NMOS to reduce speed.	Delay and Area	CLA	Recursive (R) algorithm	Yes	1989	[47]*

Size of Multiplier under test	Process Parameters (Simulation or Fabrication)	Evaluation Criteria	Optimization approach	Objective	Multiplier Structure			Year	#
					Final addition	Comp. algorithm	Booth Encoding		
(64x64)	4.9 V 1600nm CMOS (Fabrication)	Pipeline (85MHz)	4:2 compressor tree for the first time	Delay	CPA	4:2 adder tree	Yes	1989	[48]*
(16x16)	4V 500nm CMOS (Fabrication)	On-Chip source follower circuits, including one WC.	Introduce and deploy CPL family first time for 16x16 multiplier with significant benefits	Delay and Power	CLA	Wallace	No	1990	[14]*
(32x32)	800nm 5V CMOS (Fabrication)	3000 random vectors	Use of 4:2 compressor multiplier latency by manipulating parallelism	Delay and Power	CSA	Wallace	Yes	1990	[49]*
(56x56)	1um triple metal level (Fabrication)	NA	Fused Multiplication and addition processed into one cycle of pipelined	Delay	NA	Double precession	NA	1990	[50]*
(54x54)	3.3 V 500nm COMS (Fabrication)	3000 random vectors including 100 WC.	Using RISC one latency, pseudo-CMOS in critical 4:2 compressor blocks	Delay	Hybrid CSA and CLA	Wallace	Yes	1991	[13]*
Only compression stage	Compares multiplier design in three states of the art technologies (Fabricate two slices in Bi-CMOS SABRE tech. only)	Synthesize multiplier by module generator using L language in GDT environment (two slices of IEEE double press. Multiplier)	Introduce and use first time wide compressors and counters families based on FA and HA.	Delay	CLA	Array	Yes	1991	[51]*

Size of Multiplier under test	Process Parameters (Simulation or Fabrication)	Evaluation Criteria	Optimization approach to achieve the goal	Objective	Multiplier Structure			Year	#
					Final addition	Comp. algorithm	Booth Encoding		
(54x54)	5V 800nm CMOS (Fabrication)	20,000 random vectors including 12 WC's	Repeating regular block (7D,4D,3D) consists of 4:2 compressor, without wiring optimization	Delay	Manchester adder based CPA	Modified (Regular) Wallace	Yes	1992	[33]*
(12x9, 16x16, 24x24)	1.2um CMOS 4.2V for speed and 5V for power (Simulation only)	Generate netlist of multiplier for software testing	Software generated Wallace compression tree with minimum intermediate delay	Delay and Power	CSA	Modified (heuristic) Wallace	Yes	1993	[52]
(12x12)	1um CMOS 5V (Fabrication)	Number of clock cycles	Efficient FA design and pipelined approach with self-timed blocks utilizing Quasi Domino circuits.	Delay and Area	Not required	Pipelined CSA tree	No	1993	[53]*
(24x24)	1um, LSI 100k library (Simulation only)	Theoretical critical path and multiple test cases	Flatten the structure by increasing the possible paths in the middle column (by using small compression blocks. Tuned final adder according to arrival profile of signal.	Critical path/Delay	Tuned hybrid adder	Modified Wallace	Yes	1995	[54]
(54x54)	250nm 2.5V (Fabrication)	NA	Reduce DPL based 4-2 compressor structure from 4 to 3 using parallelism. Mux based CLA structure the first time.	Delay	Modified CLA	Wallace	Yes	1995	[15]*
(implement 4x4 forecast 54x54)	800nm 1.5V current mode CMOS (Fabrication)	Number of clock cycles	Reduce critical path in signed digit compression by using multiple-valued implementations using a dual-source coupled pair.	Delay	Signed-digit to binary converter	Pipelined (array structure)	Modified Booth	1995	[55]*

Size of Multiplier under test	Process Parameters (Simulation or Fabrication)	Evaluation Criteria/(Logic used to realize the circuits)	Optimization approach to achieve the goal	Objective	Multiplier Structure			Year	#
					Final addition	Comp. the algorithm	Booth Encoding		
(6x6)	80nm Bi-CMOS 3.3V, 20MHz (Fabrication)	CMOS, CPL+TG, CMOS+CPL	Novel circuits of FA based on hybrid CPL and TG circuits	Comparison of circuit tech. for multipliers	NA	NA	Modified Booth	1996	[19]*
(54x54)	Four metal 300nm 2.5V CMOS (Fabrication)	Three clock pulses of pipelined approach (Domino and PTL)	modified booth Encoding with skewed buffers for each pipelined stage	Delay	CLA	Modified Wallace	Only Multiplicand	1996	[16]**
(12x12)	1um, 5v (Simulation only)	Theoretical critical path and multiple test cases?	Careful injection of inputs to 3D block for global optimization	Optimize critical path/time	Tuned CLA	Wallace	No	1996	[56]
(54x54)	500nm triple metal CMOS 3.3V (Fabrication)	10,000 random vectors with one WC. (Hybrid TGL and CMOS)	Introduce redundant binary multiplier with no additional circuits causing delay penalty	Delay and Power	Mux based RB to NB converter	RB adders (RBA) tree	NB to RB converter	1996	[17]*
(54x54)	250nm 2.5V CMOS (Fabrication)	Not Available (Hybrid CMOS and PTL)	Optimize Booth Encoder and 4:2 compressor circuit to achieve compact compression tree	Transistor count and Area	CPA	CSA tree	Modified (sign select)Booth	1997	[34]*
(54x54)	250nm 2.5V CMOS (Fabrication)	Double stage pipelined frequency (Dual rail domino and PTL)	Improved Domino PTL design of XOR and 4:2 compressor to optimized compression tree, both encoder, and PPG	Delay	CLA	Wallace (4:2 compressor tree)	Yes	1998	[57, p. 25]**

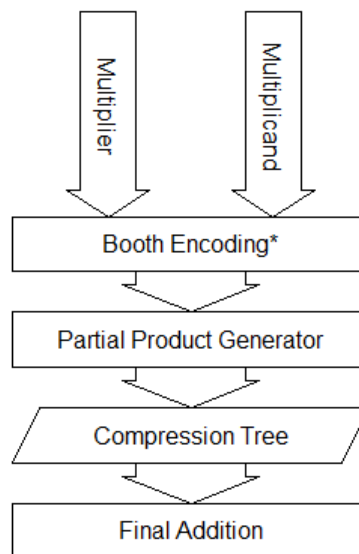
Size of Multiplier under test	Process Parameters (Simulation or Fabrication)	Evaluation Criteria	Optimization approach to achieve the goal	Objective	Multiplier Structure			Year	#
					Final addition	Comp. the algorithm	Booth Encoding		
(16x16)	800nm double metal Bi-CMOS (Fabrication)	Trigger critical path by one of the explained vector (PTL)	Full utilizing of CPL by making the hybrid structure of NFS and FS 4.2 compressor blocks for low power. Optimize PPG like this also.	Power and Area(Penalty in performance)	Carry Select Adder (CSA)	Wallace	Yes	1999	[42]*
(54x54)	350nm 3.3V CMOS (Fabrication)	2,000 random vectors for testing and two worst cases (TGL)	Introduce multiplier in RB without final adder, only efficient conversion block from RB to NB	Power, delay, and Area	Not required	Redundant binary compression tree	Yes	2001	[58]*
(8x8, 16x16, 32x32, 64x64)	250nm, 2.5v Spice	NA	Introduce nonbinary (10,4) and (11,4) counters for a regular compression tree	Worst case power, critical path, Area	NA	NA	Proposed own arrangement of PP's	2001	[59]
(16x16)	90nm 1.95V Dual V _t CMOS (Fabrication)	(Static CMOS)	Compact tiling, Low power right port flop, arrival-profile aware final adder	Power and delay	CLA	Modified 3,2 counter tree	Yes	2006	[18]*
(8x8, 16x16, 32x32)	Altera Quartus-II	1000 random generated vectors (FPGA)	Using wide counter especially (7,3) and (2,3,3) counters were possible for short interconnect	Dynamic power and PDP/EDP	CLA	Wallace	No	2012	[43]
NA	UMC-65nm	NA	Compress partial product by using speculative counters with error correction block	Delay	speculative CPA	NA	Encode some of the partial product	2014	[60]
NA	16nm CMOS	NA	Using approximate compressors to reduce power while compromising output result	Delay, PDP, accuracy	CPA	NA	No	2015	[61]

2.1.2 COMPONENTS OF AN INTEGER MULTIPLIER

Figure 2.2 depicts the three major components of typical parallel integer multiplier; encoder, compression, and final addition. The **Encoder** encodes the input bits to minimize partial products or intermediate results, **compression tree** compresses partial products and the **final adder** sums up the resultant partial products. In [62], two major processes (compression and addition) has been merged together to optimize speed and hardware, which is out of the scope of this study. Each of these three components of multiplier will be discussed separately later in this section.

2.1.3 BOOTH ENCODING

Encoding by using famous Booth technique [63] decreases the number of partial products and depth of compression tree without changing the basic structure of the multiplier. Booth encoding also optimizes the activity before even processing the partial product bits [64]. For example in 8-bit multiplier, the number of the partial products is 64, with eight stage deep compression tree. This can reduce to 56 with five stage deep compression by radix-2 booth encoding [51]. Similarly for 54-bit multiplier, Booth encoding results in half of the partial products (27 bits) [13].



*optional and out of scope of this study

Figure 2.2. Parallel Multiplication flow

Despite that, there are some disadvantages associated with booth encoding, it gives fast compression with slightly high overall average power [65]. In addition, encoded signals go all-over the tree to drive logic blocks, therefore Booth circuitry requires high fan-out capabilities. This is usually accomplished by using expensive Bi-CMOS technologies [51]. Also, there is a trade-off between the size of the multiplier and Booth encoding [66]–[70]. For the small size multipliers delay in encoding, circuitry is typically quite prominent [51], and a few studies [71] suggest the overall negative impact of Booth encoding on multipliers. Therefore, for small sized multiplication architectures simulated in this study (8-bit, 10-bit, 12-bit and 16-bit), encoding circuits has been omitted.

2.1.4 COMPRESSION STAGE

Compression is the most crucial part of multipliers which occupies large silicon area and contributes mainly to dynamic current and most of the delay [20]. In early years, compression was accomplished by splitting partial products into arrays [31], but recent works typically use tree structures i.e. Wallace [29] and Dadda [30] trees for compression. The smaller the size of compression block in a tree, the more the compression tree looks like array structure, and wide the compression block results into irregular trees structures.

Table 1 shows that enormous work has been done on optimization of Wallace tree, while there are very few studies optimize Dadda tree [72] because of its implementation difficulties. For N-Bit Wallace tree, speed is varies with $\log(N)$ of operand size [56], which means Wallace tree have relatively long delays for wide multipliers. For N-bit Wallace tree, the number of (2,2) counters or HA is \sqrt{N} , whereas Dadda tree contains large number of HA i.e. $(N - 1)$ [73], which results into inefficient compression of partial products. Wallace tree is therefore better in terms of both power and delay, also it has flexibility to deploy different types of counters and compressors [43].

The regularity of the compression stages is significant in multipliers since it will increase productivity and result into small capacitive loading of wires [36]. By

compromising on regularity in tree structures, delay and hardware can be optimized, if the partial products of the same weight are lumped together before compression [62]. Regularity makes multipliers predictive and easier to scale-up and implementation of layouts becomes simple and fast. Authors in [74] demonstrate by simulations that regularity and tiling of identical blocks result in low power multipliers. Compared to array structures, irregularity and interconnect overload are common disadvantages associated with Wallace tree, but smaller depth or size and high performance makes Wallace a better candidate for partial product compression [75].

Along with regularity, another challenge of compression stage is to pass the carry signals generated in lower order bits heading towards high order bits. This can be done either by flattening parallelogram structure of partial product tree or by usage of small compression blocks in middle column of compression stage. In the former method, large compression blocks are required (as in ABACUS [76], [77]), whereas later case results in extra switching and high dynamic current. Un-equal size of columns in compression stages of multiplier led to the different arrival time at final addition regardless of the compression algorithm, which can makes final adder also a curtail part of multipliers.

2.1.5 FINAL ADDITION

Final addition deployed after the completion of compression is used to sum-up the processed partial products. Addition starts when there are only two rows of partial product left in compression stage. In the right half of parallelogram structure of compression tree, processed partial products readily available for the addition, while on the left half, partial products arrives after long delays because carries generated by right half and middle column of compression tree. Therefore, in large multipliers, it is worth not to postpone final addition until compression to finish, and therefore final accumulation begins prior to last few stages [13]. Commonly used adders in multipliers are: ripple carries adder (RCA), carry save/select adder (CSA) and carry look-ahead adder (CLA) [51] [56]. Several studies i.e. [33], [54], are dedicated to final addition only as in some cases speed of the final addition dictates the multiplier performance

[33]. In pipelined multipliers particularly, where a delay of the least significant bit (LSB) is essentially same as the most significant bit (MSB), the final adder defines the speed of overall multiplier [57]. Each of the commonly used final adders will be discussed in the subsequent part of this section.

Ripple carry adder (RCA)

Ripple carry adder or RCA is typically considered as the building block of arithmetic and non-arithmetic functions. RCA occupies smaller silicon area and has simple structure with the penalty of large latency[78]. In RCA, (3,2) counters deployed in RCA are equal to the width of added, Approximate delay time (ignoring the capacitance of the wires) in RCA is therefore \ varies linearly with the number of input bits. For this reason, RCA is not suitable for very large multipliers.

RCA structure used in this study is shown in Figure 2.3 with the mentioned worst case (red line). For LSB, (2,2) counter has been deployed to start ripple operation, whereas at MSB, XOR has been used to suppress overflow in RCA. Manchester carries chains (MCC) are the modified version of RCA with high-speed carry propagation. Perri et al. [79] shows that the delay is a quadratic function of chains length in MCC.

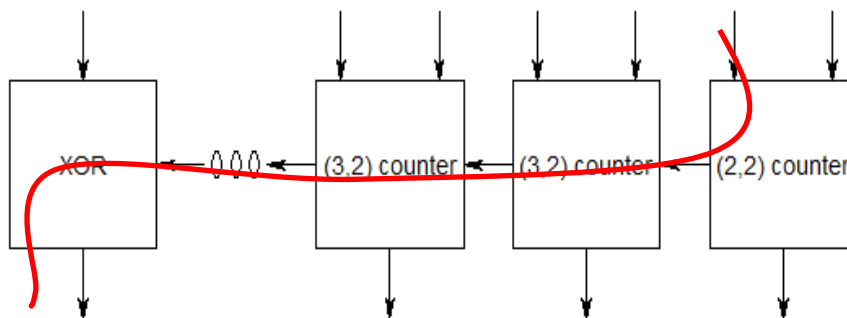


Figure 2.3. Typical structure of RCA

Carry Skip adder (CSA)

CSA is another modification of RCA first introduced by Kilburn et al. [80]. This method, (further discussed by Lehman et al. [81]) shows better performance than other known techniques at that time. In CSA, RCA is divided into groups and the size of each group

depends on, (i) fan-in limits and (ii) a logical decision (either to skip a group or not). The decision is made by a *Skip Block* introduced with each *Ripple block* or a group in CSA. A maximum number of skip blocks are limited by the area of the chip [82]. This method of addition is not suitable for multipliers because of their large size and high power consumption and leakage.

Carry look-ahead adder (CLA)

In 1958 Weinberger et al. [83] first introduced this method which was further modified by several researchers i.e. [84]–[88] and [89]. This is the best-known adder for large multipliers. Richard et al. [90] implemented area efficient model of CLA and found that delay is directly proportional to the *log* of the number of input bits. This means CLA benefits over RCA can be observed clearly in large multipliers (i.e. 54x54, 104x104 or above). The drawback associated with this adder is the chip area, which grows in proportion to width of CLA. The Latency of typical 4-bit block unit based CLA is given by (2.1) [91].

$$T_{\text{latency}} = 4 \log_4 (n + 1); \text{ where } n \text{ is the width of added.} \quad (2.1)$$

Ling [92] proposed optimized CLA with less logical levels. Afterward, Doran [93] suggest other possible variation in CLA for particular applications. Ling’s method is only useful for large arithmetic operations by reason of its complexity [94]. Carry (C_i), generate (G_i) and propagate (P_i) signals in CLA block can be presented by (2.2):

$$C_i = G_i + (P_i \cdot C_{i-1}) \quad (2.2)$$

(2.2) was implemented by [22] using multiplexers to avoid typical cascaded AND/OR gates combination. Noticing (2.2), if the previous carry C_{i-1} is ‘1’, carry of the current stage will depend on propagating signal ($C_i = 1$, if $P_i = 1$ and vice versa), whereas if the previous carry is ‘0’, carry of current stage depends on the generated signal. ($C_i = 1$, if $G_i = 1$ and vice versa). The logical implementation of (2.2) through multiplexer is depicted in

Figure 2.4.

The carry-select adder (CSA) introduced by Bedrij et al. [95] in 1961 provides a compromise between a small area but longer delay RCA and a larger area with the shorter delay in CLA [78]. Recent work on CSA includes [96]–[101]. **Carry Save adder** is another modification of CSA [101].

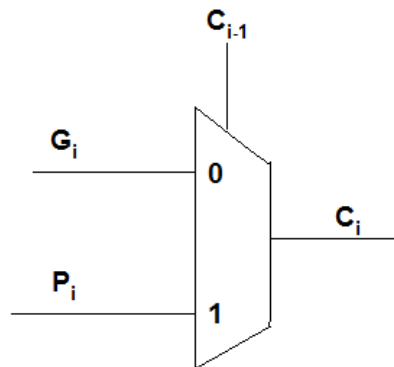


Figure 2.4. Mux gate for implementation of CLA carry

In this study, RCA has been deployed for the final addition because of its simple low power structure. We have tested both RCA and the famous CLA for the multipliers in final addition for 8-bit multiplier and observed a common trend in delay and power variation for all types of multipliers; ,CLA has shorter delays but high average power [102] compared to RCA. In addition to high average power, CLA benefits are prevailing for large multipliers, while in this study we restrict our self to 16-bit multiplier only.

2.2 COUNTER AND COMPRESSOR CIRCUITS

In this Section we will discuss the type and realization of compression circuits. Compression circuits are the area of interest for researchers since 1950’s. They are mainly classified into counters and compressors. Both counters and compressors, essentially serve the same purpose. The simple counter circuit (depicted in Figure 2.5) receives ‘p’ input bits and gives ‘q’ outputs bits to next stage, where ‘p’ and ‘q’ are given by (2.3) [54]. Compressors (Figure 2.5) have intermediate inputs and outputs.

$$q \geq \log_2 (p + 1) \quad (2.3)$$

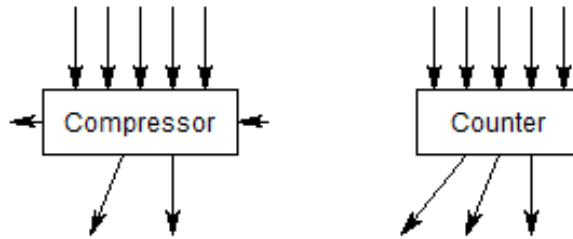


Figure 2.5. Compressor (left) and Counter (right) block

2.2.1 DEFINITIONS

For **conventional (p,q) counter**, output bits ‘q’ are always equal to $(2^q - 1)$. For example, (7,3) counter present seven bits of the same weight (2^0) into three equivalent bits of different weight (2^0 , 2^1 and 2^2) to next stage. On the other hand **p:2 compressor** not only collect p-inputs bits from the previous stage, but also absorb p-3 carries from the right column of the same stage and present two-bits to next stage and bounce the same amount of carries to the left column of the same stage.. For example, 4:2 compressors collects one input carry ($4-3=1$), and four new inputs bits from the previous stage and compress them into two bits to the next stage with one output carry to the column on left. Ideally, outputs of both counters and compressors have same gate delays. This is *theoretically possible* in counters as all inputs bits are *ideally* available for compression. Whereas in compressors, horizontal carries arrive after one or two stages, therefore outputs generated according to available signals.

Compression circuits are mainly divided into **saturated and unsaturated**. A compression circuit is *saturated* if the number of inputs is equal to a maximum number of bits, which can be represented by output bits. For instance, in (7,3) counter, three output bits can represent at most seven bits hence it is a saturated counter, while (6,3) or (5,3) are not saturated. Similarly, 4:2 compressor has three outputs, one having weight 2^0 , while other two has the weight of 2^1 , which can at most represent five bits in 2^0 , therefore, 4:2 compressors are saturated. While inaccurate 4:2 compressors (without bouncing carry to the same stage) [103] are not saturated. **Special compression circuits**, briefly discussed later in the last section of this chapter, do not follow binary rules and built for some specified tasks i.e. image processing.

Counters circuits further classified w.r.t their usage in multipliers, into single column input (**SCI**) and multiple columns input (**MCI**) counters. (3,2) or (7,3) are examples of SCI counters, in which all bits have the same weight, while (2,3,3) and (5,5,4) counters are the example of MCI counters. In MCI counters leftmost number in nomenclature depicts the number of inputs bits from the most significant column, followed by a least significant number of bits and the right most number depicts a number of outputs. Column height of the following stage for cascaded SCI counters is equal to output bits of the particular counter. For instance, (7,3) counter has three outputs, hence cascaded (7,3) counters results into three rows of partial products to next stage. In MCI counters, bits are sparsely distributed and therefore considered as a disadvantage of using MCI counters alone in compression stages.

Either small blocks or direct implementation using logic gates of wide compression blocks is possible. The former method of constructing wide counters/compressors gives regular structure, but high intermediate wires capacitance, which tend to affect critical path. Whereas in the latter case, usage of direct logic gates reduces intermediate wires capacitance and increase the probability of optimization, but it results in high input gate capacitance. In addition, the low activity in compression circuits is also desirable to avoid glitches as the number of transistor increase roughly N^4 for N bit multiplier [104]. Therefore, it is typically appropriate to implement large counters with small counters or compressors [51].

Initially, compression circuits were realized on IC with distributed control over the chip [105]. As research grew, several counter designs have been formulated. For example, threshold logic gates to detect a number of high bits in TTL [105] or faster residual detection function and realization using ROM [106]. Another faster implementation of the counter was proposed in [107] yet it contains complex analog circuitry. In [108], authors introduce *switching trees logic* for counters, which are basically pipelined binary trees for N-channel transistors. Before detailed discussion on each counter and compressor circuit, a summary chart is presented in Figure 2.6 for the compression circuits briefly discussed before.

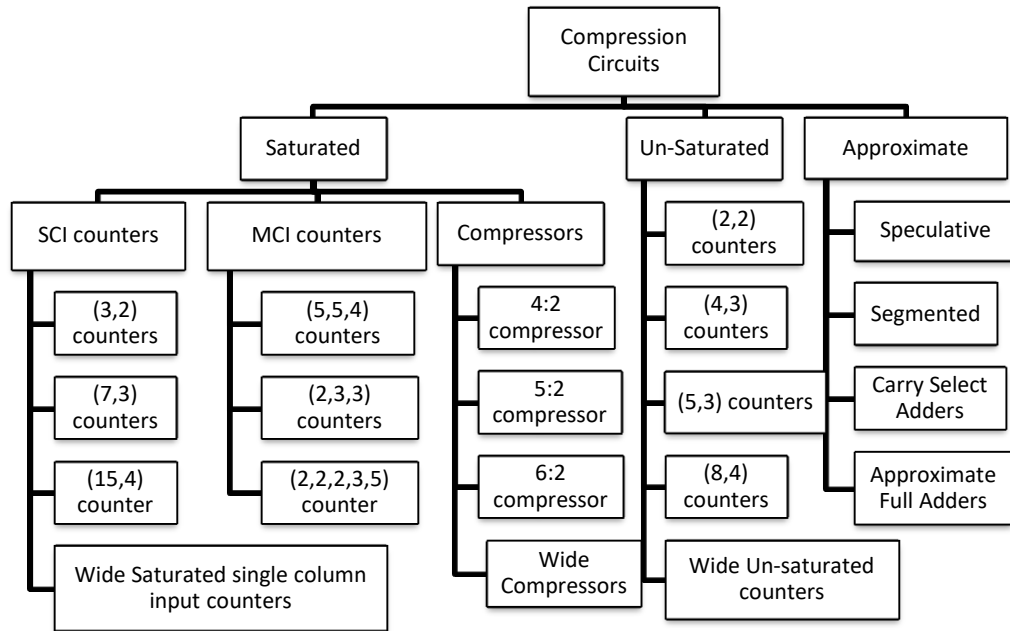


Figure 2.6. Classification of counter and compressor circuits

2.2.2 SATURATED SCI COUNTERS

(3,2) COUNTERS

This is most widely researched counter circuit, also commonly known as full adder (FA) circuit. Some selected examples are [53], [19] and [19], where the major emphasis was to optimize (3,2) counter to enhance overall multiplier performance. Authors in [51] proposed crossed coupled PMOS circuits for high speed (3,2) counter. (3,2) counter or FA circuit with its symbol is depicted in Figure 2.7. This circuit was proposed by N. Ohkubo et al. [15] which comprises of two XOR and one MUX gates.

Both outputs i.e. ‘Sum’ and ‘Carry’, have two gate delays. Typically ‘Sum’ delay is not contributed to multiplier delay [44]. For brevity, instead of drawing inverting and non-inverting signals of DPL, the only single wire has been depicted in all counter and compressor figures of this work. Also, only at the outputs of each counter or compressor discussed here, inverters have been installed to restore the signal quality for next stage.

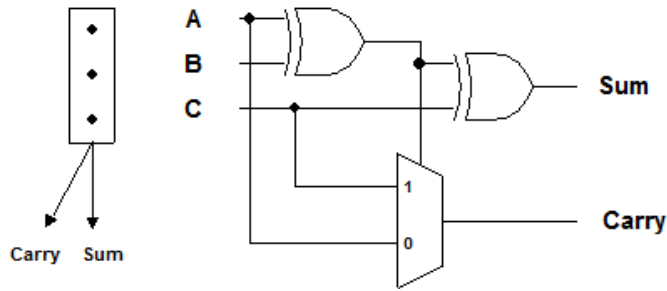


Figure 2.7. (3,2) the counter circuit or full Adder [15]

(7, 3) COUNTER AND (15, 4) COUNTER

Authors in [51] propose (7,3) counter (depicted in Figure 2.8) because of a natural growth concept of using small counters to construct wide counters. (7,3) counter, likewise (3,2) counter, compresses seven bits with an equivalent weight of seven in binary. Depending on size, multiplier implementation based on (7,3) counters are typically more regular compared to equivalent (3,2) counters with smaller wire capacitance [51].

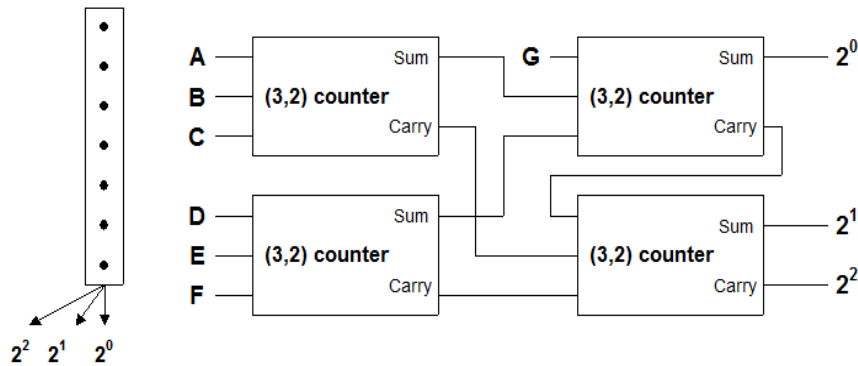


Figure 2.8. (7,3) counter circuit based on full adder [51]

Inverters have been installed at the counter's outputs only, as mentioned before. Connecting intermediate inverters with first stage in Figure 2.8 may give better performance, but counter (7,3) counter circuit becomes heavy and overall average current effects adversely. A similar approach used for (7,3) counter is implemented by [109] to create (15,4) saturated SCI counter depicted in Figure 2.9. Direct implementation of (7,3) is also possible by logic gates which is out of scope of this study.

Wide (15,4) counter depicted in **Figure 2.9** is not discussed much in literature, whereas several studies i.e. [110]–[112] deployed (7,3) counters for compression of partial products. (15,4) counter, likewise (7,3) counter, deploy (3,2) counters to compress input bits. It consists of three stages, twelve new bits processed in first stage, whereas two and one in second and third stage respectively.

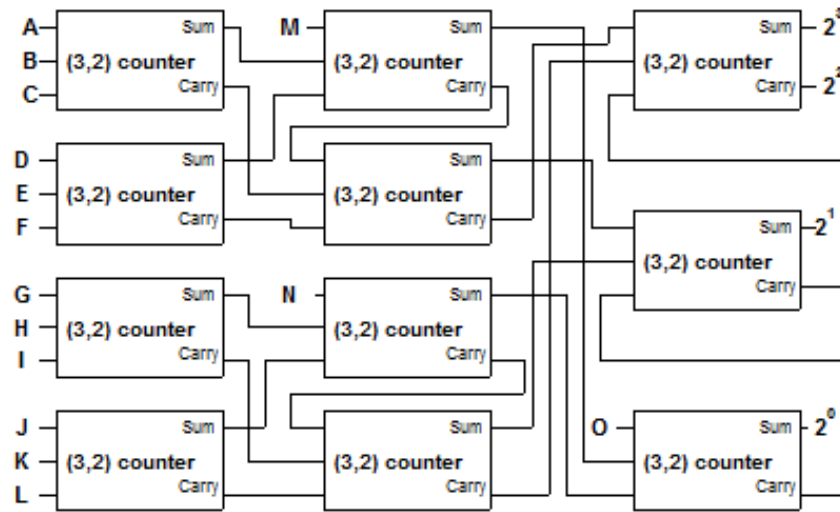


Figure 2.9. (15,4) counter circuit based on full Adder [109]

SATURATED MCI COUNTERS

MCI counters also exist in literature with high compression ratio and deployment flexibility. Since regularity of compression tree is needed, therefore, MCI counters with the same height of input columns i.e. (5,5,4), is more desirable [113]. (2,3,3) MCI counter used in this study is depicted in Figure 2.10. This counter introduced by O. Kwon et.al [114], has the ability to compress bits equivalent to seven bits having the weight of 2^0 . It compresses five bits; three bits of n^{th} column of weight 2^n , similar to (3,2) counter and manipulates carry generated from this (3,2) counter with two more bits of $(n+1)^{\text{th}}$ column, having an equivalent weight of $2^{(n+1)}$ (Figure 2.10). This counter is disregarded as irregular, but one can place this counter in parallelogram structures of partial products tree, where equivalent SCI saturated (7,3) counter cannot be placed.

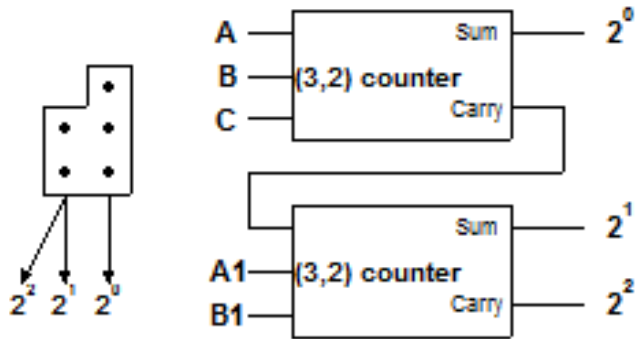


Figure 2.10. (2,3,3) counter circuit [114]

Authors in [113] proposed MCI saturated (5,5,4) counter depicted in Figure 2.11, which later discussed by [115]. This MCI counter compresses five bits having weight ‘n’ and five more bits having weight ‘n+1’,. It has four gates in its critical path. In both Figure 2.10 and Figure 2.11, variables (A~E) having ‘1’ with their nomenclature, represents bits having one bit more significant than other. Since we restrict our multiplication implementation to 16- bits, therefore, this (5,5,4) MCI saturated counters and equivalent wide SCI (15,4) counters has not been deployed in partial product tree. However, these counters have been compared and discussed in Chapter 4 for an inclusive synopsis on compression circuits.

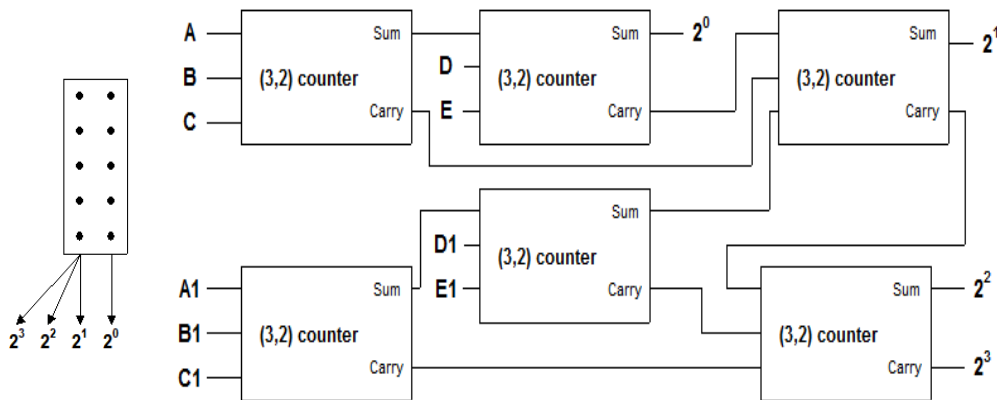


Figure 2.11. (5,5,4) counter symbol (left) and circuit (right) [51]

UNSATURATED SCI COUNTERS

Unsaturated SCI wide counters are not very common in literature because i) smaller compression ratio compared to saturated counters ii) nearly same average power consumption iii) inadequate utilization of resources [113] iv) irregularity v) bulk number of half adder which contains unlikeable AND gates. A simple half adder or (2,2) counter [116], which is essentially unsaturated counter is depicted in Figure 2.12.

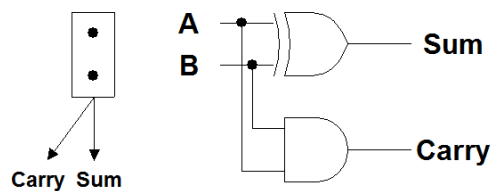
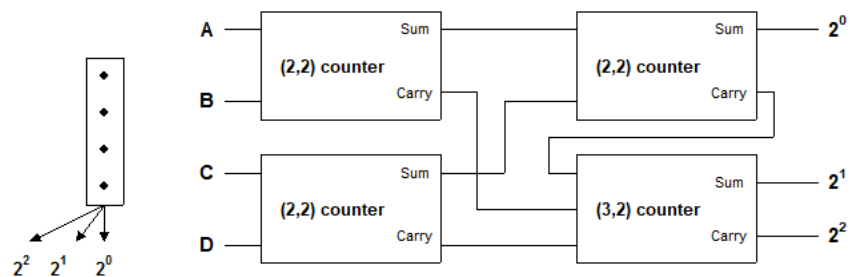
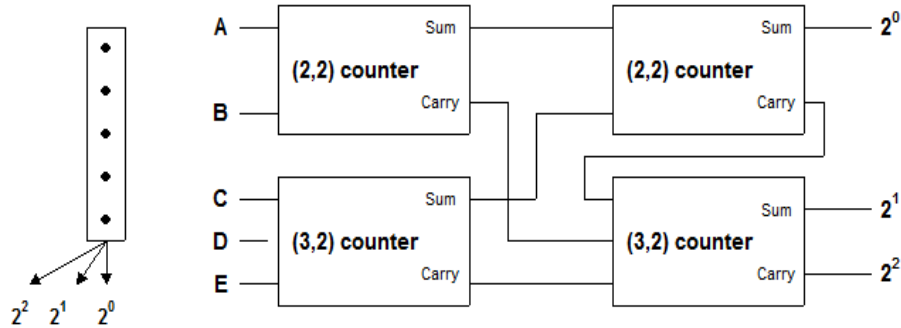


Figure 2.12. (2,2) counter (right), symbol (left)

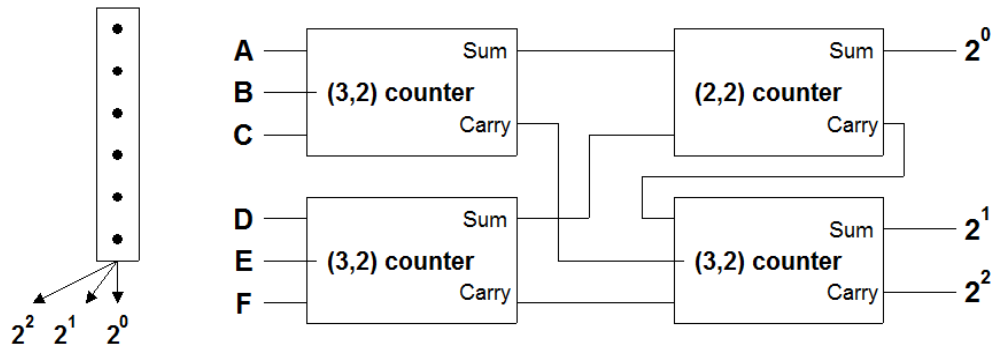
Unsaturated counters are particularly useful when the multiplier has just one more bit than any of the upper-bound numbers in each row of compression tree [117]. For example in 9-bit conventional Wallace multiplier based on (3,2) counters, compression accomplished in three stages. By increment of only one bit, (10-bit multiplier) the number of stages increases from three to four and cause extra leakage and switching. This can be avoided by using unsaturated counter in the first stage of compression stage [117]. Dandapat et al. [118] and [117] are one of the very few studies on the unsaturated counter as unsaturated counters have limited application in compression circuits and multipliers. The basic idea of Dandapat design was to maximize the usage of (3,2) saturated counter or full-adder in compression circuits, to reduce average current. Unsaturated counters from (4,3) to (14,4), based on Dandapat et al. [118] proposed design, are depicted in Figure 2.13.



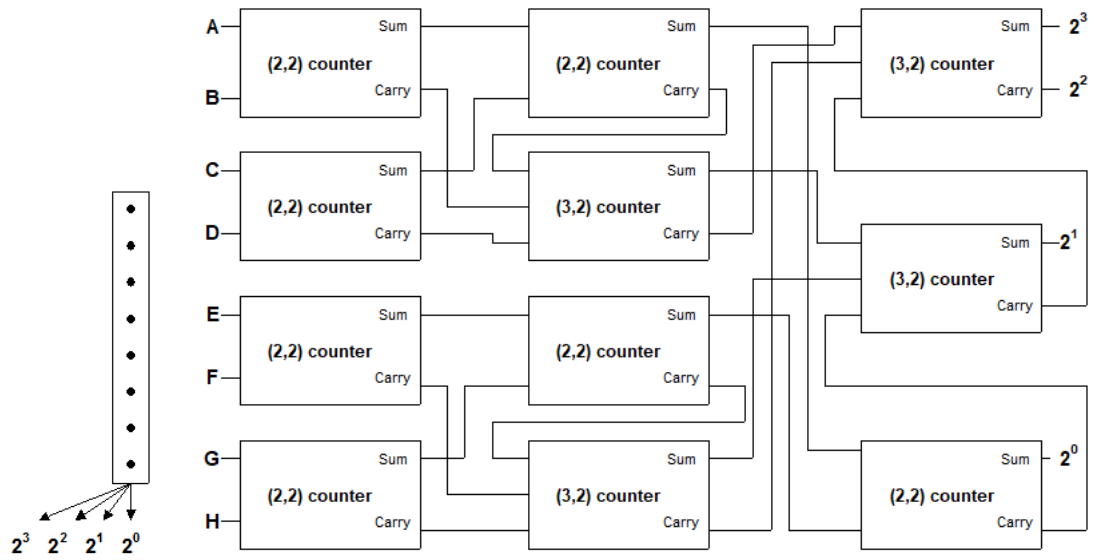
a. (4,3) counter (right), symbol (left)



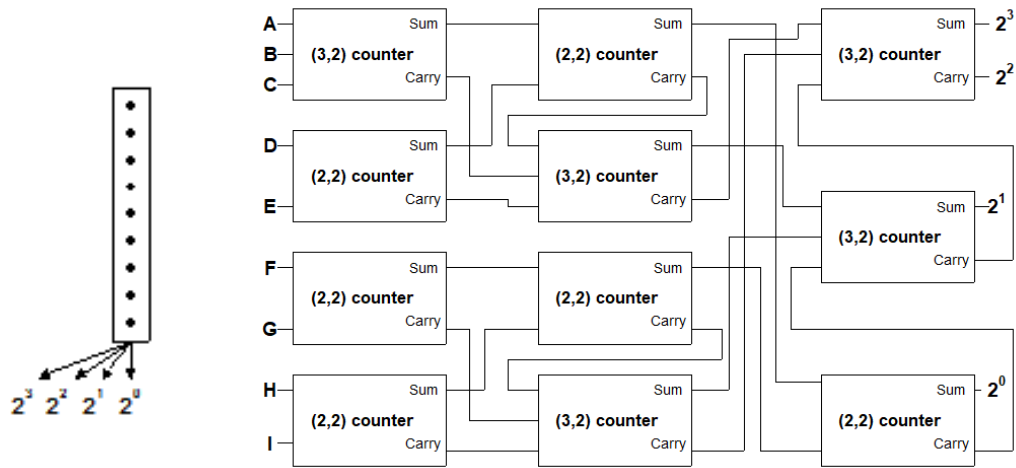
b. (5,3) counter (right), symbol (left)



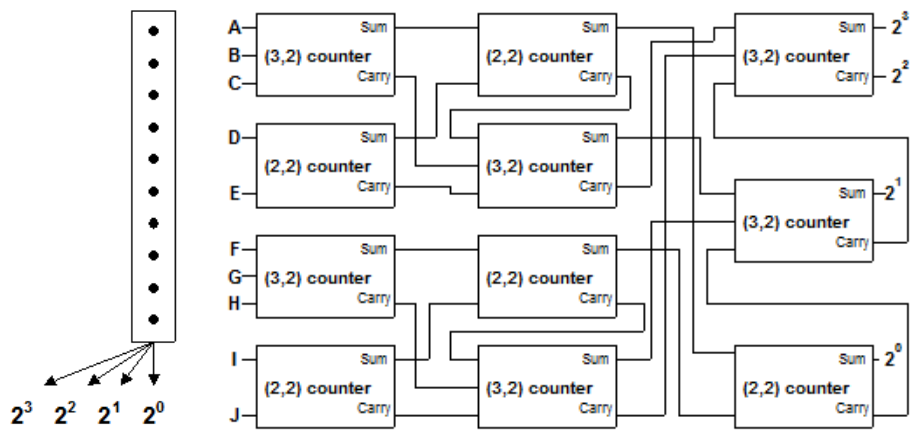
c. (6,3) counter (right), symbol (left)



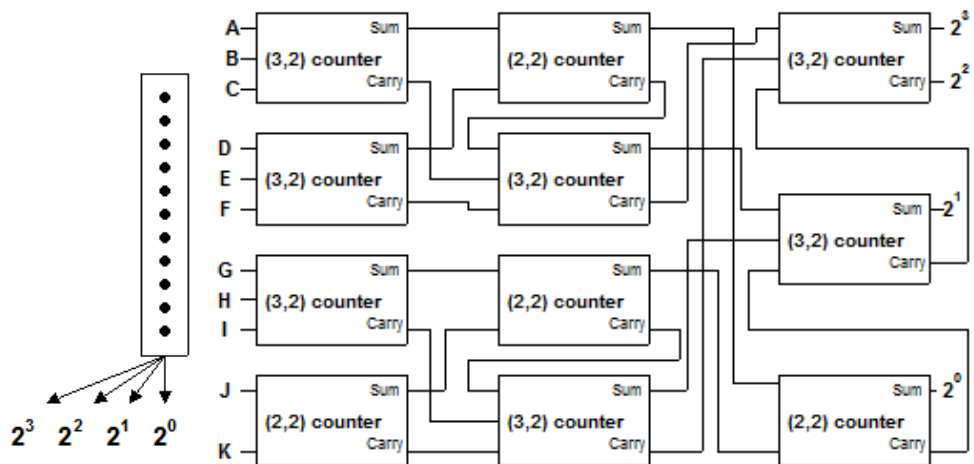
d. (8,4) counter (right), symbol (left)



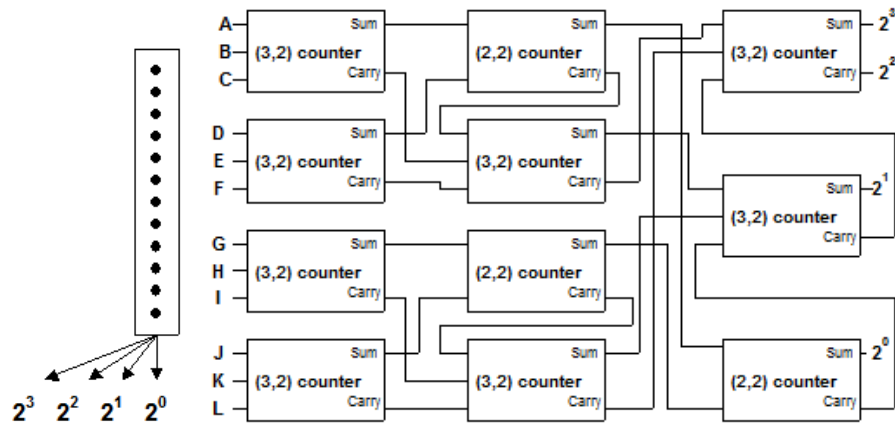
e. (9,4) counter circuit (right), symbol (left)



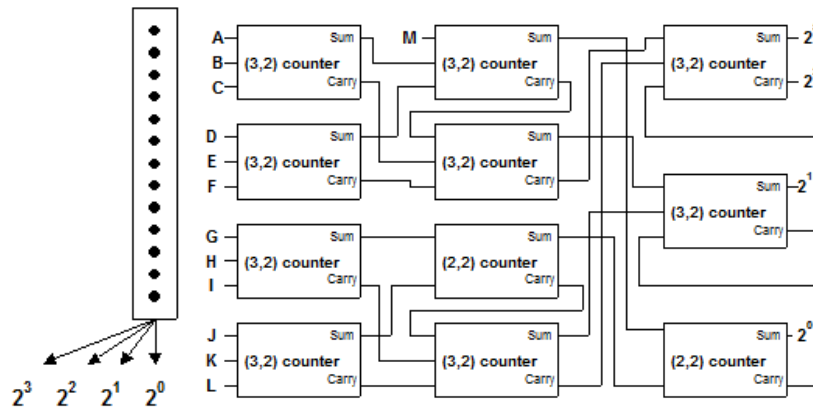
f. (10,4) counter circuit (right), symbol (left)



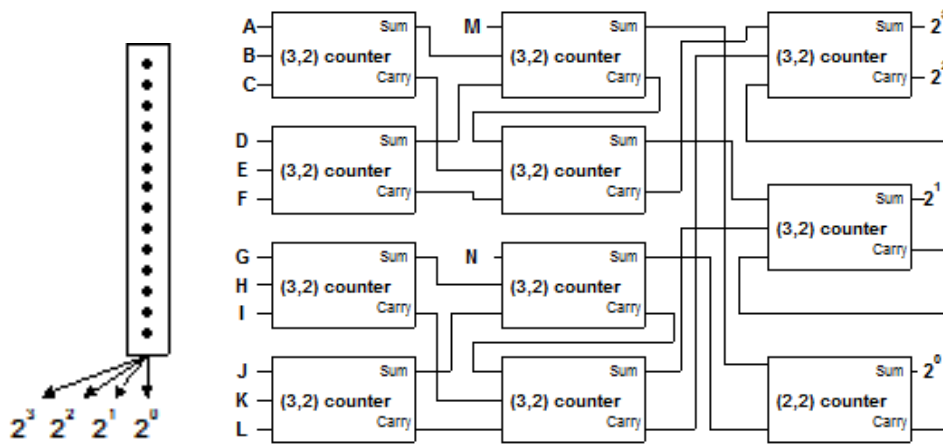
g. (11,4) counter circuit



h. (12,4) counter circuit (right), symbol (left)



i. (13,4) counter circuit (right), symbol (left)



j. (14,4) counter circuit(right), symbol (left)

Figure 2.13. Un-Saturated Counters

2.2.3 SATURATED COMPRESSORS

4:2 COMPRESSORS

A simple illustration of 4:2 compressor is depicted in Figure 2.14, which contains a pair of cascaded full-adders or (3,2) counters [51]. Worst case path of this structure (Figure 2.14) has four XOR gate, which further optimized to three by dissolving cell hierarchy into one single unit using parallelism [15], while the total number of gates remains the same (Figure 2.15). Each output ‘Carry’ and ‘Sum’ has three gate delays, while ‘C_{out}’ intended for next column of the same stage has two gate delays. Several studies i.e. [119]–[122], afterward proposed the modifications in this structure for incremental benefits which is out of scope of this study. Due to its simplicity and flexibility of moving signal in both directions, 4:2 compressors extensively used as building block for multipliers and wide counters i.e. 6:2 compressors and 9:2 compressors. Outputs of 4:2 compressors are given by equation (2.4) to (2.6).

$$\text{Sum} = (A \oplus B \oplus C \oplus D \oplus C_{in}) \quad (2.4)$$

$$\text{Carry} = ((A \oplus B \oplus C \oplus D).C_{in}) + ((A \oplus B \oplus C \oplus D)'.D) \quad (2.5)$$

$$C_{out} = ((A \oplus B).C) + ((A \odot B).A) \quad (2.6)$$

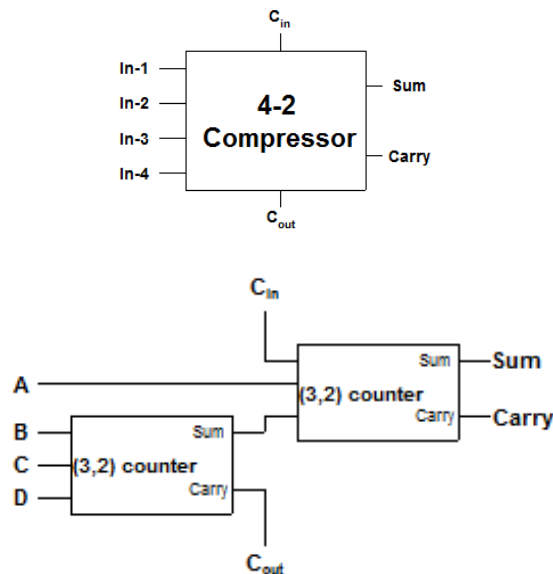


Figure 2.14. 4:2 compressor based on (3,2) counter [51] (bottom) and its symbol (top)

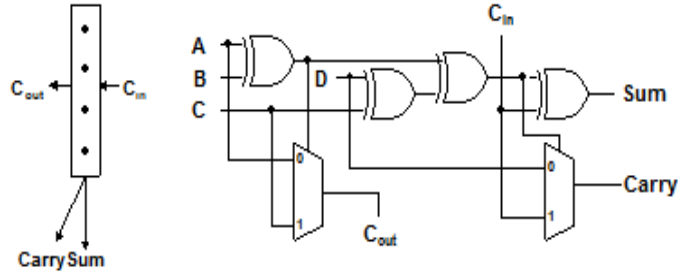


Figure 2.15. Optimized 4:2 compressor(right) [15], symbol used in multiplier (left)

5:2 COMPRESSOR

5:2 compressors compress five bits, excluding two carries interchange ($5-3=2$) and similar to 4:2 compressor, ‘Carry’ and ‘Sum’ to next stage. 5:2 compressors can also realize by three full-adders circuits [51] having the worst delay of five gates as depicted in Figure 2.16. 5:2 compressor circuit further discussed by [122], [123] and optimized by K. Prasad [124] to four gate delays (Figure 2.17). 5:2 compressor structures are in fact the extension of 4:2 compressor with two extra XOR gates and one multiplexer. (2.7) to (2.10) represents Sum, Carry, $C_{out\ #1}$ and $C_{out\ #2}$.

$$\text{Sum} = (A \oplus B \oplus C \oplus D \oplus E \oplus C_{in1} \oplus C_{in2}) \quad (2.7)$$

$$\text{Carry} = ((A \oplus B \oplus C \oplus D \oplus E \oplus C_{in}). C_{in\#2}) + ((A \oplus B \oplus C \oplus D \oplus E \oplus C_{in})'. E) \quad (2.8)$$

$$C_{out2} = ((A \oplus B \oplus C \oplus D). C_{in\#1}) + ((A \oplus B \oplus C \oplus D)'. D) \quad (2.9)$$

$$C_{out1} = ((A \oplus B). C) + ((A \odot B). A) \quad (2.10)$$

Rapid generation of $C_{out\#1}$ and $C_{out\#2}$ is necessary because columns on the left and later stages down in hierarchy depends on these carry signals. Authors in [20] and [124] further optimized the structure in Figure 2.17 by deploying fast carry generation blocks (CGEN1 and CGEN2) from three available bits which is out of the scope of this study. Rather 5:2 compressor block composed of fast generating blocks is *as good as* 5:2 compressor circuit depicted in Figure 2.17 in DPL [20].

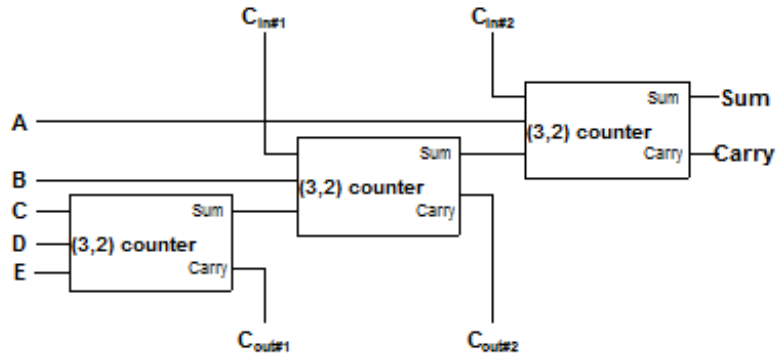


Figure 2.16. 5:2 compressor [51]

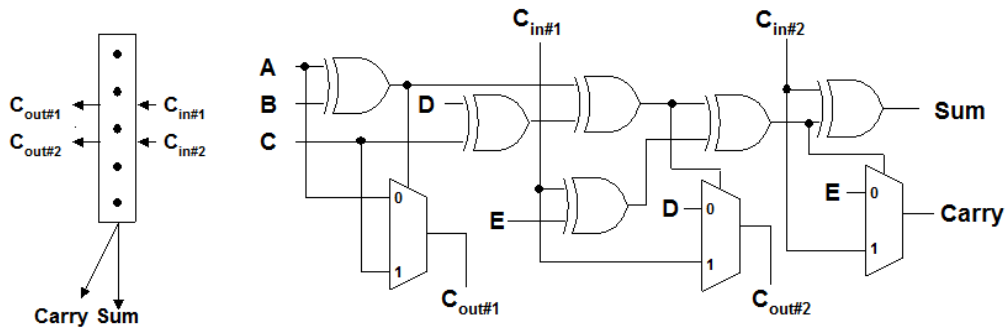


Figure 2.17. Optimized 5:2 compressor [124]

6:2 COMPRESSOR

Similar to 4:2 and 5:2 compressors, 6:2 compressor made up of (3,2) counter is depicted in Figure 2.18. This design is inefficient especially for wide compression trees since the worst path has eight gate delays. Extending the optimization concept of 4:2 and 5:2 compressors, 6:2 compressor can reduce to fewer XOR and MUX gates as depicted in Figure 2.19. This design has five gate delays for ‘Sum’ and ‘Carry’. An alternate method depicted in Figure 2.20 proposed by [51] to construct 6:2 compressors is by using optimized 4:2 compressor and (3,2) counter. This design approach is more desirable since it composed of smaller compressor units, which allow tiling process of very large compression trees.

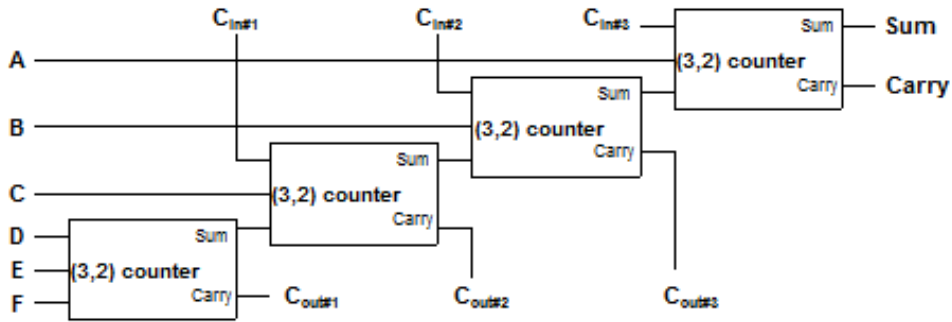


Figure 2.18. 6:2 compressors

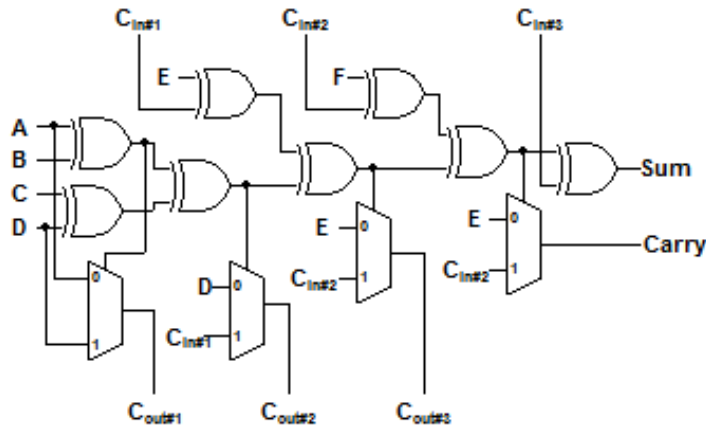


Figure 2.19. Optimized 6:2 compressor

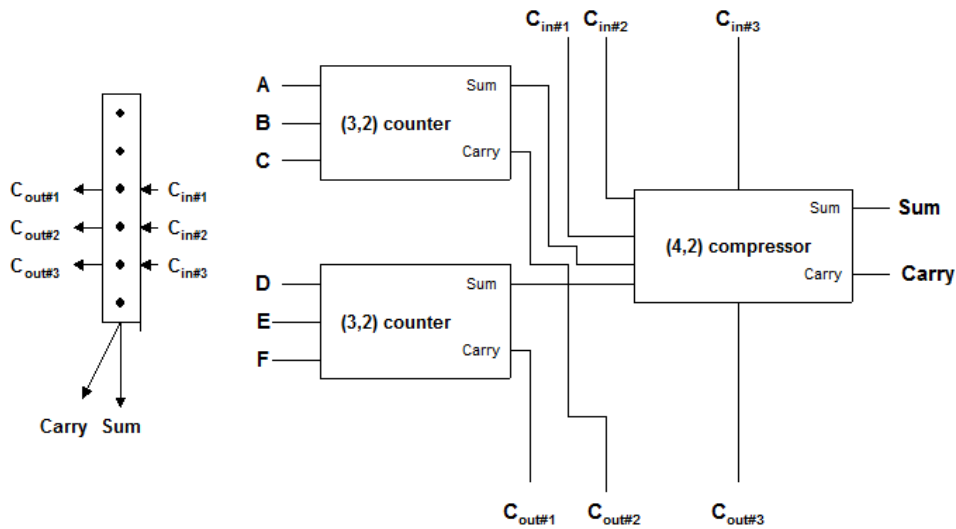


Figure 2.20. Optimized 6:2 compressor [51]

(2.11) to (2.15) represents the output carries to next columns of the same stage and to next stage. . Similar to 6:2 compressor authors in [51] proposed 9:2 compressor (later discussed by [54]) composed of 6:2 compressor and two (3,2) counters, which is out of the scope of this study.

$$\text{Sum} = (A \oplus B \oplus C \oplus D \oplus E \oplus F \oplus C_{in1} \oplus C_{in2} \oplus C_{in3}) \quad (2.11)$$

$$\text{Carry} = ((A \oplus B \oplus C \oplus D \oplus E \oplus F \oplus C_{in\#1} \oplus C_{in\#2}).C_{in\#3}) + ((A \oplus B \oplus C \oplus D \oplus E \oplus F \oplus C_{in\#1} \oplus C_{in\#2})'.F) \quad (2.12)$$

$$C_{out3} = ((A \oplus B \oplus C \oplus D \oplus E \oplus C_{in}).C_{in\#2}) + ((A \oplus B \oplus C \oplus D \oplus E \oplus C_{in})'.E) \quad (2.13)$$

$$C_{out2} = ((A \oplus B \oplus C \oplus D).C_{in\#1}) + ((A \oplus B \oplus C \oplus D)'.D) \quad (2.14)$$

$$C_{out1} = ((A \oplus B).C) + ((A \odot B).A) \quad (2.15)$$

2.3 SPECIAL COMPRESSION CIRCUITS

Unsaturated compression blocks are also known as *approximate or inexact compression circuits*, does not fully represent inputs bits. In approximate compression circuits [40] outputs are not completely deterministic and their usage is significant, where *accurate results are not required*. For example, in image and video processing applications, where a small compromise on quality led to large power and area benefits in processing circuits. Therefore for a meaningful comparison, approximate compression circuits evaluation criteria divided into two categories: average current, delay, PDP, EDP etc. and error distance (ED), mean error distance (MED) and normalized error distance (NED) etc. [125].

R.Lin [59] introduce *non-binary counters* for 8-bit and 4-bit multiplier to achieve low power high-speed with the small activity and area. Since the two-thirds portion of the inaccurate counter circuit process 0's, therefore, the circuit has less leakage current. In another paper [74], the author proposed (6,3) non-binary or unsaturated counter for the regular 8x8 multiplier. Partial product distribution in their proposed regular topology is

similar to ABACUS [76], while a major component of multiplier i.e. (6,3) unsaturated counter, is based on shift switch logic circuits [126].

A different example of (3,2) approximate counter was proposed by [127]. Authors in [61] deploy conventional SCI (3,2) to construct *approximate 4:2 compressor*. Conventional 4:2 compressor has four input bits which require three bits at the output (including C_{in}), but approximate 4:2 compressor contains only two output bits, therefore certainly giving errors for few combinations. *Speculative inaccurate counters* represent output only in two bits irrespective of a number of inputs to it, which means speculative counter works fine until only three of input bits are high, for rest, it gives errors, which needs to be correct in next cycle of the process. Authors in [60] claim performance benefits by using speculative counters.

2.4 CHAPTER SUMMARY

Parallel bits multipliers mainly composed of Booth encoding, compression of partial products, and final addition. Booth encoding is out of the scope of this study since it requires high fan-out capabilities. Compression mainly discussed in this study is the most time and energy consuming process in multipliers. Different counter and compressor used in compression stage exist in literature with different compression ratio, gate delays and a total number of gates. Saturated (3,2) counter is most widely discussed, while 4:2 compressor is most widely used in multipliers. SCI counters results into regular tree structure compared to MCI counters.

CHAPTER 3

MULTIPLIERS UNDER STUDY

This chapter discusses and compares multiplication architectures under study. Architecture details, methodology, advantage, and disadvantage of architectures based on a number of logic gates, have been presented in comparison. Five different architectures of Wallace architecture have been investigated based on (3,2) counter, 4:2 compressor, 5:2 compressor, 6:2 compressor architectures and hybrid Wallace multiplier. Hybrid Wallace multiplier, perhaps the most recent work on Wallace multipliers and composed of wide (7,3) counter and (2,3,3) counters. To best of our knowledge, this type of detailed comparison of VLSI multipliers has not done before, especially using wide compression block i.e. 5:2 and 6:2 compressor blocks. The multipliers architectures later have been compared the w.r.t number of compression blocks, stages, size of final adder etc. For a better understanding of trends in multipliers behaviors, four sizes multiplier have been investigated which includes 8x8, 10x10, 12x12, and 16x16. This gives four data samples to forecast result for wide multipliers. Several logical reasons with theoretical backups have been presented based on configuration comparison of above-mentioned architecture, which forecasts *expected* superiority of multiplier architectures over others. Lastly configuration of each multiplier, including logic blocks, number of stages, unit devices etc. has been summarized in form of tables for all architecture and presented in figures to observe graphically the trends in multipliers.

3.1 WALLACE MULTIPLIER

3.1.1 WALLACE MULTIPLIER BASED ON (3,2) COUNTERS

Wallace [29] introduce fundamental concept of parallel bits reduction for multipliers, which theoretically discussed further by Strollo et al. [128]. *8x8 bits Wallace multiplier* based on (3,2) counter is shown in Figure 3.1. This architecture has been used as a

reference by several studies i.e. [129] and [43]. Recursive equations for Wallace multipliers to calculate the height of partial products for any compression stage also presented by [128]:

$$W_0 = N \quad (3.1)$$

$$W_{j+1} = 2 \text{ floor} \left(\frac{W_j}{3} \right) + (W_j \bmod 3) \quad (3.2)$$

In (3.2) the first term is multiplied by two because at each position in a given column; two processed bits are expected, one from the same column ('Sum') and one from right column ('carry'). 'Floor' function has been used since a number of counters is an integer number. The second term in (3.2) containing mod, represents the bits left in the previous stage after stack of (3,2) counters. In the first stage of the 8x8 multiplier, (W_0) height of the stage is equal to the size of the multiplier ($N = 8$) from (3.1). For second stage (W_1), (3.2) becomes: $2 * \text{floor} (8/3) + (8 \bmod 3) = 6$, therefore, the second stage has a height of six partial products. Similarly, height reduces to four and three partial product bits in 3rd and 4th stage respectively, as depicted in Figure 3.1.

In each stage of 8-bit Wallace multiplier based on (3,2) counters (Figure 3.1), row is paired in a set of three and (3,2) counters have been used in each column of the set. Extreme ends of columns, where two bits are left, (2,2) counter has been used inevitably. At most two sets of three bits can be made out of eight bits of partial products in the first stage. The remaining two rows of partial products have been passed to next stage for further processing, even if (2,2) counters can be deployed. Otherwise, if (2,2) counter used for last two row in the first stage of Figure 3.1, the structure of compression tree and interconnect overhead will becomes unpredictable, inefficient and even more irregular. Size of the final adder can be calculated by using (3.3) given by [128]. Note that characteristic equations present in this chapter are independent the logic used to realize circuit and process technology etc.

$$\text{Size of final Adder} = \begin{cases} 2N - (\# \text{ of stages}) - 2 & \text{if } 3 \leq N \leq 6 \\ 2N - (\# \text{ of stages}) - 1 & \text{if } N \geq 6 \end{cases} \quad (3.3)$$

If the size of multiplicand and multiplier is different, say $M \times N$, then instead of ‘ $2N$ ’ in (3.3), $(M \times N)$ will replace ‘ $2N$ ’. For 8×8 Wallace multiplier based on (3,2) counters, a number of compression stages is four. Putting this number in (3.3) gives 11 bits in a final adder, which is evident from Figure 3.1. The total number of (3,2) counters can be found in final addition stage by using (3.4) given by [128]. For multiplier size wider than 6×6 , one or two will be added depending on leftover bits at the most significant column.

$$\# \text{ of (3,2) counters in final adder} = \begin{cases} N^2 - 2N + (\# \text{ of stages}) + 3 & \text{if } 3 \leq N \leq 6 \\ N^2 - 2N + (\# \text{ of stages}) + 2/1 & \text{if } N \geq 6 \end{cases} \quad (3.4)$$

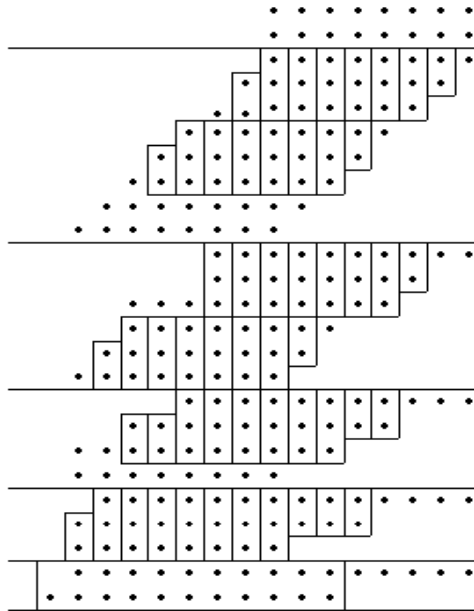


Figure 3.1. 8×8 Wallace Multiplier based on (3,2) counters [128]

Ripple carry adder (RCA) realized by (3,2) counters, discussed in Section 2.2 of Chapter 2, has been used for final addition in this work. The number of (3,2) counters in ‘ M ’ bits wide RCA are $(M-2)$, whereas (2,2) counter only installed at the beginning of RCA to start to ripple operation. The overflow or carry signal from a most significant column of the multiplier has not been used and therefore suppressed by using single XOR gate.

10x10 bits Wallace multiplier based on (3,2) counter is depicted in Figure 3.2. Applying (3.1) and (3.2) gives the height of partial products as ten, seven, five, four, three respectively. Final adder width is also in agreement with (3.3) i.e. $2*10 - 5 - 1 = 14$.

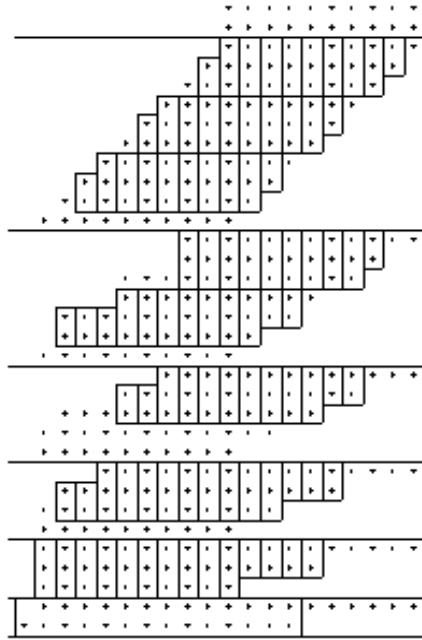


Figure 3.2. 10x10 Wallace Multiplier based on (3,2) counters

12x12 bits Wallace multiplier based on (3,2) counters discussed by [30] is depicted in Figure 3.3. The pairing of rows of three-bit has been done in a similar manner as discussed before. In the first stage, four set of rows, while in the second stage, only two sets of rows are possible, rest of bits have been passed to next stage. By applying equations, (3.1) and (3.2) validates the compression structure. gives; $W_0 = N = 12$, and height of the second stage of compression is $W_1 = 8$. Similarly, height/depth of structure decreases from six, four and then three in 3rd, 4th, and 5th stages respectively. From (3.3) size of the final adder is 18-bit wide. i.e. $(2*12 - 5 - 1) = 18$ bits. 16x16 bits Wallace multiplier based on (3,2) counter is shown in Figure 3.4. This architecture has been simulated and discussed by several studies including S. Abed et al. [43]. Applying recursive equations (3.1) and (3.2), for 16-bit multiplier gives partial products height for 2nd, 3rd, 4th, 5th and 6th stages as 11, 10, 8, 4 and 3 bits respectively. Size of the final adder calculated by (3.3) of 16x16 is twenty-five bits, which is in agreement with the structure shown in Figure 3.4.

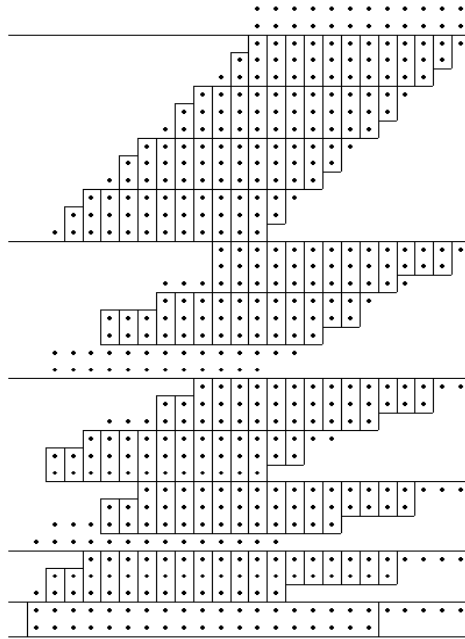


Figure 3.3. 12x12 Wallace Multiplier based on (3,2) counters [30].

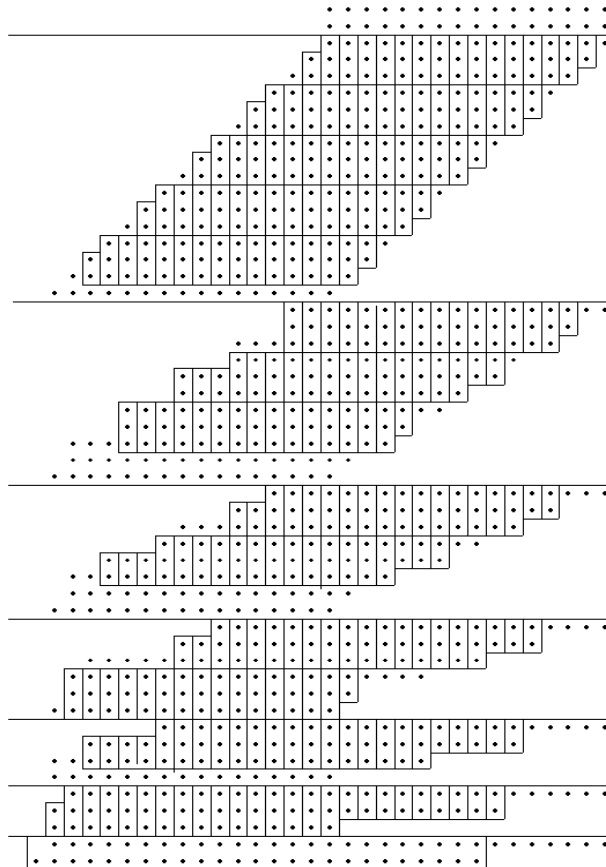


Figure 3.4. 16x16 Wallace Multiplier based on (3,2) counters [43]

3.1.2 WALLACE MULTIPLIERS BASED ON COMPRESSORS

Typical implementations of Wallace tree multiplier are based on (3,2) counters. However, with the advancement in research on compression circuits, wide compression blocks also deployed in Wallace multiplier. Wide compressors equalize arrival time of carrying signals from the right [130] and reduce interconnects and activity. Some selected examples of using 4:2 compressors in Wallace implementation are: [13],[33],[42],[49],[57]. Wallace tree multipliers based on 4:2 compressors is unlike (3,2) counter based architectures, allows the horizontal flow of carries, which flattens the parallelogram structure of multiplier and slightly moves the worst path towards left [130]. In spite of that, the efficiency of wider compression blocks does not increase linearly w.r.t their size [54]. We will discuss this in more detail and prove by simulation results in Chapter 5. For wider compressors based Wallace implementations (3.1) and (3.2) still, hold with minor changes. The updated characteristic equations for Wallace multiplier based on ‘p’ input compressors are:

$$W_0 = N \tag{3.5}$$

$$W_{j+1} = 2 \text{ floor} \left(\frac{W_j}{p} \right) + (W_j \bmod p) \tag{3.6}$$

Whereas, equations (3.3) and (3.4) remains holds true for compressors based Wallace implementation. Wallace multiplier based 4:2 compressors for 8x8, 10x10, 12x12 and 16x16 are depicted in Figure 3.5, Figure 3.6, Figure 3.7 and Figure 3.8 respectively.

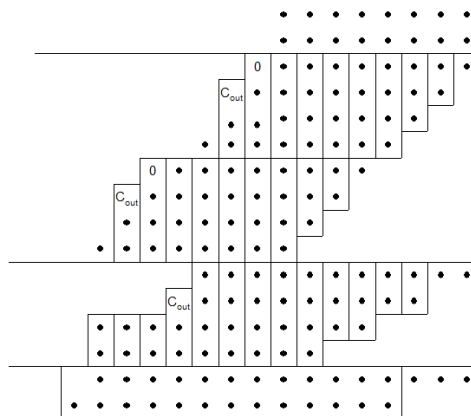


Figure 3.5. 8x8 Wallace Multiplier based on 4:2 compressors [131]

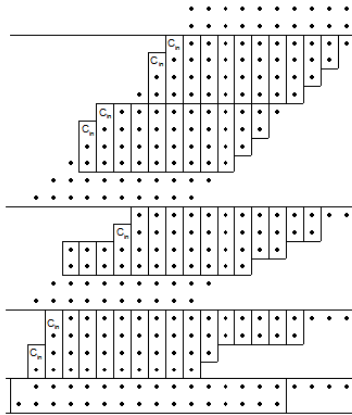


Figure 3.6. 10x10 Wallace Multiplier based on 4:2 compressors

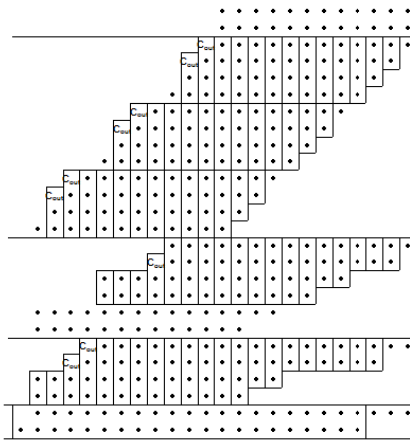


Figure 3.7. 12x12 Wallace Multiplier based on 4:2 compressors

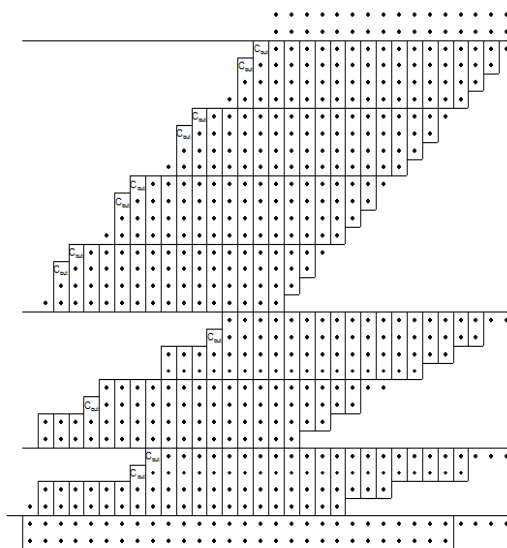


Figure 3.8. 16x16 Wallace Multiplier based on 4:2 compressors

Similar to Wallace based on (3,2) counters, in every stage of 4:2 compressor based implementations, partial products segregated into groups of four bits and 4:2 compressors have been used in each column to compress bits. Remaining rows of partial products passed to next stage for further processing. It can be noted in carrying in and carry out of 4:2 compressor cannot observe from aforementioned figures. At the extreme ends of rows in compression stage, either input carry is grounded permanently to deploy 4:2 compressor or (3,2) counter has been used. In the former case, carry out from following 4:2 compressor acts as a bit from the previous stage (mentioned as C_{out} in figures). Similarly, 5:2 and 6:2 compressors based Wallace implementations of all prescribed sizes are depicted in Figure 3.9 to Figure 3.16.

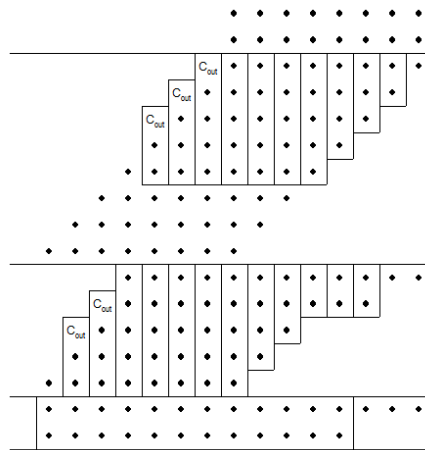


Figure 3.9. 8x8 Wallace Multiplier based on 5:2 compressors

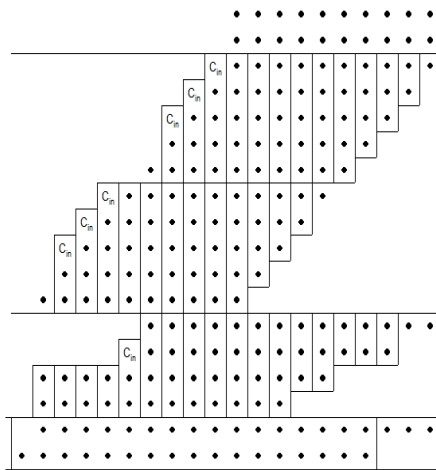


Figure 3.10. 10x10 Wallace Multiplier based on 5:2 compressors

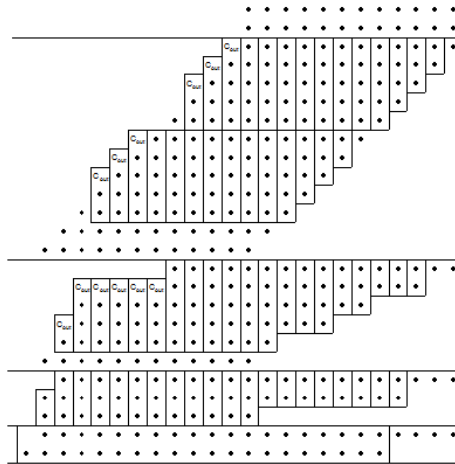


Figure 3.11. 12x12 Wallace Multiplier based on 5:2 compressors

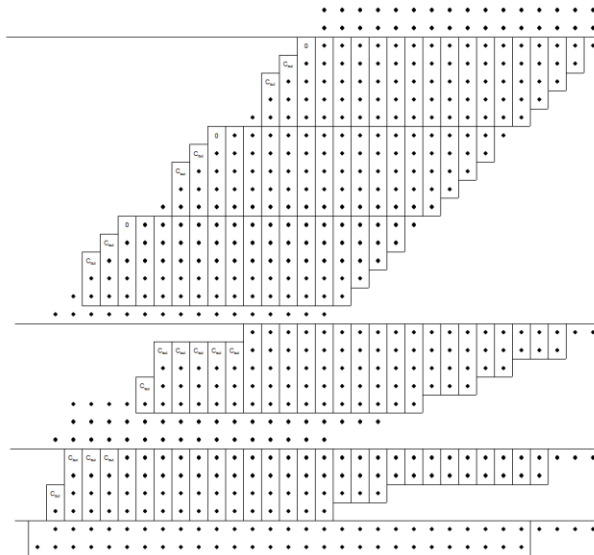


Figure 3.12. 16x16 Wallace Multiplier based on 5:2 compressors

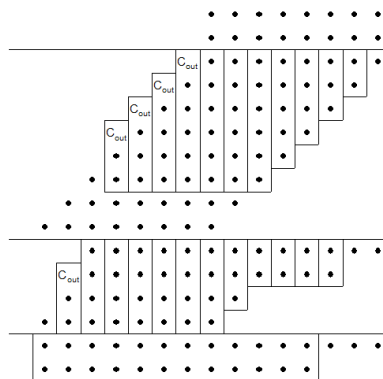


Figure 3.13. 8x8 Wallace Multiplier based on 6:2 compressors

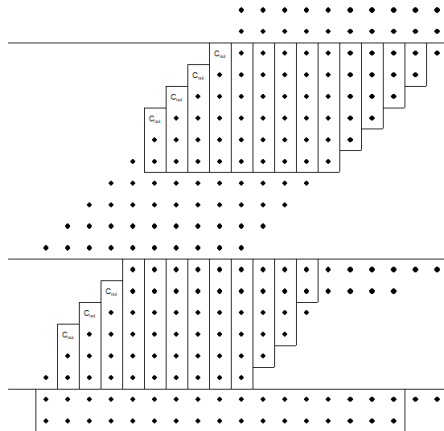


Figure 3.14. 10x10 Wallace Multiplier based on 6:2 compressors

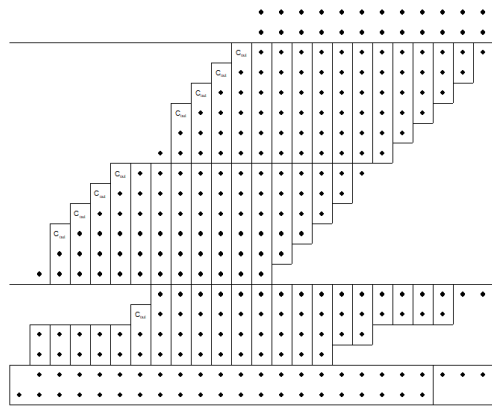


Figure 3.15. 12x12 Wallace Multiplier based on 6:2 compressors

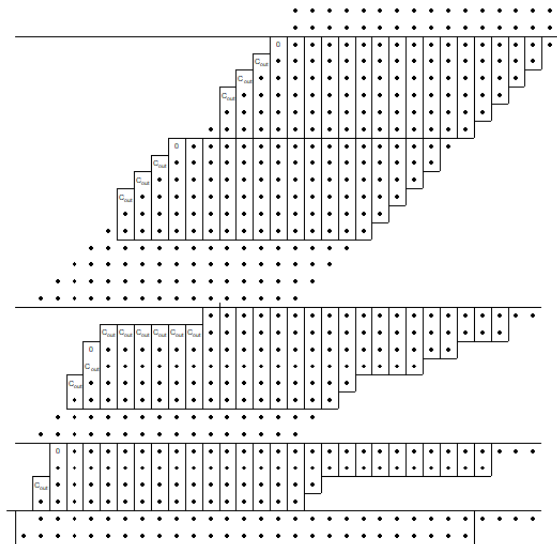


Figure 3.16. 16x16 Wallace Multiplier based on 6:2 compressors

3.2 HYBRID WALLACE MULTIPLIER

A large portion of literature in mid-90's on parallel bits binary multiplier is devoted to the improvement in Wallace multipliers based on (3,2) counters or 4:2 compressor. Correspondingly, S. Abed et al. [43] proposed hybrid Wallace by modifying compression tree by deploying saturated wide counters i.e. (7,3) and (2,3,3) counters. Proposed 8-bit hybrid Wallace multiplier structure is depicted in Figure 3.17. Authors, prefer (7,3) and (2,3,3) counters and limits the width only to seven bits, because of the next saturated counter i.e. (15,4) counter, is either not conceivable or have poor performance compared to equivalent smaller saturated counter units, as discussed in Chapter 4.

In the proposed hybrid 8-bit Wallace multiplier (Figure 3.17), three (7,3) counters initially placed, after that (2,3,3) counter were deployed where possible, and later (3,2) and (2,2) counter has been used to process the leftover bits. A similar approach was followed in all stages to make a compact hybrid Wallace structure. Authors projected the proposed algorithm from 8-bit, 16-bit and 32-bit multipliers. Using the similar approach, by giving priority to wider saturated counter, in this study, we have implemented 10-bit and 12-bit multipliers, along with 16-bit multiplier to have a fair comparison with conventional Wallace multiplier. 10x10 and 12x12 hybrid Wallace architectures are depicted in Figure 3.18 and Figure 3.19 respectively. Originally proposed 16-bit multiplier by S. Abed et al. [43] contains a small mistake in the drawing, which was corrected in Figure 3.20.

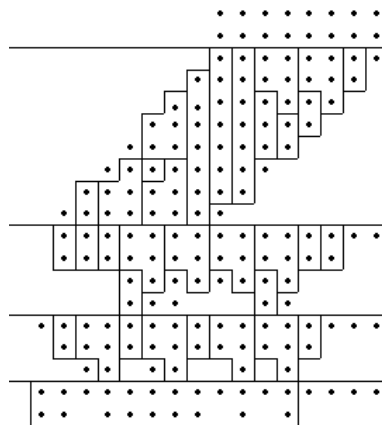


Figure 3.17. 8x8 hybrid Wallace Multiplier [43]

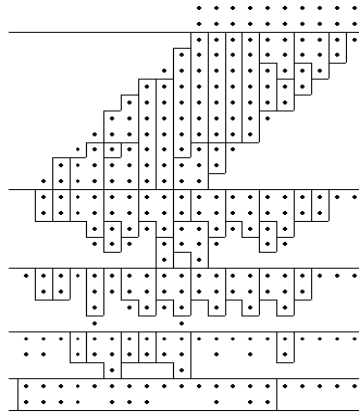


Figure 3.18. 10x10 hybrid Wallace Multiplier

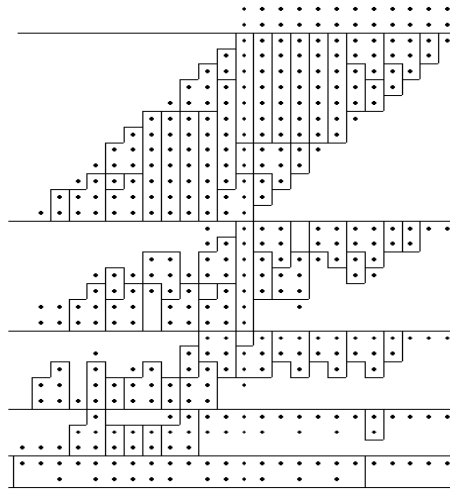


Figure 3.19. 12x12 hybrid Wallace Multiplier

Since the proposed algorithm of hybrid Wallace is based on hand-picked placement of prioritized wide counters, therefore no generic equations have been driven by authors to calculate configuration of proposed compression stage (i.e. number of stages, number of SCI and MCI counters etc.), whereas the size of the final adder still can be found by (3.3). For RCA in this hybrid Wallace approach, the number of (3,2) counter cannot be determined by generic principle $(M - 2)$, because some columns in final adder contain only one bit, which requires (2,2) counters. Authors argued this inconsistency as a potential advantage over Wallace based on (3,2) counters. Indeed, it is a drawback because columns where only one bit left, inefficient unsaturated (2,2) counter will be used instead of saturated (3,2) counter in RCA. Moreover, if the final addition is CLA (as

authors used in their work), one bit should leave empty or permanently connected to ground, to maintain homogeneity of cascaded 4-bit CLA modules. This results into the excessive usage of resources which tends to high average current and PDP.

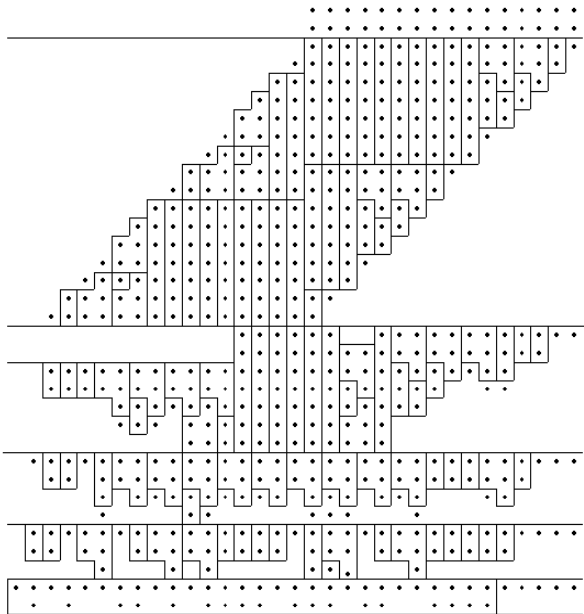


Figure 3.20. 16x16 hybrid Wallace Multiplier [43]

3.3 ABACUS MULTIPLIER

ABACUS architecture proposed by A. Muhtaroglu [76] evaluates all partial products with the same rank at the same time by constructing a heap structure of partial products. Each column, after the heap structure of partial products, is processed by using saturated and unsaturated counters. The major benefit of ABACUS is the ability to process additional bits using wider compression units and thus reduces the number of compression stages by pushing carries more towards significant columns on the left half. Originally proposed ABACUS algorithm recommends the removal of logic zeros (bubbles) sandwiched between one's partial product columns in compression stages since they do not contribute final adder. After removing bubbles, threshold level or a maximum number of 1's in each column can be detected and an equivalent amount of carries signals passed on to left columns.

Removing bubbles requires extra hardware to filter each column by inspecting every bit, which is not proficient. Gurdur [132] worked on the high-level implementation of ABACUS and found that 20% improvement is required in the middle column of ABACUS to make it as good as Wallace multiplier based on (3,2) counter. F.Ercan [24], primarily focused on threshold detection techniques used in ABACUS, implemented 8x8 bits ABACUS architecture (depicted in the Figure 3.21) and compared with conventional (3,2) counter based Wallace multiplier in complementary CMOS logic. To process more than three bits, unsaturated counters similar to the ones proposed by Dandapat [118] mentioned earlier were used in the F.Ercan implementation. Pre-layout simulations reported by F.Ercan shows that ABACUS has 1.66% smaller delay, 8.93% smaller average power, 23% low activity and 4% high static power. The potential advantage of ABACUS architecture is its ability to push carry bits to multiple levels at once, based on the threshold level of each column.

Wider unsaturated SCI counters composed of (2,2) and (3,2) counters in ABACUS multiplier are able to process extra bits, but results into complex hardware overhead and expected to increase average current and PDP of overall multipliers. With the penalty of unsaturated counters in the compression process, compression stages in ABACUS architecture drastically reduce, for instance, 8x8 bit multiplier compression accomplished in three stages compared to four in Wallace multiplier based on (3,2) counter (Figure 3.1). Scaled description of the ABACUS i.e. 10x10, 12x12 and 16x16 are shown in Figure 3.22 to Figure 3.24.

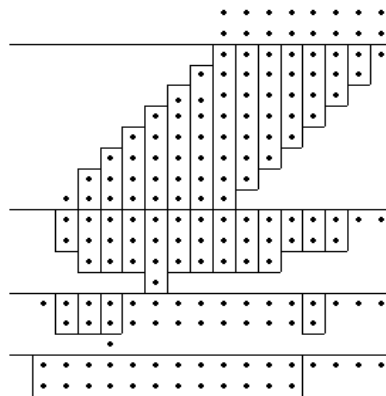


Figure 3.21. 8x8 ABACUS Multiplier [24]

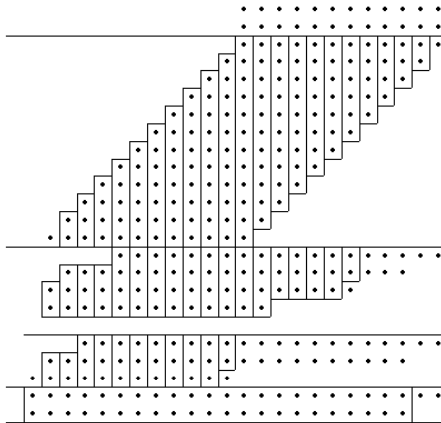


Figure 3.22. 10x10 ABACUS Multiplier

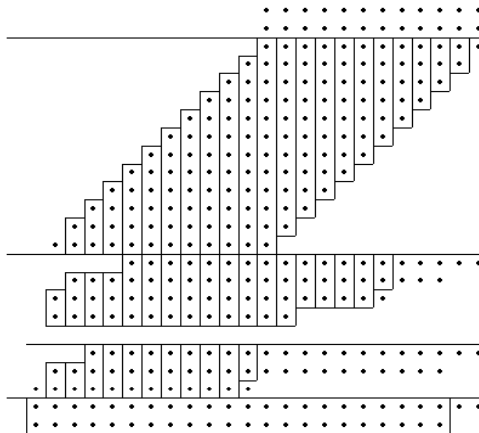


Figure 3.23. 12x12 ABACUS Multiplier

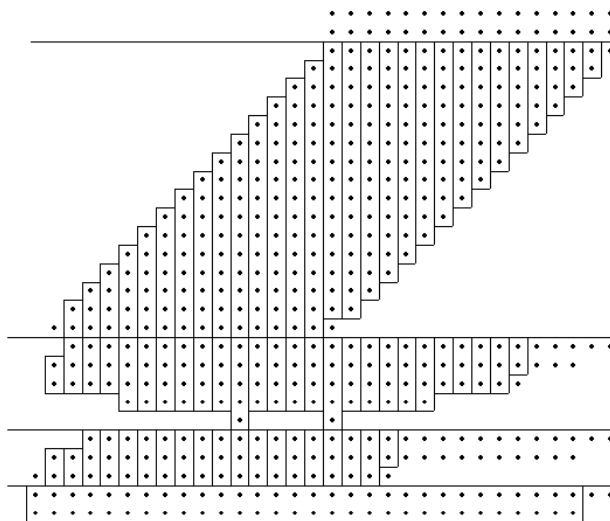


Figure 3.24. 16x16 ABACUS Multiplier

3.4 CONFIGURATION OF MULTIPLIER ARCHITECTURES

Table 2 depicts the configuration of the 8-bit multipliers. For both compression and final addition, the numbers of compression blocks, stages and logic gate have been mentioned. For compression stage, Wallace based on (3,2) counter and ABACUS multipliers have a maximum number of (2,2) counter or AND gates, which degrades their performances. The effects of AND gates on multiplier performance will be discussed in detail in Chapter 4.

Table 2 Configuration of 8x8 multiplier

	Wallace Multiplier				Hybrid Wallace	ABACUS	
	(3,2) count	4:2 comp	5:2 comp	6:2 comp			
Compression Stage	(2,2) counters	15	7	4	5	15	9
	(3,2) counters	38	7	3	4	1	10
	(2,3,3) counters	-	-	-	-	13	-
	(4,3) counters	-	-	-	-	-	3
	4:2 compressor	-	17	4	8	-	-
	5:2 compressor	-	-	10	2	-	-
	(5,3) counters	-	-	-	-	-	2
	(6,3) counters	-	-	-	-	-	2
	6:2 compressor	-	-	-	4	-	-
	(7,3) counters	-	-	-	-	3	2
	(8,4) counters	-	-	-	-	-	1
	No. of stages	4	2	2	2	3	3
	RCA final adder	MUX	38	41	41	41	39
XOR		91	89	86	87	93	94
AND		15	7	4	5	15	24
Inverters		212	158	132	136	160	138
Adder Size (bits)		11	13	13	13	12	13
(2,2) counters		1	1	1	1	4	1
(3,2) counters		9	11	12	12	7	11
MUX		9	11	12	12	7	11
XOR	20	24	25	25	20	25	
AND	1	1	1	1	4	1	
Inverters	42	50	54	54	46	50	

In final addition, all multiplier have only one (2,2) counter at the beginning, except hybrid Wallace which has four (2,2) counter in the final adder (Table 2). Final adder size is inversely proportional to the number of stages, for instance in Table 2, Wallace based

on (3,2) counter has the most number of stages and least size of the final adder, (vice versa for wide compressors based architectures).

Table 3 depicts the configuration of the 10-Bit multipliers. It can be observed that the number of XOR and MUX gates is nearly same for all multipliers, while AND gate varies w.r.t. the number of (2,2) counter or unsaturated counters. ABACUS multiplier, for instance, has the most number of unsaturated blocks, which led to thirty-seven AND gates in compression stage. In addition, inverters in compression stages are inversely proportional to the size of compression blocks used. ABACUS has the least number of restoring inverters (182) because wide unsaturated compression blocks have been used.

Table 3 Configuration of 10x10 multiplier

	Wallace				Hybrid Wallace	ABACUS	
	(3,2) count	4:2 comp	5:2 comp	6:2 comp			
Compression Stage	(2,2) counters	25	13	9	2	16	2
	(3,2) counters	67	8	7	3	3	14
	(2,3,3) counters	-	-	-	-	20	-
	(4,3) counters	-	-	-	-	-	7
	4:2 compressor	-	32	11	4	-	-
	5:2 compressor	-	-	14	4	-	-
	(5,3) counters	-	-	-	-	-	2
	(6,3) counters	-	-	-	-	-	2
	6:2 compressor	-	-	-	12	-	-
	(7,3) counters	-	-	-	-	7	2
	(8,4) counters	-	-	-	-	-	2
	(9,4) counters	-	-	-	-	-	2
	(10,4) counters	-	-	-	-	-	1
	No. of stages	5	3	2	2	4	3
RCA final adder	MUX	67	72	71	71	71	63
	XOR	159	157	151	144	158	163
	AND	25	13	9	2	16	37
	Inverters	368	276	242	196	238	182
	Adder Size (bits)	14	16	17	17	16	16
	(2,2) counters	1	1	1	1	6	0
	(3,2) counters	12	14	15	15	8	15
	MUX	12	14	15	15	8	15
	XOR	26	30	32	32	23	32
	AND	1	1	1	1	6	0
Inverters	54	62	66	66	58	62	

Similarly, Table 4 and Table 5 depict the configuration of 12-bit and 16-bit multipliers. The differences in a number of inverters AND gates and size of the multiplier are more prominent in this size of multipliers. For instance, 16-bit multipliers, ABACUS has 418 inverters, compared to 1008 in Wallace multiplier based on (3,2) counter. Similarly, a number of stages in Wallace multipliers based on (3,2) counters is double, compared to regular implementation. On the other hand, ABACUS has the largest amount of AND gates in the 16-bit multiplier, i.e. ninety-two compared to nineteen in Wallace implementation based on 6:2 compressor.

Table 4 Configuration of 12x12 multiplier

	Wallace Multiplier				Hybrid Wallace	ABACUS	
	(3,2) count	4:2 comp	5:2 comp	6:2 comp			
Compression Stage	(2,2) counters	34	15	15	11	21	8
	(3,2) counters	102	12	17	7	3	17
	(2,3,3) counters	-	-	-	-	27	-
	(4,3) counters	-	-	-	-	-	11
	4:2 compressor	-	49	12	13	-	-
	5:2 compressor	-	-	24	4	-	-
	(5,3) counters	-	-	-	-	-	2
	(6,3) counters	-	-	-	-	-	2
	6:2 compressor	-	-	-	16	-	-
	(7,3) counters	-	-	-	-	12	2
	(8,4) counters	-	-	-	-	-	2
	(9,4) counters	-	-	-	-	-	2
	(10,4) counters	-	-	-	-	-	2
	(11,4) counters	-	-	-	-	-	2
(12,4) counters	-	-	-	-	-	1	
No. of stages	5	3	3	2	4	2	
MUX	102	110	113	111	105	98	
XOR	238	235	241	233	231	263	
AND	34	15	15	11	21	67	
Inverters	544	402	392	350	330	274	
RCA final adder	Adder Size (bits)	18	20	20	21	19	21
	(2,2) counters	1	1	1	1	4	0
	(3,2) counters	16	18	18	20	14	20
	MUX	16	18	18	17	14	20
	XOR	35	38	38	36	33	42
	AND	1	1	1	1	4	0
	Inverters	70	78	78	74	74	82

Table 5. Configuration of 16x16 multiplier

	Wallace				Hybrid Wallace	ABACUS	
	(3,2) count	4:2 comp	5:2 comp	6:2 comp			
Compression Stage	(2,2) counters	52	23	17	19	38	8
	(3,2) counters	200	20	11	7	5	25
	(2,3,3) counters	-	-	-	-	42	-
	(4,3) counters	-	-	-	-	-	18
	4:2 compressor	-	95	29	24	-	-
	5:2 compressor	-	-	49	11	-	-
	(5,3) counters	-	-	-	-	-	3
	(6,3) counters	-	-	-	-	-	2
	6:2 compressor	-	-	-	32	-	-
	(7,3) counters	-	-	-	-	28	2
	(8,4) counters	-	-	-	-	-	2
	(9,4) counters	-	-	-	-	-	2
	(10,4) counters	-	-	-	-	-	2
	(11,4) counters	-	-	-	-	-	2
	(12,4) counters	-	-	-	-	-	2
	(13,4) counters	-	-	-	-	-	2
	(14,4) counters	-	-	-	-	-	2
	(15,4) counters	-	-	-	-	-	3
No. of stages	6	3	3	3	4	3	
MUX	200	210	216	216	201	194	
XOR	452	443	449	451	440	480	
AND	52	23	17	19	38	92	
Inverters	1008	742	678	656	592	418	
RCA final adder	Adder Size (bits)	25	28	28	28	27	29
	(2,2) counters	1	1	1	1	5	0
	(3,2) counters	23	26	26	26	21	28
	MUX	23	26	26	26	21	28
	XOR	48	55	54	54	49	58
	AND	1	1	1	1	5	0
	Inverters	98	110	110	110	106	114

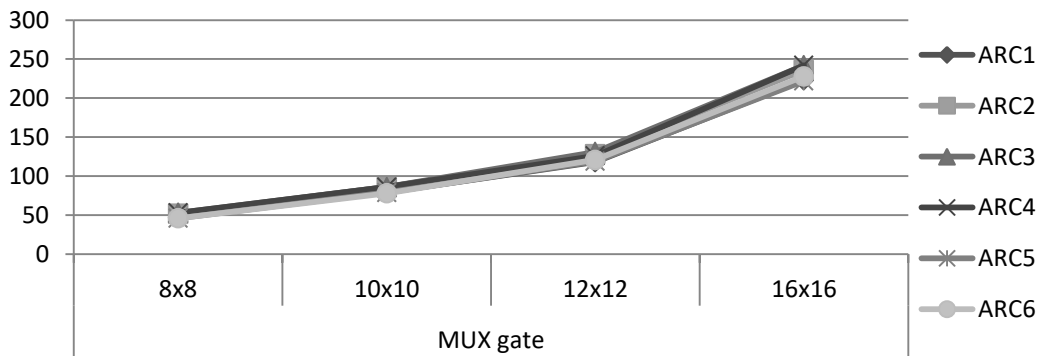
3.5 CHAPTER SUMMARY

For simplicity now and onwards, we have used the abbreviation for multipliers architecture mentioned in Table 6. A total number of MUX gate, XOR gate and inverters in multipliers for different sizes are depicted in Figure 3.25.

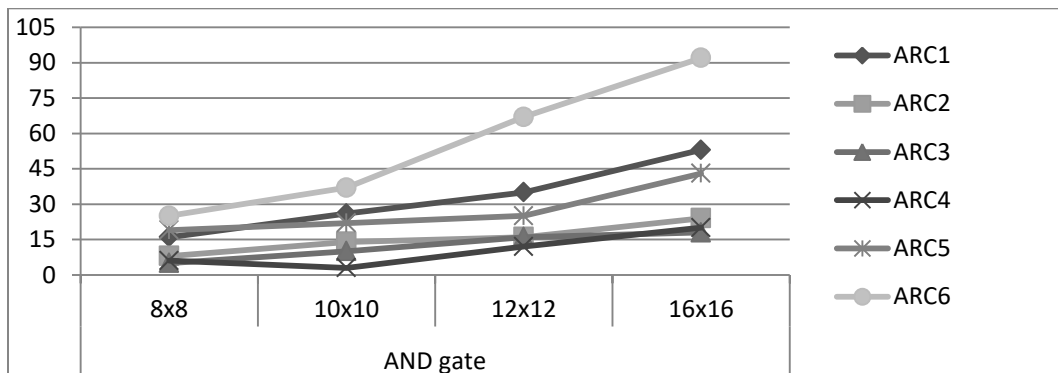
Table 6. Abbreviation of multiplier architecture

(3,2) counter based Wallace	ARC1
4:2 compressor based Wallace	ARC2
5:2 compressor based Wallace	ARC3
6:2 compressor based Wallace	ARC4
Hybrid Wallace	ARC5
ABACUS	ARC6

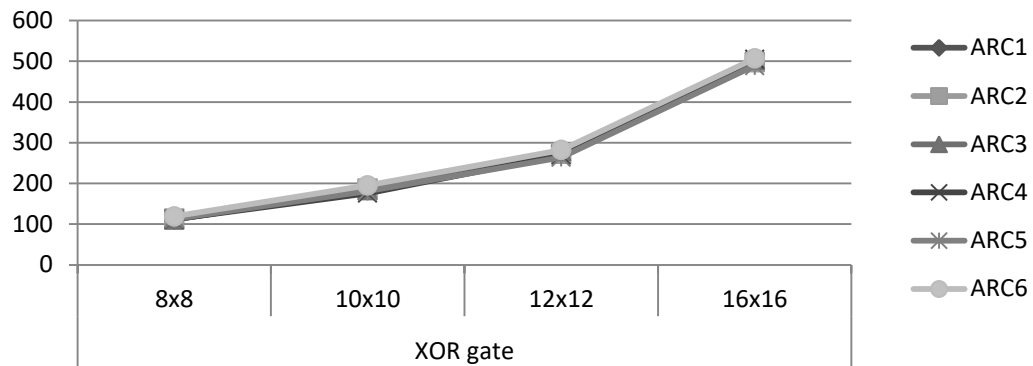
Multiplexers and XOR gates are nearly same for all sizes, which shows that for a given architecture, placement and activity are essential, drives the average current and PDP of multipliers. AND gate also varies in irregular manner w.r.t number of (2,2) or unsaturated counters. Inverter's growth rate is considerably large compared to logic gates.



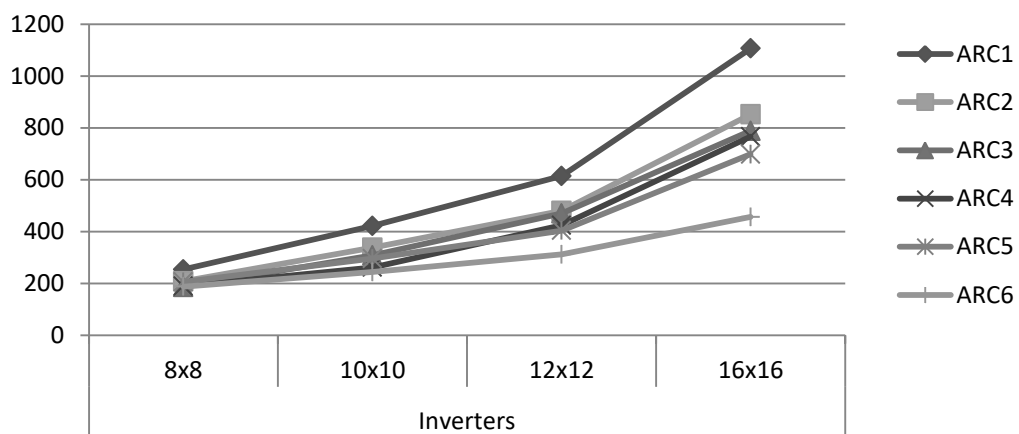
a. MUX gate w.r.t. multiplier size



b. AND gate w.r.t. multiplier size



c. XOR gate w.r.t. multiplier size



d. Inverters w.r.t. multiplier size

Figure 3.25. Trends of logic blocks for different sizes of multipliers

CHAPTER 4
ANALYSIS OF COMPRESSION BLOCKS FOR ENERGY
AWARE MULTIPLIERS

In this chapter, different types of saturated and unsaturated counters and compressors have been discussed and compared theoretically, and later through simulation. Results grounded on proposed evaluation metrics are compared and discussed with the existing metric. The trends observed from the new metric revise the finding based on existing metrics and helps to objectively forecast the effectiveness of wider counters and compressors. Latterly, a comprehensive table of counters and compressors have been generated, which includes: simulated worst-case delay, number of bits compressed with equivalent decimal value, depth of counter or compressors and also compression ratio (C_r) and compression ratio per worst gate delays (c_R). The C_r , c_R , and normalization of compression ratio w.r.t PDP in graphs at the end of this chapter summarizes the discussion on compression circuits.

4.1 INTRODUCTION

DPL introduce by M. Suzuki et al. [22] has been used in this study for circuit realization, as mentioned before. DPL has high average current but PDP is relatively low compared to other logic styles [20]. In addition, efficient implementation of XOR gates in DPL is particularly important for multipliers circuits [10], since it has minimum skew delays with full-swing voltage. Also, authors in [110] showed that PTL based circuits are better than complementary logic for complex circuits like multipliers. Compression blocks used in DPL based VLSI multipliers essentially required AND, MUX and XOR gates for realization. DPL implementation of these logic gates is depicted in Figure 4.1, Figure 4.2 and Figure 4.3 respectively. MUX and XOR gates have the same structure with a same number of devices, while only difference is the way they are a connection.

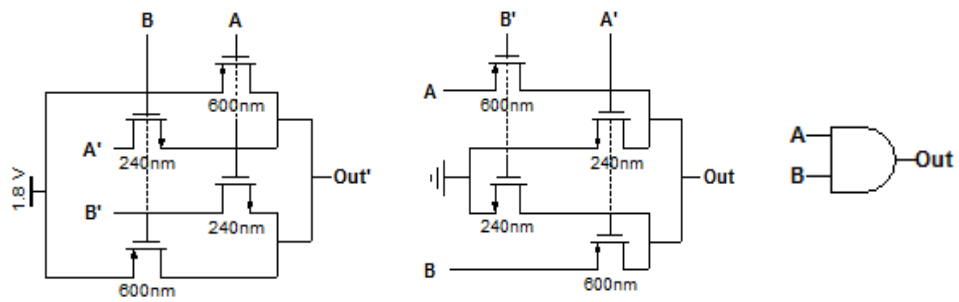


Figure 4.1. AND implementation in DPL (left) and symbol (right).

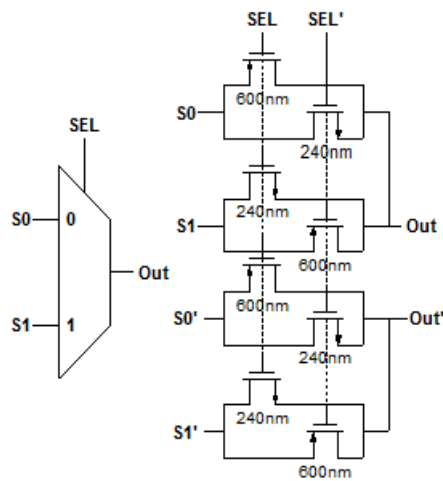


Figure 4.2. MUX implementation in DPL: symbol (left) and gate level implementation (right)

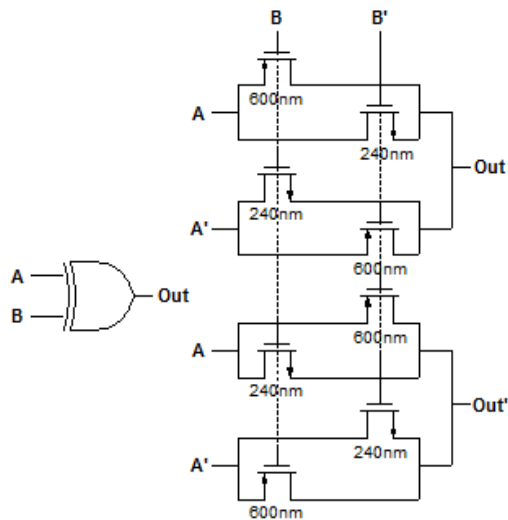


Figure 4.3. XOR implementation in DPL: symbol (left) and gate level implementation (right).

The AND gate in DPL not only causes high leakage currents as one of the input is permanently connected to ground or V_{DD} (Figure 4.1), but also generates a path for short circuit. Therefore, if the connected signals to associate gates have poor voltage swing, short-circuits will drastically decrease the performance of AND gate and respective compression circuits. But the AND-gates are inevitable, especially in half-adder or (2,2) counters when only two bits are left in a row for processing.

4.2 CHARACTERIZING COMPRESSION CIRCUITS

A number of gate delays varies with respect to outputs of a compression circuit and depends on depth of counter. For instance, in (7,3) counter (Figure 2.8), the worst path of LSB output (2^0) has four gate delays, while (2^1) and (2^2) have six gate delays. Similarly (15,4) counter (Figure 2.9) has six gate delay in (2^0), eight gate delays for (2^1) and ten gate delays for (2^2) and (2^3) respectively, the. For wide saturated SCI counters, generalize equation (4.1) has been constructed to calculate depth or stages of the counter.

$$\text{Number of stages for 'n' bit SCI saturated counter} = \log_2(n + 1) - 1 \quad (4.1)$$

Not all input bits of a counter processed in the first stage, for instance, in (15,4) counter three bits remains unprocessed intentionally for next stages, although these can be processed using a (3,2) counter. The last stage always has one new bit and followed by have two, four and eight, *unprocessed* or *new* bits respectively. Hence, we can quantify processed and unprocessed bits in each stage of 'n' bit counter by given equations (4.2) to (4.5).

$$\text{Number of unprocessed bits left after the first stage} = N_1 = \sum_{x=0}^{\log_2(n+1)-3} 2^x \quad (4.2)$$

$$\text{Number of bits processed in the first stage} = n - N_1 \quad (4.3)$$

$$\text{Number of (3,2) counters in first stage} = \frac{n - N_1}{3} \quad (4.4)$$

Applying equations (4.2) to (4.5) to (7,3) counter (Figure 2.8): For the first stage, the upper limit of summation becomes; $\log_2(8) - 3 = 0$ which gives one unprocessed bits ($2^0 = 1$) to second stage, while six bits processed at first stage, using two (3,2) counters. Applying similar approach to (15,4) counter, upper limit becomes one for first stage and number of unprocessed bits are three, while number of bits processed in first stage are ($15 - 3 = 12$), as depicted in Figure 2.9. Generalizing this for z^{th} stage of 'n' bit saturated SCI counter following recursive equations have been formulated (for $z! >$ No. of input bits):

$$\text{Unprocessed bits left after } z^{\text{th}} \text{ stage} = N_z = \sum_{x=0}^{(\log_2(n+1) - z - 2)} 2^x \quad (4.5)$$

$$\text{New bits processed in } z^{\text{th}} \text{ stage} = (N_{z-1} - N_z) \quad (4.6)$$

$$\text{Bits processed in } z^{\text{th}} \text{ stage} = (N_{z-1} - N_z) + \frac{\# \text{ of } (3,2) \text{ counters in } (z-1) \text{ stage}}{2} \quad (4.7)$$

For 'n' the position of the column, the height of partial product bits and their value has been calculated by [113] as:

$$\text{Height}_{\text{ith}} = 2 \times \text{floor} \left(\frac{i}{m} \right) + 1 \quad \text{For } i = 0 \text{ to } n-1 \quad (4.8)$$

$$\text{Height}_{\text{ith}} = 2 \times \text{floor} \left(\frac{2n-1-i}{m} \right) + 1 \quad \text{For } i = n \text{ to } 2n-1 \quad (4.9)$$

$$\text{Value of output} = \sum_{i=0}^{n-1} \text{Binary. Weight of column}_i \quad (4.10)$$

For cascaded MCI saturated counter, the height of next stage can be given by [113] as:

$$\text{Height of following stage} = \text{floor} \left(\frac{\text{No. of outputs of MCI counter}}{\text{No. of input columns}} \right) \quad (4.11)$$

If 'x' number of columns, a number of bits in each column is c_i , having the weight 2^i , and 'n' is a number of outputs then the value of output can be calculated by (4.12) given by [113]:

$$\text{Value of output} = \sum_{i=0}^{m-1} \sum_{j=0}^{c_i-1} \text{Binary. Weight of column}_{ij} 2^i \quad (4.12)$$

4.3 THEORETICAL COMPARISON OF COMPRESSION CIRCUITS

Compressor ratio (C_r) of counters or compressors is the ratio of the input bits to the output bits. However, evaluating compression circuits based on only compression ratio is not meaningful, since they have different gate delay on the worst path and depth. Therefore, authors in [54] introduce compression ratio per gate delays (c_r), which normalize ' C_r ' by a total number of gates in the critical path (WGD). Evaluating compression circuit by ' c_r ' covers not only compression capability, but also its delay. ' C_r ' and ' c_r ' of counters and compressors discussed in Chapter 2 are depicted in Table 7. In parallel with compression ratio and normalized compression ratio, depth of compression circuits and bits equivalent in binary have also mentioned in Table 7. It is worth to note that at output of each counter or compressor circuits, a unit inverter stage with aspect ratio of (2.5/1) has been deployed to refresh both inverting and non-inverting signals and also to ensure full voltage swing. Graphical description of Table 7 is depicted in Figure 4.4. The observations from Figure 4.4 are listed below:

- HA or the (2,2) counter has minimum ' C_r ' and maximum ' c_r '. This counter, therefore, serves as a reference for compression blocks.. Compression circuits performs well if its ' c_r ' is close to HA, whereas it performs worse if ' C_r ' value is close to HA.
- For SCI, MCI counters and compressors, compression ratio ' C_r ' increases with the size of compression block, but C_r /WGD or ' c_r ' decreases.
- For compressors and MCI saturated counters in Figure 4.4, the decay in ' c_r ' is linear w.r.t. size, while in SCI counter ' c_r ' decreases irregularly.
- ' c_r ' curve in MCI counters is steeper compared to compressors, which show that wider MCI counters are worse than wider compressors.
- Unsaturated counters following saturated SCI counter shows better performance, than unsaturated counters next to saturated SCI counter.

Table 7. Configuration of counters and compressors with their 'C_R' and 'c_R'

Counter/ compressor type	Compression ratio (C _r)	C _R /Worst gates delays (c _R)	Value of outputs in decimal	Depth/ No. of stages	Weight of output	Number of gates on Worst path
Saturated SCI Counters						
(3,2) counter	1.5	0.75	3	1	2 ⁰	2
					2 ¹	2
(7,3) counter	2.33	0.3883	7	2	2 ⁰	4
					2 ¹	6
					2 ²	6
(15,4) counter	3.75	0.375	15	3	2 ⁰	6
					2 ¹	8
					2 ²	10
					2 ³	10
Saturated Compressors						
4:2 compressor	2	0.66	4	3	2 ⁰	3
					2 ¹	3
					C _{out}	2
5:2 compressor	2.5	0.625	5	5	2 ⁰	4
					2 ¹	4
					C _{out#1}	2
					C _{out#2}	3
6:2 compressor	3	0.6	6	5	2 ⁰	5
					2 ¹	5
					C _{out#1}	2
					C _{out#2}	3
					C _{out#3}	4
MCI Saturated Counters						
(2,3,3) counter	1.66	0.556	7	2	2 ⁰	2
					2 ¹	3
					2 ²	3
(5,5,4) counter	2.5	0.416	15	4	2 ⁰	3
					2 ¹	5
					2 ²	6
					2 ³	6
(2,2,2,3,5) counter	1.8	0.3	31	2	2 ⁰	2
					2 ¹	3
					2 ²	4
					2 ³	5
					2 ⁴	6

SCI unsaturated Counters						
(2,2) counter	1	1	2	1	2^0	1
					2^1	1
(4,3) counter	1.33	0.33	4	2	2^0	2
					2^1	4
					22	4
(5,3) counter	1.66	0.33	5	2	2^0	3
					2^1	5
					22	5
(6,3) counter	2	0.4	6	2	2^0	3
					2^1	5
					2^2	5
(8,4) counter	2	0.285	8	3	2^0	3
					2^1	5
					2^2	7
					2^3	7
(9,4) counter	2.25	0.281	9	3	2^0	4
					2^1	6
					2^2	8
					2^3	8
(10,4) counter	2.5	0.3125	10	3	2^0	4
					2^1	6
					2^2	8
					2^3	8
(11,4) counter	2.75	0.34375	11	3	2^0	4
					2^1	6
					2^2	8
					2^3	8
(12,4) counter	3	0.375	12	3	2^0	4
					2^1	6
					2^2	8
					2^3	8
(13,4) counter	3.25	0.3611	13	3	2^0	5
					2^1	7
					2^2	9
					2^3	9
(14,4) counter	3.5	0.3888	14	3	2^0	5
					2^1	7
					2^2	9
					2^3	9

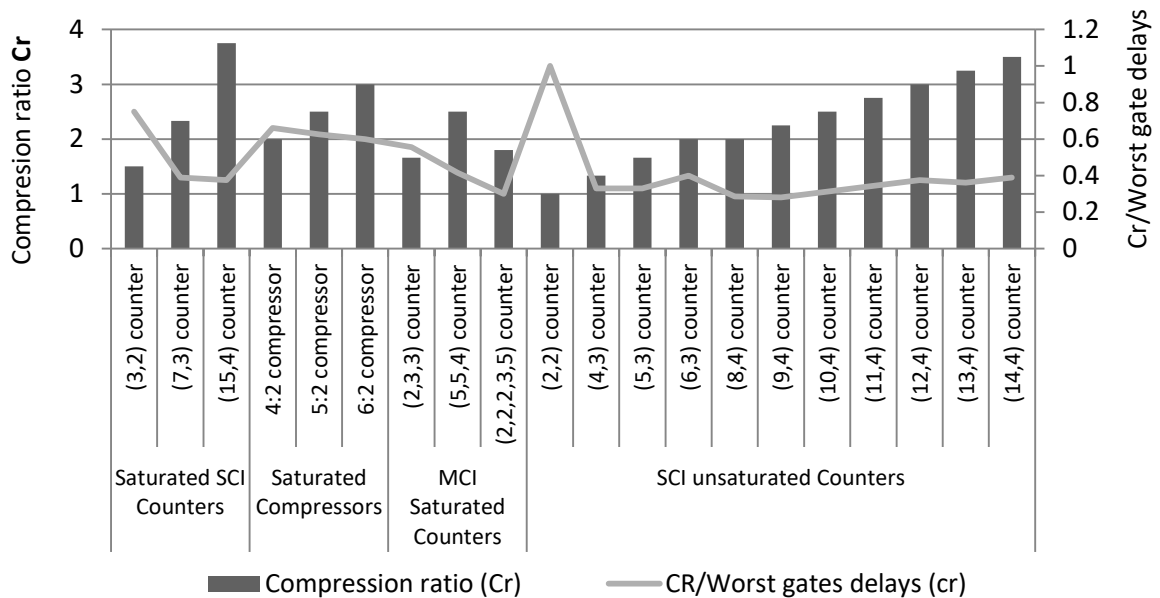


Figure 4.4. ‘C_R’ and ‘c_R’ of counter and compressor circuits

Trends of compression ratio, normalized compression ratio w.r.t. gate delays and a total number of gates are depicted in Figure 4.5. These trends shows that as the size increases normalization w.r.t. WGD first decreases then increase, contrary to the normalization w.r.t. a total number of gates, which gradually decreases. The normalization w.r.t total number of gates is more realistic as it accounts switching effect of gates and leakage that is significant in wide compression circuits, and implicitly includes gate delays. The effect of a number of gates on performance will be discussed in the next section.

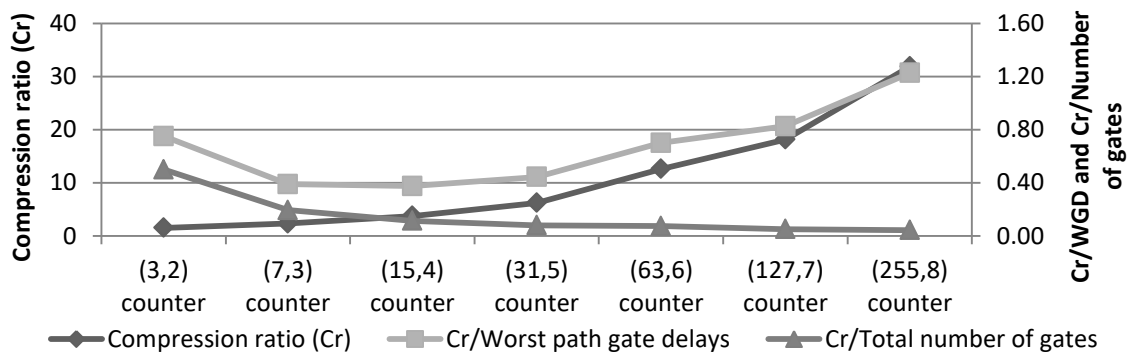


Figure 4.5. Compression ratio and Normalized compression ratio w.r.t worst gate delays and total number of gate delays, for saturated counters

4.4 SIMULATION COMPARISON OF COMPRESSION CIRCUITS

This section aims to discuss compression ratio ' C_R ' and its normalization w.r.t. simulated power delay product (PDP). For a fair PDP calculation of compression circuits, two loading scenarios have been designed to stress the compression circuits systematically through all possible input vectors: i) When the most active bit is the most left bit. ii) When the most active bit is the most right bit. All simulations in this study are completed using Cadence tool in 180nm technology. Supply voltage and temperature represent nominal condition at 1.8 V and 27°C respectively. The edge rate of all inputs is 20ps/V with most active input switching at a rate close to system maximum frequency limit i.e. 250 MHz. To provide more realistic slew rates, two unit buffers have been installed at inputs while every output of compression circuit under test has been loaded with 5 fF of capacitance, as depicted in Figure 4.6. For an example the circuit under test (CUT) is (3,2) counter then input sweeps all possible vectors first from 000 to 111 and then 111 to 000. By doing so not only gives more refined power consumption values but also have more deterministic coverage of worst delay of circuits.

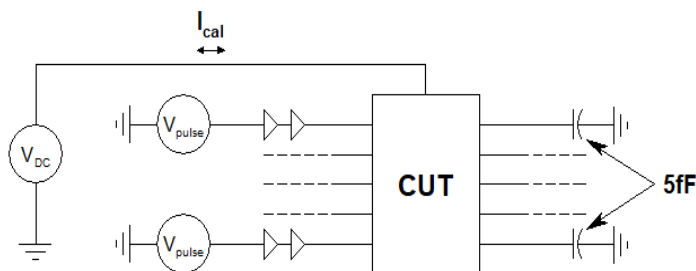


Figure 4.6. Test bench for compression circuit

4.4.1 EXISTING AND PROPOSED METRIC FOR EVALUATION OF COMPRESSION CIRCUITS

Although the delay of circuit C_R/WGD is an important metric as it covers compression ratio and gate delay simultaneously, power variation with respect to application is also significant.. For example, in low activity circuit applications such as the sensors in IoT applications, power rarely fluctuates, while in highly active systems, i.e. data rendering and image processing, where dynamic power is very large. Hence, the selection of

counters and compressors based only on ' C_R/WGD ' will not fully serve the purpose of optimization. This study, therefore, proposes another metric; ' C_R/PDP ', which implicitly covers worst delay of the compression blocks as well as the power consumed by it.

Ideally, compression circuits have high compression ratio (C_R) with minimum PDP. Compression ratio (C_R) and normalized compression ratio (C_R/PDP) of SCI counters are shown in Figure 4.7. Similar to C_R/WGD in Figure 4.4, C_R/PDP also drops first from saturated to unsaturated, and then gains its value. However, C_R/PDP recovers *more linearly* to next SCI saturated counter, compared to C_R/WGD . For instance, C_R/WGD in Figure 4.4 indicates that saturated (7,3) counter has poorer performance than (6,3) unsaturated counter, which is corrected by C_R/PDP in Figure 4.7.

Same is the case with (12,4) and (13,4) counters. Furthermore (5,4) counter is more efficient in terms of C_R/PDP compared to (4,3) unsaturated counters in Figure 4.7 contrary to the prediction of C_R/WGD . C_R/PDP drops after each saturated counter, but after (7,3) counter, it recovers gradually compared to the trend following (7,3) counter. This indicates that PDP improves more rapidly from saturated (3,2) to (7,3) counters compared to saturated (7,3) to (15,4) counters. This can be explained by examining the number of gates used in SCI saturated and unsaturated counters depicted in Figure 4.8.

The AND gate as mentioned before, due to short circuit current and permanent connection with supply or ground, drastically effects the PDP of multipliers. . Figure 4.8. depicts that the number of AND gates abruptly changes after saturated to the unsaturated counter circuit and then linearly decrease. Whereas, XOR or MUX gate linearly varies the w.r.t counter size. For *small counters* (smaller than (7,3) counter), the effect of short circuit current dominates, so C_R/PDP grows linearly with the linear decrease in AND gates.

In *large counters* (larger than (7,3) counter), the switching of XOR or MUX gates is high enough such that activity of node capacitance dominates the short-circuit current and PDP shows the intermediate effect of both *activity* and *short circuit currents*. Figure 4.4 indicates that wider saturated counters i.e. (15,4), will perform equally well in terms

of C_R/WGD as saturated (7,3) counters, while simulation results and C_R/PDP in Figure 4.7 showed that wider saturated counters have a relatively high PDP penalty for the delivered high compression ratio.

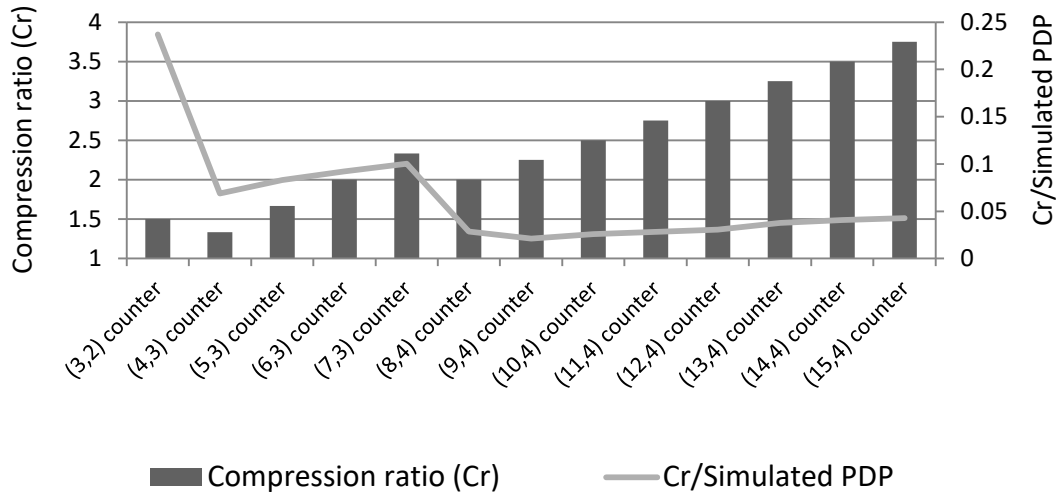


Figure 4.7. C_R and normalized C_R w.r.t simulated PDP of single column input saturated and unsaturated counters

4.5 CHAPTER SUMMARY

Figure 4.9 depicts the C_R/PDP of MCI counter and compressor circuits. MCI saturated (2,3,3) counter is the best option for multiplier based on C_R/WGD (in Figure 4.4), whereas C_R/PDP metric in Figure 4.9 corrects this as 4:2 and 5:2 compressors are more efficient than (2,3,3) counter for energy-aware systems. C_R/WGD decreases linearly in Figure 4.4, whereas a sharp decrease in C_R/PDP can be observed in MCI counters Figure 4.9. This supports our general conclusion that wider saturated compression circuits perform worse than smaller saturated counters and compressors. The new metric, therefore, helps the designer to compare and choose the most appropriate compression block for different applications. A summary of corrections done by new evaluation metric (C_R/PDP) is shown Table 8.

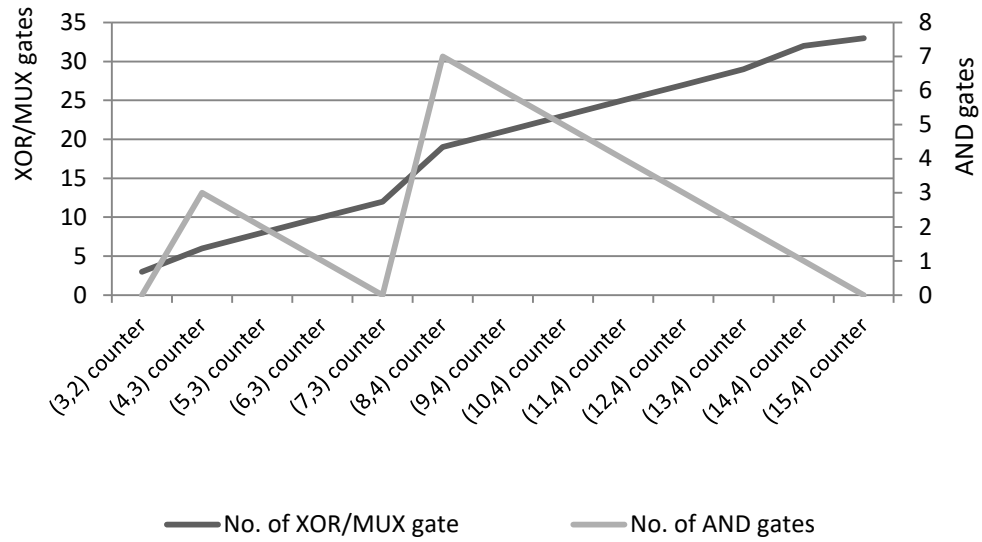


Figure 4.8. XOR/MUX gates and AND-gates in single column input counters

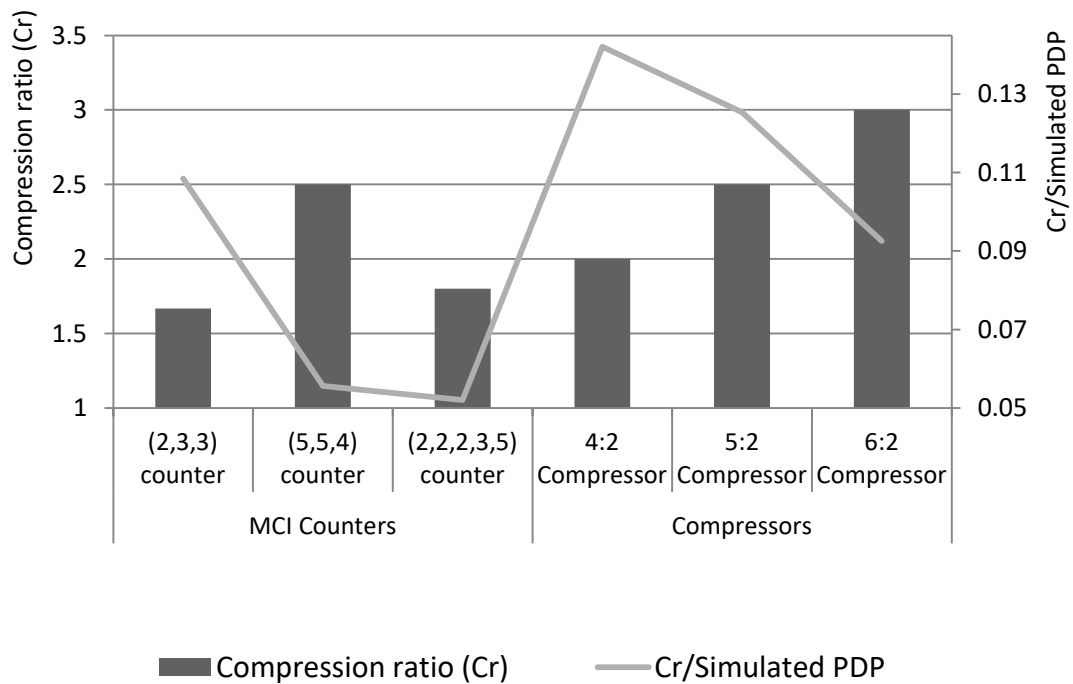


Figure 4.9. CR and normalized CR w.r.t simulated PDP of single column input saturated and unsaturated counters

Table 8. Correction and characterization of compression circuits with new metric:
C_R/PDP

C _R /WGD (Existing metric)	C _R /PDP (Proposed Metric)
(2,3,3) counter is 25% , 49% and 66% better than 4:2,5:2 and 6:2 compressors respectively.	4:2 and 5:2 compressor are 24% and 13.6% better than (2,3,3) counter respectively, while 6:2 compressor is only 16.1% worse than (2,3,3) counters
(7,3) counter is 14.5% and 28.5% superior than 4:2 and 5:2 compressor	4:2 and 5:2 compressor are 29.5% and 20.1% better than (7,3) counter respectively.
4:2 compressor is 17% and 25% and superior to 5:2 and 6:2 compressor	4:2 compressor is 12% and 34% and superior to 5:2 and 6:2 compressor
(2,3,3) counter is 25% and 46% superior than (5,5,4) and (2,2,2,3,5) counter respectively.	(2,3,3) counter is 48.7% and 52% superior than (5,5,4) and (2,2,2,3,5) counter respectively

CHAPTER 5

ANALYSIS OF MULTIPLIERS FOR ENERGY AWARE APPLICATIONS

In this chapter, simulation results of multiplication architecture discussed in Chapter 3 have been presented. This chapter also includes details of simulation setup, proposed evaluation approach, number of input vectors and detailed discussion on worst delay. In addition, the calculation to measure PDP, average current, leakage and power delay product have also presented before commenting on trends of different multiplier results. A new System Compression Energy Efficiency Score (SCEES) has been proposed at the end of this chapter, which estimates PDP of multiplier based on the individual performance of compression blocks indicated in previous chapter, without thorough simulation of multiplier architectures. Results in the summary of this chapter give a pictorial description of the key metrics i.e. PDP and EDP, for all sizes of multipliers (8-bits, 10-bits, 12-bits, 16-bits).

5.1 EVALUATION METHOD OF MULTIPLIER ARCHITECTURES

Evaluation methodology is a key factor for the validity of the approach, hence specifically mentioned in the review Table 1. Size of the multiplier under test, which depicts extent or applicability of suggested approach, is also crucial. Lastly, the process parameters having a strong influence on circuit behavior also depicted to give a complete overview on parallel bit multipliers. Area power, and delay are three major constraints for VLSI circuits [133]. The area and power dissipation of multipliers increase roughly N^4 for N-bit multiplier, whereas delay is logarithmic to input size [134]. Literature is not consistent with the method of testing these key constraints of multipliers. Classical multiplier analysis is based on randomly generated vectors by MATLAB and simulated in EDA tools (Cadence or Synopsys) [20], [43]. Random vectors work fine for small cells or architectures where input combinations are not very

large and activity has a minor impact on average current or power. However, for large multipliers, random generation is dubious as it is hard to differentiate activity impact and there is always an ambiguity about the worst-case delay of the multiplier. Therefore, the adequate analysis is extremely important to validate a particular design approach.

In this study, we have proposed a systematic approach of input vectors generation. Test bench setup for this healthy evaluation is shown in Figure 5.1. Each size of the multiplier has been evaluated:

- i) When all input bits of multiplier tied to logic 1, and multiplicand sweeps for all combination and
- ii) When all input bits of multiplicand tied to logic 1, and multiplier sweeps for all combination.

For 8x8, 10x10 and 12x12, each of these scenarios are further classified as:

- a. The most active bit is on most left of multiplier circuit under test.
- b. The most active bit is on most right of multiplier circuit under test.

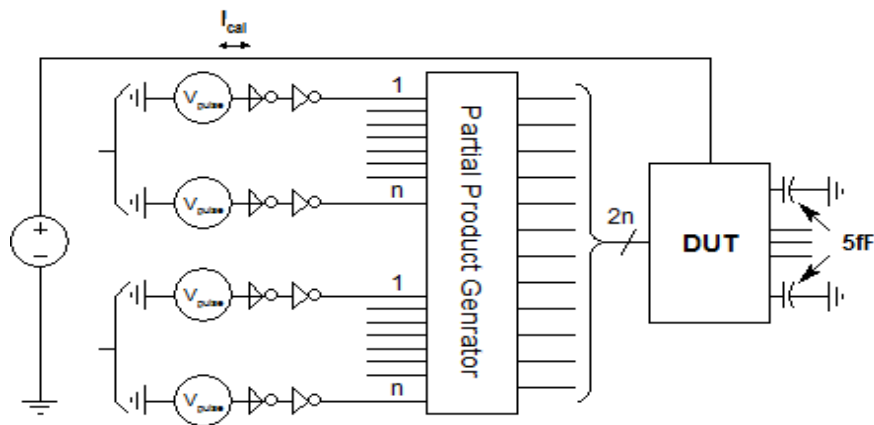


Figure 5.1. Test bench for multipliers

By simulating this bulk amount of vectors, we can ensure worst delay and refines the average current of multiplier architectures. This approach of evaluation results into 4×2^8 vectors out of 2^{16} vectors for the 8x8 multiplier, 4×2^{10} vectors out of 2^{20} vectors for 10x10 multipliers and 4×2^{12} vectors out of 2^{24} vectors for 12x12 multipliers. For 16x16, 2×10^{16} vectors have been simulated due to the limited memory capacity of the

system. Therefore, a total number of simulated vectors for the 16-bit multiplier is $2 \times (2^{16}) = 131,072$. These vector sets have better and more deterministic coverage for the evaluation of multiplier architectures compared to random vector sets previously used in the literature as described in Table 9.

Table 9. Evaluation Criteria for Multipliers in Literature

Multiplier	Authors	Random vectors
16x16	[36]	1,200
	[47]	10,000
32x32	[49]	3,000
	[13]	3,000
54x54	[33]	20,000
	[17]	10,000
	[58]	2,000

5.2 WORST DELAY IN MULTIPLIERS

One of the major goals of compression stage is to decrease critical or the worst path to fewer gate delays without violating regularity of topology [51]. For array structures critical path is easy to predict since it always passes through $2N-1$ cells for N -bit array multiplier [43]. Whereas in tree multipliers, critical path is not predictable, rather it depends on

- Size of compression block used
- Placement of compression blocks in compression tree
- Input pattern of signals to compression blocks in the worst path.

Hence the worst path can be optimized by not only the optimized structure of local compression block used but their and interconnection (also known as global optimization [56]). All inputs at particular compression block of same weight treated equally, but worst case calculation highly dependent on the connectivity of these inputs [36], and in some cases determines the worst delay. Careful injection of carrying signals to compression block can considerably reduce the worst delay. Consequently, operating frequency can be scaled up not only by the supply voltage, body biasing and optimizing worst delay but also by varying compression block size and its position in compression tree.

5.2.1 WORST DELAY ANALYSIS IN LITERATURE

Finding worst delay for the multiplier is always challenging. Most researchers simulate some *dedicated vectors* for the worst case, accompanied by random vectors analysis to support their estimations. For example, authors in [14] showed that worst case for CPL based 16x16 multiplier is when; multiplier tied to 1000 0000 0000 0001 and multiplicand swaps from (1111 1111 1111 1111 \rightarrow 0111 1111 1111 1111), authors calculate power by sweeping one input to all combinations while tied the other input to all ones. Authors in [135] [36, p. 16] simulated 1200 random vectors including special cases for 16x16 multiplier. (1000000000001111) x (1111111111111111) and (0001111111111111) x (1111101110110111), and (0000000000000000) x (1111111110100000). Goto et al. [33] conducted a simulation study for regular array 54x54 multiplier with 20,000 random vectors and 12 worst vectors. Authors in [51] synthesize the compression tree by generating pattern module in L language. Authors in [13] performed the simulation for 3000 random vectors including 100 intentional worst vectors. Authors in [45] showed two testing schemes: ripple testing and exhaustive testing. Ripple testing was performed to test circuit before fabrication while exhaustive testing was performed on-chip. Authors showed that worst case is when input swings from (1111 1111 X 1000 0000) \rightarrow (1111 1111 X 1000 0001).

Authors in [17] simulate 10,000 vectors for 54x54 and show that worst case is when input switch from all zeros to all ones to again all zeros, whereas output follows same as inputs. In [42] author simulated worst case for CPL based 16x16 multiplier when multiplier tied to all ones and multiplicand switch from 0000 0000 0000 0001 \rightarrow 1111 1111 1100 0001. In [36], authors report worst case for *only compression stage* as carrying moves from least significant bit to most significant bit, depicted in Figure 5.2 for 32x32 bit multiplier. However, in actual multipliers changing only one partial product of compression stage is not possible through input bits. Worst delay in multiplier occurs when carrying flow horizontally in RCA from the right columns towards the left columns.

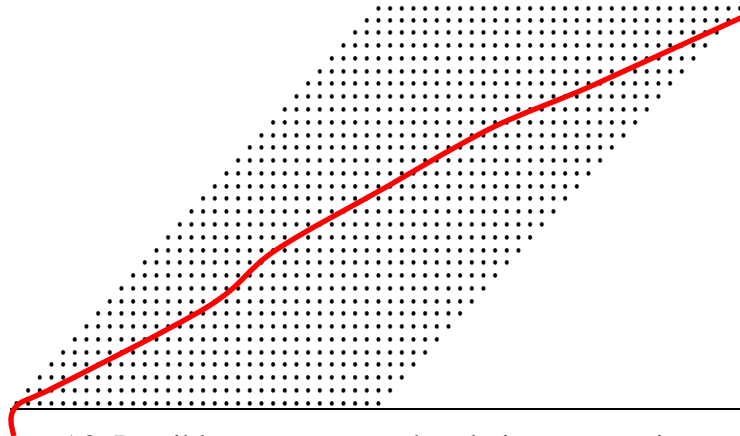


Figure 5.2. Possible worst-case path only in compression stage

5.2.2 ESTIMATED WORST CASE

In compression stage of multipliers, middle column ideally allows carrying signals to pass through the right half to left half with a minimum number of partial products to next stages [54]. What we claim for multiplier under study, *input vector that makes two rows of partial products active high that overlaps at middle column is actually a worst combination, which generates a carry in middle column that ripples all along the final addition to MSB*. Consider the first stage of the 8-bit multiplier in Figure 5.3. Input combination, which makes the stated condition in compression tree, is (11111111 x 10000001). Figure 5.3 also depicts the path of carrying generated in the middle column to MSB. Extra care has been taken for compression blocks in the middle column to make sure that generated carry has the **possible gate delays** as it flows downwards.

The large systematically vectors results (.csv file) generated by Cadence tool, manipulated in MATLAB confirms our claim of worst-delay. Table 10 depicts the details of worst delay distribution, the number of gate delay (XOR/MUX/AND) and restoring inverter for both compression and final addition stages. Optimal delays have been highlighted for each architecture size. For both 8x8 and 16x16, ARC2 has a minimum delay because of good agreement of compressor size with multiplier size, which results in compact architecture.

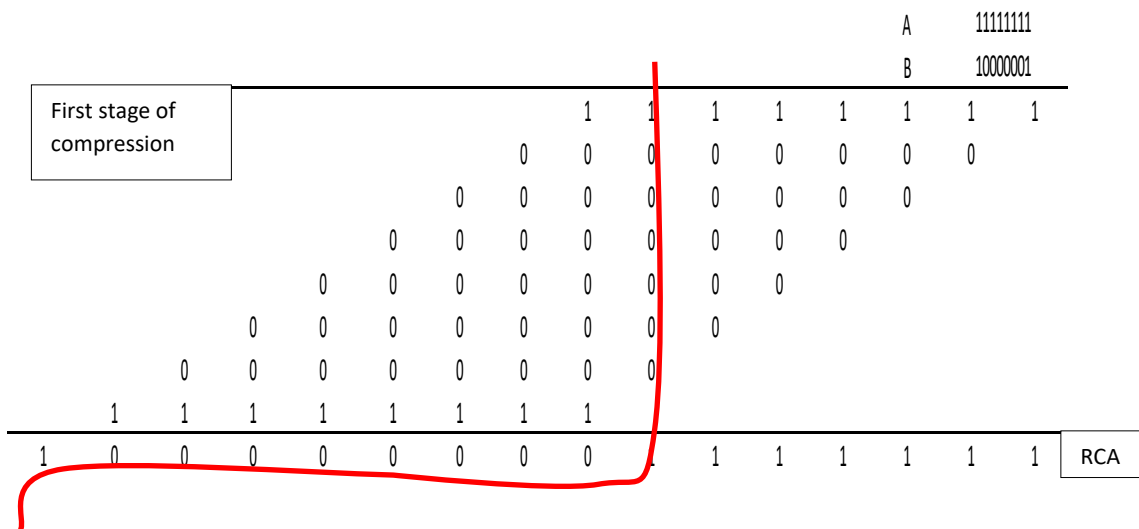


Figure 5.3. Worst case delay input vectors, 8x8 bits multiplier as an example

Table 10. Worst case distribution in multiplier architecture

Counter	Size of multiplier	Compression Stage			Final Addition Stage			Total Actual Delay including PP (ps)
		XOR gate	Inverters	Actual delay (ps)	XOR gate	Inverters	Actual delay (ps)	
ARC1	8x8	8	4	937	9	8	1323	2656
	10x10	10	5	1228	11	10	1814	3558
	12x12	10	5	1208	14	12	2143	3840
	16x16	12	6	1462	17	16	2795	4837
ARC2	8x8	6	2	686	9	8	1340	2462
	10x10	9	3	1132	11	10	1783	3365
	12x12	9	3	1197	13	12	2108	4069
	16x16	9	3	1188	18	16	2756	4716
ARC3	8x8	8	2	1072	9	7	1210	2691
	10x10	7	2	929	11	10	1773	3145
	12x12	10	3	1236	13	12	2182	3907
	16x16	11	3	1497	17	16	2756	4833
ARC4	8x8	8	2	1014	9	7	1282	2631
	10x10	8	2	1052	11	10	1840	3303
	12x12	8	2	1052	13	12	2166	3675
	16x16	13	3	1630	17	16	2759	4974
ARC5	8x8	14	3	1384	7(1A)*	6	930	2680
	10x10	14	4	1826	9(1A)*	8	1285	3521
	12x12	14	3	1552	12(3A)*	11	1827	3832
	16x16	14	4	1690	16(1A)*	15	2447	4716

* 'A' represents AND gates in worst path

5.3 LEAKAGE AND POWER ESTIMATION

Multiplier power is bonded by wiring capacitance, regularity of compression stage, size of compression block, and efficiency of individual compression block. Power and delay are commonly used design metrics for VLSI system, while power and delay product or PDP combines both power and delay gives a better understanding of the goodness of architectures having the same functionality. Average power can be reduced by optimizing a) dynamic power b) static power c) short circuit power. From old technology i.e. 350nm and above, dynamic power dominates over static power, whereas 90nm to state of the art technologies, static power directs circuit performance [136]. Therefore modern circuits having high leakage current and energy instead of power is better evaluation criteria [23].

As depicted in Figure 5.1, for the constant supply voltage (1.8 V), the current has been averaged for all simulated systematic vectors to calculate average power (including both dynamic and static components). One method to calculate the static component of power is to estimate activity by simulating multipliers under test with different operating frequencies. From the activity, the dynamic component can be calculated, and subtracting dynamic component gives the static power of circuit under test. This method was conducted by previous graduate student [24] working on this project. Another approach to calculating the static power does not require detailed simulation runs; It simple takes current samples just before next input switch i.e. when there is no activity in the circuit (annotated in Figure 5.4).

Inputs switching frequencies of multipliers have been selected to achieve steady-state conditions in compression stage of multiplier; therefore, this method is safe and accurate. (For 8-bit and 10-bit, frequency is 142.85 MHz, for 12-bit frequency is 111.11 MHz whereas for 16-bit multiplier frequency is 71.42 MHz). Averaging sampled current values through simulation tool makes even easier to calculate the static current, which multiplied by voltage to get static power. After presenting the methodology of calculation, remaining section of this chapter discusses the simulation results of 8x8, 10x10, 12x12 and 16x16 bits multipliers.

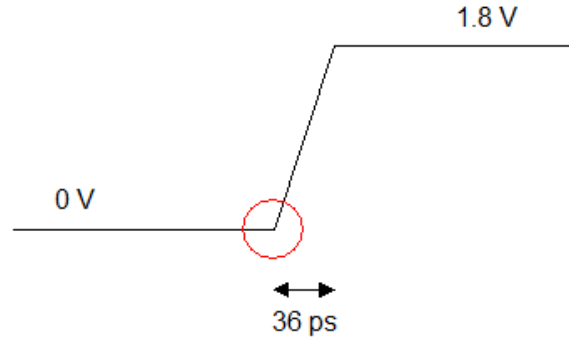


Figure 5.4. A sampling of input current when $\alpha = 0$.

5.4 SIMULATION RESULTS

5.4.1 8x8 MULTIPLIER

Simulation results and worst delay for the 8-bit multiplier are shown in Table 11 and Table 12 respectively. The average current of ARC2 has minimum current in compression stage, while ARC1 has minimum average current for final addition. In former case 4:2 compressor based architecture (ARC2) makes a compact design with minimum number of stages and moderate number of inverters, while in the latter case since (3,2) counter based architecture (ARC1) has the most number of the stage or restoring inverters, therefore all signals arrived at final adder are of good quality.

ARC1 has the highest leakage in compression stage as expected, because of a large number of MOSFETs. While ARC3 has maximum average current due to large compression block size (5:2 compressor) and a smaller number of restoring inverters. 6:2 compressor in ARC4 is larger than 5:2 compressor in ARC3, but in 8-bit ARC4, only four 6:2 compressors have been deployed, rest partial products are compressed by 4:2 compressor.

This result in better performance of ARC4 compared to ARC3. ABACUS architecture (ARC6) in 8-bit multiplier has 9.2% and 18.7% worst delay than ARC1 and ARC2, while 6.2%, 2.7% and 7.1% efficient than ARC3, ARC4, and ARC5 respectively. These percentages can improve hybridizing ABACUS with either saturated counters or compressors.

Table 11. Simulation results for 8-bit multiplier architectures

		ARC1	ARC2	ARC3	ARC4	ARC5	ARC6
Com. stage	Avg. Current (uA)	395.7	365.2	426.5	418.1	447.7	422.6
	Avg. Leakage(nA)	158.1	154.3	154.6	147	152.3	149.5
	Worst delay(ps)	1334	1117	1481	1379	1750	1337
RCA	Avg. Current (uA)	39.8	55.6	78.3	74	64.3	57.4
	Avg. Leakage (nA)	3.6	4.1	3.4	3.9	3.5	3.7
	Worst delay (ps)	1323	1340	1210	1282	930	1317
PDP (fJ)		2082	1864	2445	2357	2469	2293
EDP (YJs)		5530	4591	6581	6271	6616	6086

Table 12. Contribution of each block in worst case for 8-bit multipliers in Pico seconds (ps)

	PP. Generator	(8,4) counter	(7,3) counter	6:2 compressor	5:2 compressor	4:2 compressor	(2,3,3) counters	(3,2) counters	Final addition
ARC1	397.2	-	-	-	-	-	-	936.5	1322
ARC2	431	-	-	-	-	686	-	-	1340
ARC3	397.3	-	-	-	1084	-	-	-	1210
ARC4	365.3	-	-	630.8	-	382.9	-	-	1251
ARC5	366.3	-	616.5	-	-	-	767	-	930
ARC6	396.6	719.5	-	-	-	-	-	222	1317

5.4.2 10x10 MULTIPLIER

Simulation results and worst-case distribution of 10x10 bits multiplier are depicted in Table 13 and Table 14 respectively. 10x10 bits results into irregular multiplier structures and thus shows irregular trends in simulation results depicted in Table 13.

Table 13. Simulation results for 10-bit multiplier architectures

		ARC1	ARC2	ARC3	ARC4	ARC5	ARC6
Com. stage	Avg. Current (uA)	430.2	421.5	408.8	489.2	463.7	476.6
	Avg. Leakage(nA)	172.7	169	164.3	163.5	163.5	104.7
	Worst delay(ps)	1671	1581	1372	1464	2236	1925
RCA	Avg. Current (uA)	34.4	50.2	60.6	68.8	48.4	61.8
	Avg. Leakage (nA)	4.3	4.8	5.3	5.1	4.6	4.8
	Worst delay (ps)	1814	1783	1773	1839	1285	1528
PDP (fJ)		2915	2857	2657	3317	3245	3346
EDP (YJs)		10158	9613	8356	10958	11424	11555

Table 14. Contribution of each block in worst case for 10-bit multipliers in pico-seconds (ps)

	PP. Generator	(10,4) counter	(7,3) counter	6:2 compressor	5:2 compressor	4:2 compressor	(2,3,3) counters	(3,2) counters	Final addition
ARC1	443	-	-	-	-	-	-	1228	1814
ARC2	450	-	-	-	-	1131.6	-	-	1783
ARC3	443	-	-	-	501	428	-	-	1774
ARC4	412	-	-	683	-	369.1	-	-	1840
ARC5	410	-	506.1	-	-	-	1320	-	1284
ARC6	410	901	-	-	-	614.1	-	-	1528

For 10x10 bits multiplier, ARC3 (5:2 compressor based architecture), results into a compact design and have smallest average current in compression stage and PDP. Compression accomplished only in two stages in ARC3, therefore, it has fewer MOSFETs in the worst path (Table 10) and accordingly has minimum simulated delay of 1372 ps (Table 13). ABACUS shows worst PDP results for this size of multiplier because:

- The unsaturated counter in compression stage results in bulky multiplier with the higher average current.
- Usage of wide saturated and/or unsaturated blocks are better as they push carries far away, but wide compression circuits result in poor noise margin at the final stage and result in long delays and glitches.
- Fan-out within the wide compression blocks is relatively high with equivalent cascaded smaller compression blocks, which result in longer delays.

Methods and suggestions to address these issues will later discuss in Chapter 6 as future work, but for now, in 12-bit and 16-bit multipliers, ABACUS will not be considered for simulation.

5.4.3 12x12 MULTIPLIER

Simulation results and worst-case distribution for each compression block of the 12-bit multiplier are depicted in Table 15 and Table 16 respectively. Two rows of cascaded 6:2 compressor in first of ARC4 make a compact design for the 12-bit multiplier. This results in the optimized delay (1827 ps) for ARC4 architecture (Table 10). However, due to a small number of restoring inverters, it has higher average current compared to ARC2 (4:2 compressor based architecture). Following the trend of 8-bit, ARC1 has minimum average current in RCA whereas ARC2 has minimum PDP or EDP due to lower average current.

Table 15. Simulation results for 12-bit multiplier architectures

		ARC1	ARC2	ARC3	ARC4	ARC5
Com. stage	Avg. Current (uA)	583.4	549.4	574.8	640	588.8
	Avg. Leakage(nA)	225	220.5	215.3	209	211.6
	Worst delay(ps)	1697	1693	1725	1509	2004
RCA	Avg. Current (uA)	44.5	65.5	75.1	82	72
	Avg. Leakage (nA)	5.6	6.1	6.1	6.4	5.6
	Worst delay (ps)	2143	2108	2182	2166	1827
PDP (fJ)		4340	4206	4570	4776	4557
EDP (YJs)		1666	1600	1785	1755	1746

Table 16. Contribution of each block in worst case for 12-bit multipliers Pico seconds (ps)

	PP. Generator	(7,3) counter	6:2 compressor	5:2 compressor	4:2 compressor	(2,3,3) counters	(3,2) counters	final addition
ARC1	489	-	-	-	-	-	1208	2143
ARC2	496	-	-	-	1197	-	-	2108
ARC3	489	-	-	1073	-	-	162	2182
ARC4	456	-	684	-	369	-	-	2166
ARC5	453	1098	-	-	-	453	-	1826

5.4.4 16x16 MULTIPLIER

Simulation results and worst-case distribution for each compression block of the 16-bit multiplier are depicted in Table 17 and Table 18 respectively. 16x16 bit multiplier needs

a bit more attention because this size has been used as a benchmark for comparison for enormous studies (Table 1) in addition, the number of compression blocks used in this configuration is sufficient to comment on the effect of particular compression block on multiplier's behavior. For example, in ARC4 8x8 bits multiplier, only four 6:2 compressor has been used while rest architecture was filled with 5:2, 4:2 compressors and (3,2) and (2,2) counters. Therefore, observations on effects of 6:2 compressors on 8-bit multipliers are perhaps deceptive.

Table 17. Simulation results for 16-bit multiplier architectures

		ARC1	ARC2	ARC3	ARC4	ARC5
Com. stage	Avg. Current (uA)	609.2	451.3	556	702	587.6
	Avg. Leakage(nA)	257.4	241.3	242	246	247
	Worst delay(ps)	2042	1774	2077	2214	2269
RCA	Avg. Current (uA)	30.8	35.4	64.3	87.6	56.2
	Avg. Leakage (nA)	7.6	9.3	8.5	8.6	7.8
	Worst delay (ps)	2795	2756	2756	2760	2447
PDP (fJ)		5571	3968	5395	7071	5465
EDP (YJs)		2695	1798	2607	3516	2577

Table 18. Contribution of each block in worst case for 16-bit multiplier in pico-seconds (ps)

	PP. Generator	(7,3) counter	6:2 compressor	5:2 compressor	4:2 compressor	(2,3,3) counters	(3,2) counters	Final addition
ARC1	580	-	-	-	-	-	1462	2795
ARC2	586	-	-	-	1188	-	-	2756
ARC3	580	-	-	1215	282	-	-	2755
ARC4	585	-	1274	-	356	-	-	2759
ARC5	578	1163	-	-	-	527	-	2447

After tracing the worst vector for each multiplier's architecture, Table 18 has been generated. The number of XOR or MUX gates and restoring inverters, in both compression stage and final addition stage has also been presented, including the delay in partial products generator. ARC2 (composed of 4:2 compressors), has a minimum

overall delay with 6.3% better performance than ARC1 and ARC3, and 8.9% and 3.9% better performance than ARC4 and ARC5 respectively.

Worst delay can be improved in compression stage by having *least optimized compression blocks in the worst path*. (3,2) the counter in ARC1 has a minimum delay, but the number of these counters integrates into ARC1 and makes worst delay relatively higher. 4:2 compressor in ARC2 has a larger individual delay, but a moderate number of compressors has been deployed which tend to have a smaller worst delay of the multiplier. Wider compressor implementations (ARC3 and ARC4) naturally have fewer units in compression stage, but the contribution from each unit exceeds 4:2 compressor delays, i.e. in ARC2, compression stage alone contributes 1188 ps while in ARC3 and ARC4, it is 1497 ps and 1630 ps respectively (Table 18). For large multipliers, it is expected that delay in 4:2 compressor-based implementation will have higher delays compared to 5:2 or 6:2 compressor based implementations because of a large number of units in the worst path.

The average currents for 16-bit multiplier architectures, which include dynamic and static components, are presented in Table 17. Average leakage or static current is nearly the same for ARC2 to ARC5 with little difference, whereas ARC1 has the highest leakage since it has the most number of AND gates and inverters. Both of these cells are connected to supply (V_{DD}) and cause leakage in idle mode. The average current of ARC2 is 24% smaller than ARC1 and 21%, 38% and 24% smaller than ARC3, ARC4, and ARC5 respectively.

ARC1 has smallest compression block size and have a large number of stages and restoring inverters, while ARC4 has the largest compression block size and a small number of stages and restoring inverters. Large compression blocks have comparatively small static current but large dynamic current component because of high activity within the block, whereas smaller blocks have larger static current component, and smaller dynamic current. Therefore, the optimal point of this trade-off varies with multiplier

size. Conclusion table summarizes this in the discussion as ‘small compression blocks’ and ‘large compression block’.

PDP and EDP of multiplier architectures under test are depicted in Table 17. PDP of 16-bit ARC2 has smaller PDP compared to 12-bit ARC2 PDP. This is because *12-bit multipliers are more active than 16-bit multipliers*. As mentioned before frequencies of multiplier architectures have selected such that there is no activity while taking a sample for static current. Therefore, for a 12-bit switching frequency of most active input is 111.1 MHz, while for 16-bit it is 71.43 MHz. Therefore, averaging current of relative less active multiplier gives smaller current and PDP.

PDP or EDP of the DPL based VLSI hybrid Wallace (ARC5) is higher than (3,2) counter based Wallace (ARC1), which is conflicting with FPGA board based results presented in [43]. Custom design of gates in VLSI results in variations in delay and power among different gates. FPGAs, on the other hand, generally use Look-Up Tables (LUTs) to implement gates such as AND, XOR, and MUX. Therefore, FPGA based analysis does not provide accurate delay and power comparisons for low cost and high-performance multiplier designs. In addition, high activity of hybrid Wallace results in a high average current, irrespective of shorter interconnects and smaller gate counts.

5.5 COMPRESSION EFFICIENCY SCORE OF MULTIPLIERS

Power and delay modeling and estimation is only possible when different architectures under test have common low-level circuit modules and gate libraries [137]. Power estimation primarily requires activity which can be obtained by detailed simulation runs at multiple frequencies [24]. However similar to the power estimation method proposed by [138], which gives *power cost* to each block irrespective of activity, an aggregate System Compression Energy Efficiency Score (SCEES) has been calculated for each multiplier organization by utilizing the ‘ C_R/PDP ’ score of the individual compression blocks from Chapter 4 and the fundamental statistics of compression blocks in each multiplier organization. SCEES roughly estimates the PDP of architecture based on the

performance of individual compression blocks used in particular multiplier without detailed simulations. SCEES has been calculated in Table 19 using (5.1):

$$\text{SCEES} = \frac{\sum \text{No.of compression blocks} \cdot \frac{C_r}{\text{PDP}}}{(\sum \text{No.of compression blocks} * \text{Fanout}). \text{No.of stages}} \quad (5.1)$$

The numerator contains the sum of products of a number of compression block used in particular architecture and C_R/PDP of that compression block. Denominator normalizes the scaled value of nominator with the product of a total number of compression blocks used times fan-out of each compression block and number of stages. In individual compression block simulations, fan-out was fixed to 5 fF, while in multiplier architectures fan-out varies w.r.t. column position and stages. For instance in 16-bit ARC4 (Figure 3.16), middle columns of first stage 6:2 compressors have relatively high fan-out, since they have 6:2 compressor at outputs, while 6:2 compressors at extremes end of rows in the first stage have 5:2 or 4:2 compressors at outputs with comparatively small fan-in. For SCEES the fan-out each compression block depicted in Table 20, have assigned w.r.t. the number of restoring inverters at outputs.

Estimated PDP in Table II and simulated PDP of architectures under test are depicted in Figure 5.5. SCEES prediction of minimum PDP (architecture having high SCEES will have minimum PDP and vice versa), strongly correlate with simulated results presented in with minor disassociates. For instance, in 8x8 SCEES supposed to be small for ARC5 than ARC1 but it is slightly large. Consequently, SCEES for 12-bit ARC4 and ARC5 is conflicting to the expected results. Otherwise, for all other sizes, especially 16-bit multiplier architectures, SCEES gives a very precise estimation of PDP. The reason for conflicts in estimated and simulated results is the activity of compression circuits. Activity plays a key role in the PDP that needs to be taken into account in SCEES, as typical power modeling methods [139]. In actual multiplier, activity varies with respect to the position of compression block; it increases from left (LSB) to right (middle column) and decreases from the 1st stage to the final addition stage. Signal arrival profile of output in APPENDIX A, which is in accordance with [54], reveals that the middle

column is the most active, and the activity decreases towards MSB. But tactlessly characterizing the switching capacitance of architectures requires detailed simulations at various frequencies [24].

Table 19. Fan-out of counter and compressor blocks

	ARC1	ARC2	ARC3	ARC4	ARC5
8-Bit Multiplier					
PDP	2082	1865	2445	2357	2469
Aggregate C_R /PDP	15.142	8.683	6.628	6.814	8.357
No. of Stages	4	2	2	2	3
Total No. of compression blocks	53	14	7	8	16
SCEES	22.9	34.6	28.4	26.6	26.2
10-Bit Multiplier					
PDP	2915	2857	2657	3318	3245
Aggregate C_R /PDP	25.228	13.263	11.034	7.198	10.978
Stages	5	3	2	2	4
Total No. of compression blocks	92	21	16	5	19
SCEES	18.3	22.3	29.1	17.9	17
12-Bit Multiplier					
PDP	4340.4	4206.7	4570.4	4776.7	4557.3
Aggregate C_R /PDP	36.724	18.075	17.012	12.993	14.410
Stages	5	3	3	2	4
Total No. of compression blocks	136	27	32	20	24
SCEES	18.1	21.1	19.5	21.3	15.3
16-Bit Multiplier					
PDP	5571.3	3968.2	5394.7	7071.0	5465.3
Aggregate C_R /PDP	66.116	30.405	23.537	20.575	24.273
Stages	6	3	3	3	4
Total No. of compression blocks	252	43	28	26	43
SCEES	14.7	20	16.2	12.3	14.9

Table 20. Multiplier Configuration, PDP, and SCEES

(2,2) counter	1.5
(3,2) counter	3
(7,3) counter	5
4:2 Compressor	3.5
5:2 Compressor	5
6:2 Compressor	9
(2,3,3) counter	3

5.6 CHAPTER SUMMARY

In the Table 20 shown below, observations are summarized for the effect of small and large compression block size on compression stage of the multiplier.

Table 20. Summary Table based on simulation results

Small Compression block size	Large Compression block size
<p>Results in large number of stages</p> <p>High leakage current</p> <p>Small dynamic current since there is no extra switching or activity within the compression block</p> <p>Large number of restoring inverters: extra switching</p> <p>Signal reached final addition is of good quality</p> <p>Worst delay will be low if number of inverters is not very large</p> <p>Good for small sized multipliers</p>	<p>Results in small number of stages</p> <p>Low leakage current</p> <p>High dynamic current since there is extra switching or activity within the compression block</p> <p>Small number of restoring inverters: no extra switching</p> <p>Signal reached final addition is of bad quality</p> <p>Worst delay will be high if number of inverters are very small</p> <p>Good for large-sized multipliers</p>

Graphical summary of the results discussed in this chapter for each size has been depicted in Figure 5.6 to Figure 5.9, whereas Figure 5.10 depicts the trends in different multipliers sizes. A strong relation can be observed between the size of multiplier and size of compression block. For 8x8 bits multiplier, 4:2 compressor-based architecture, which is a factor of 8 has minimum PDP. Similarly, for 10x10 bits multiplier, 5:2 compressor based architecture has minimum PDP. For 12x12 multiplier, both 4:2 and 6:2 compressor are factors of 12, but 4:2 compressor based architecture performed better since 6:2 compressor is 34% worse than 4:2 compressor. Consequently, for 16x16 multiplier, 4:2 compressor based architecture performs the best. In Figure 5.10, 16-bit multiplier has smaller PDP compared to the 12-bit multiplier as inputs to 12-bit multiplier switch faster than 16-bit multiplier (or 16-bit multiplier is less active than 12-bit multiplier). The expected results for wide multipliers, based on the trends observed by 8x8, 10x10, 12x12 and 16x16 multiplier, will be discussed in the next Chapter.

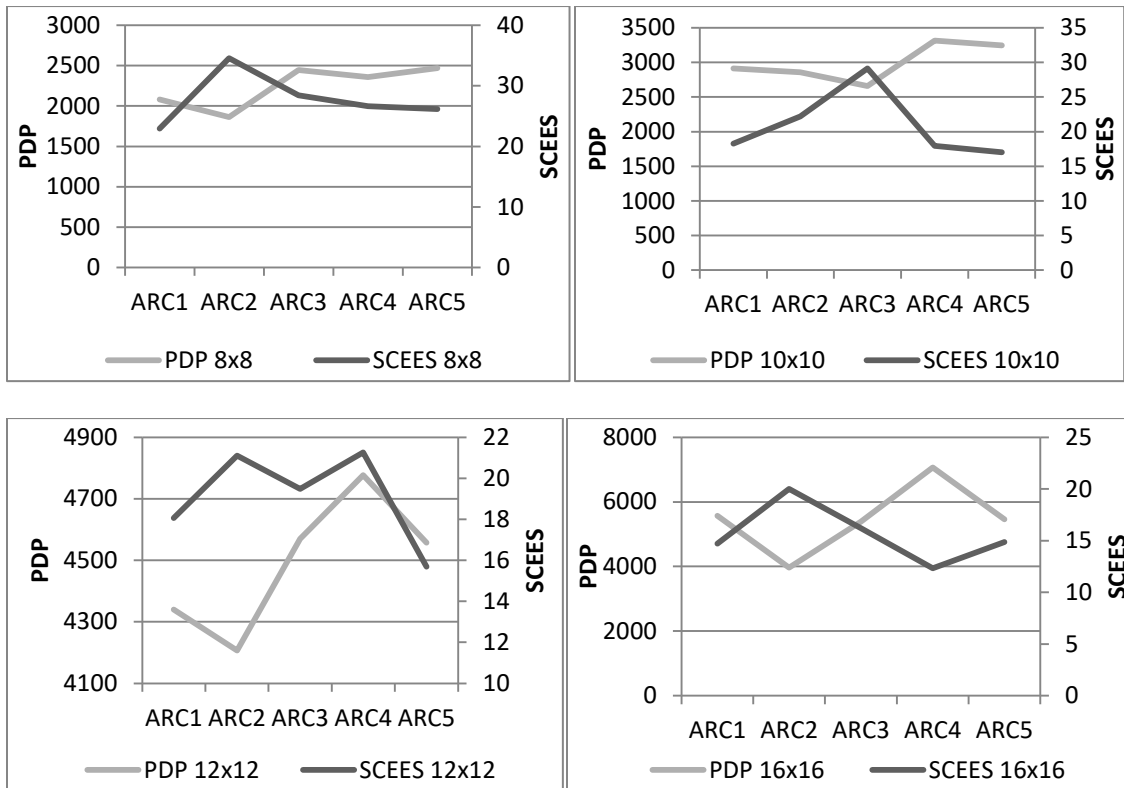


Figure 5.5 Comparison of PDP and SCEES for all given sizes

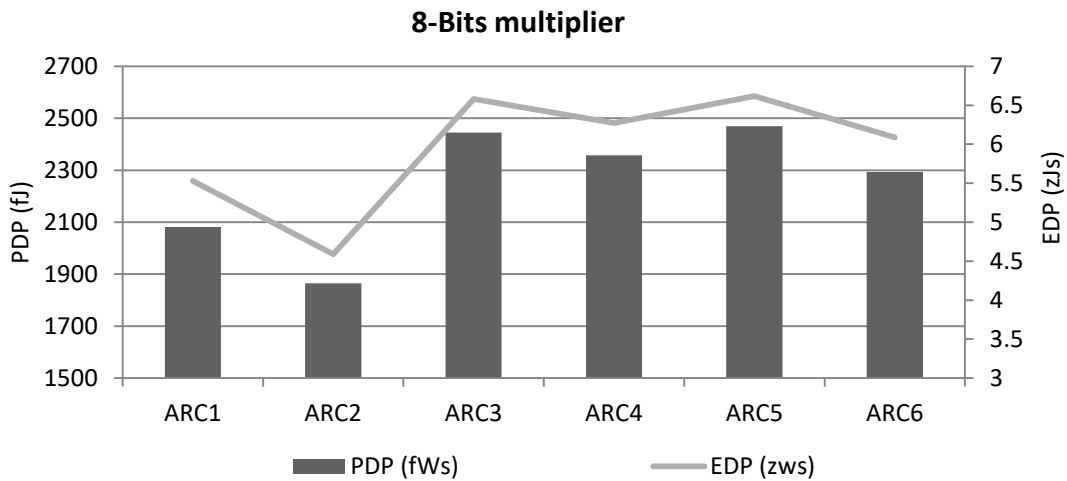


Figure 5.6. PDP and EDP in 8-bit multiplier

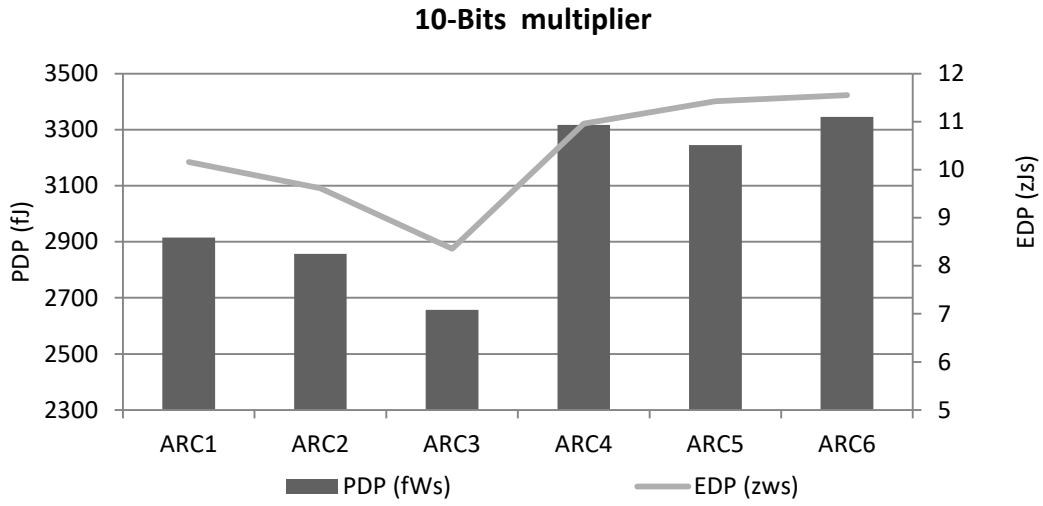


Figure 5.7 PDP and EDP in 10-bit multiplier

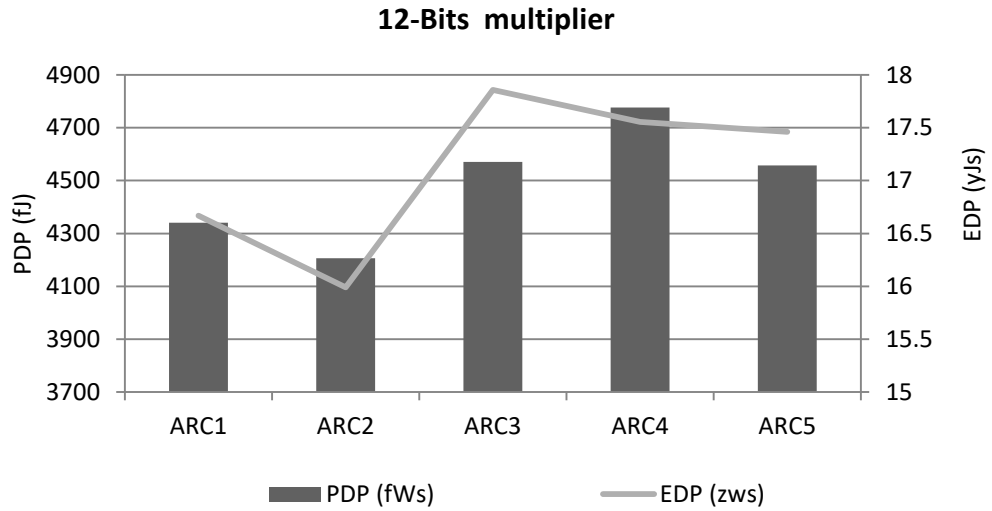


Figure 5.8. PDP and EDP in 12-bit multiplier

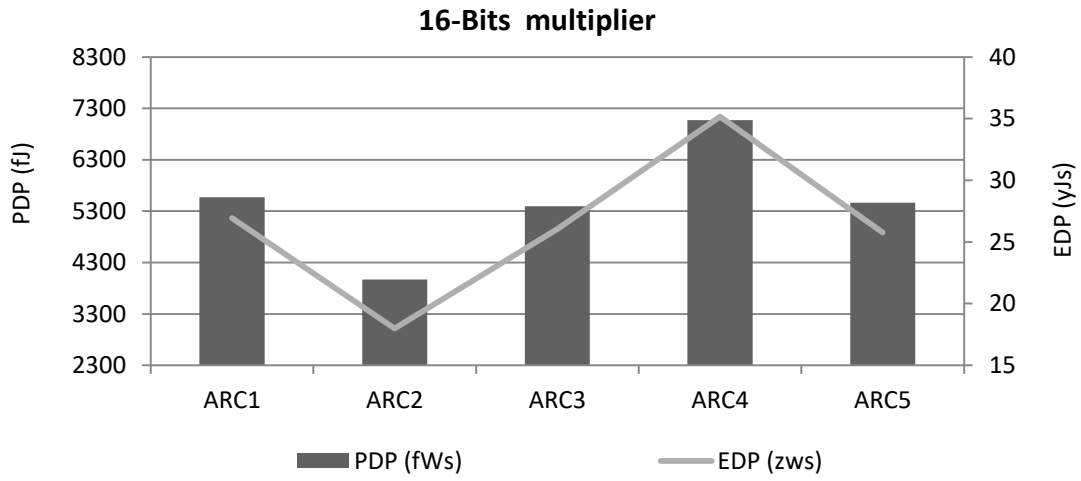


Figure 5.9. PDP and EDP in 16-bit multiplier

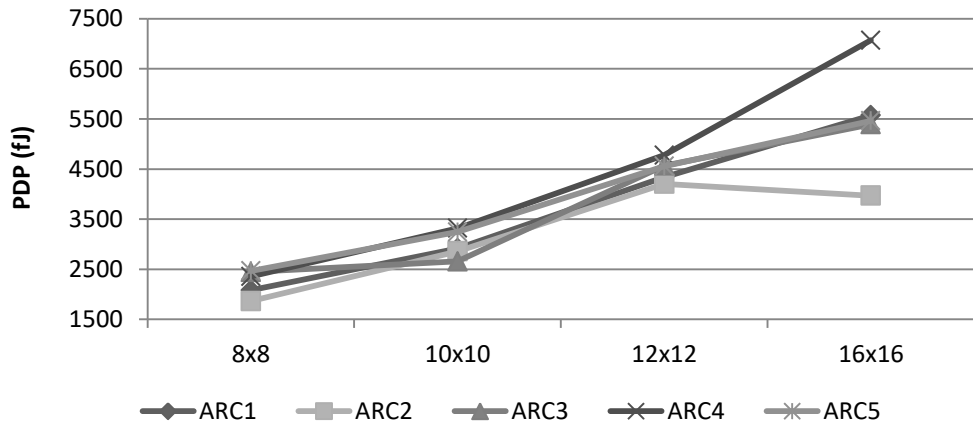


Figure 5.10. Trends in PDP for different multipliers

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 SUMMARY AND CONCLUSION

Enormous effort has been given over the last two decades on energy efficient high-speed computing due to restricted battery life. This study aims to compare and investigate methods to find energy efficient binary parallel multipliers, which are the most crucial processes in ALU, GPU, and other signal processing applications. Optimizing parallel bit multiplication was targeted by a number of researchers, therefore, a detailed comprehensive table which categorized and summarized the vast literature on parallel bit multiplication has been generated. The review Table 1 presents sub-categories into Booth encoding, type of algorithm for compression and a final adder. The aim of each study, the methodology to achieve that aim, the evaluation criteria, and the size of multiplier have also specifically mentioned in the review table. This kind of review on multipliers literature, to best of our knowledge, has not been done before.

Only the compression stage among three major components, i.e. Booth, compression and final adder, is under investigation. The compression of partial products, which contributes to almost all of the power and delay, is affected by the performance of individual compression blocks deployed, and the placement and inter-connections of these compression blocks. This study investigates both aspects.

Firstly, to investigate the performance of individual compression blocks deployed, a comprehensive discussion of different saturated and unsaturated, single and multiple column inputs counters and compressors have been presented. These compression units are later evaluated based on a number of gate delays in the worst path and simulated Power Delay Product (PDP). The former existing metric normalizes the compression ratio (C_R) with gate delay in the worst path(WGD), while the one proposed in this study normalize the compression ratio (C_R) w.r.t. PDP. With this proposed framework, the

trends observed from the new metric helped to revise the findings based on existing metric and help to forecast the effectiveness of wider counters and compressors. According to the existing metric, (2,3,3) counter is 25%, 49% and 66% superior to 4:2, 5:2 and 6:2 compressors respectively. The new metric corrects this showing that 4:2 compressor is 24% and 5:2 compressor is 13.6% better than (2,3,3) counter. However, 6:2 compressor is only 16.1% worse than (2,3,3) counters. C_R/WGD articulates that (7,3) counter is 14.5% and 28.5% superior to 4:2 and 5:2 compressor, while C_R/PDP corrects this showing that 4:2 and 5:2 compressors are 29.5% and 20.1% better than (7,3) counter respectively. The key finding of this study based on the new metric is that the wide compression blocks are not as efficient as smaller compression blocks i.e. compression of 15 bits with five (3,2) counters is more efficient than using single (15,4) counter.

Secondly, to investigate the energy efficient organization of compression blocks, five different implementations of Wallace tree algorithms of four different sizes have been realized. Four simulated data points help to forecast result for wide multipliers. All multipliers theoretically investigated and compared through exhaustive simulation that runs on Dual pass logic (DPL). DPL that is used to realize the circuits gives benefits to both complementary logic and pass transistor logic by having full swing output and low power respectively. The aspect ratio of 2.5/1 has been chosen for equal rise and fall time. Generalized equations have been generated for all implementations of Wallace tree, which helps to effectively forecast the number of stages, the number of used and unused bits in each stage and the size of final adder without architectural details.

Wallace tree multipliers under study are based on (3,2) counter, 4:2 compressor, 5:2 compressor, 6:2 compressor, and hybrid Wallace multiplier which contains a mixture of (7,3) counter and (2,3,3) counters. The novelty of this study comes from only its innovative evaluation method but also the wide 5:2 and 6:2 compressors based Wallace architectures which have not been implemented before. A new evaluation methodology is proposed to stress multipliers systematically through all possible input vectors which have better and more deterministic coverage compared to random vector sets used in the

literature. For instance, 16-bit multipliers which are typically evaluated using ~10,000 random vectors in the literature, whereas using our proposed systematic approach, 131,072 vectors have been tested, which not only gives more coverage but also refined values of average current and leakage current. The simulated PDPs of 16-bit multipliers are significant, not simply because it has been used as a benchmark for multipliers, but also for its mature number of compression blocks. PDP of 16-bit Wallace based on 4:2 compressors is 28% smaller than the conventional (3,2) counter based Wallace multiplier. Whereas it is 26%, 43% and 27% smaller than Wallace multiplier based on 5:2, 6:2 compressor and hybrid Wallace multiplier respectively.

It is likely that there is a relationship between the proposed aggregate System Compression Energy Efficiency Score (SCEES) that is built upon C_R/PDP score of the individual compression blocks and the simulated PDP performance of the multipliers. This hence enabled the development of a method to predict the most energy-efficient compression organization for a given parallel multiplier in the early stages of the design.

The simulation of different sizes multipliers has led to two significant observations:

- Small compression block results into a large number of stages in multipliers, which causes more number of devices and high leakage current.
- Large compression block results into smaller stages in multipliers, which causes poor signal quality at final adder, high activity within the block and high dynamic current.

Lastly, the important conclusion from two observations above is that for every multiplier size, there is an optimal size of compression block, which gives way to the optimized PDP. For the 16-bit multiplier, 4:2 compressors based architecture gives the best PDP results. However, for very large multiplier sizes, 4:2 compressors will not perform very well because of a larger number of stages and leakage impact.

6.2 FUTURE WORK

6.2.1 OPTIMIZATION OF ABACUS ARCHITECTURE

Optimization of the unsaturated counter

The unsaturated counters can be converted into smaller saturated counters with some extra MOSFETs coupled with them. These extra MOSFETs make it wider enough to process extra bits at the input, but simulation results showed that the resultant, saturated counters with extra MOSFETs, performs even worse than the equivalent unsaturated (2,2) and (3,2) counters. Authors in [117] proposed hybrid saturated and unsaturated counter approach, but their approach is restricted to specified multiplication, which, can be further analyzed to acquire low power efficient unsaturated counters.

Hybridization of the ABACUS

Another way of optimization of the ABACUS multiplier is to replace the unsaturated counters with saturated in compression stages, instead of changing the structure of unsaturated counter itself. Theoretical discussion of the worst path for multipliers in the previous chapter articulates that worst-case follows from middle column to final stage. Based on this, we suggest a hybrid ABACUS multiplier approach that prioritizes saturated counter and uses compressors to optimize PDP. For optimization, there are two possible methods to replace unsaturated counters with saturated counters:

1. To enhance the performance in the worst path only;
2. To decrease the average current of the multiplier in the overall architecture.

The first method of optimization requires less effort compared to the second one, while both have nearly the same impact on PDP. Replacing the unsaturated counters with the saturated counters or compressors on the worst path affects only the left half of the compression tree and later stages, whereas, in the second method, the optimization is intended on not only the worst path but all over the compression stages. This method requires more effort and time since results can only be verified through long simulation runs every time.

Following the second method, if (4,3) counter is replaced with 4:2 compressor at column #11 in the second stage of 8x8 ABACUS (Figure 3.21), four (2,2) counters and two (3,2) counters will also be replaced with 4:2 compressor to adjust this change. In total four AND gates have been removed with the addition of eight XOR and two MUX gates, which means that this hybridization method tends to consume less leakage current as there is fewer number of (2,2) counter or AND gates. Since there is no change in the worst path, the worst delay will remain the same. Figure 6.1 to Figure 6.6 depict the possible hybridization methods, to best our knowledge, of 8x8, 10x10, 12x12 and 16x16 ABACUS multipliers, which manipulates both optimizations mentioned above.

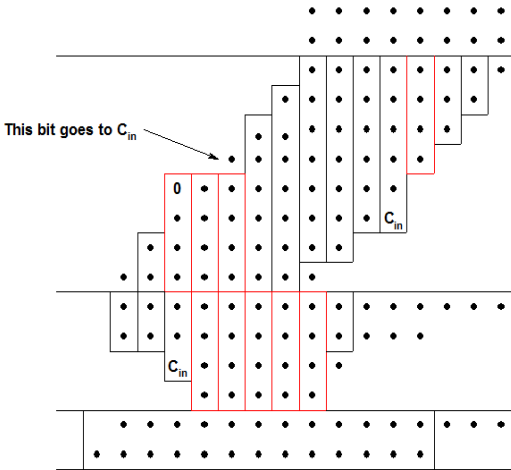


Figure 6.1. Proposed 8x8 Hybrid ABACUS Multiplier (Red rectangles depict 4:2 compressor circuits)

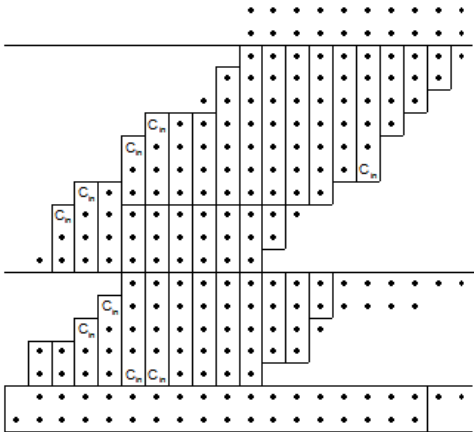


Figure 6.2. Proposed 10x10 Hybrid ABACUS Multiplier

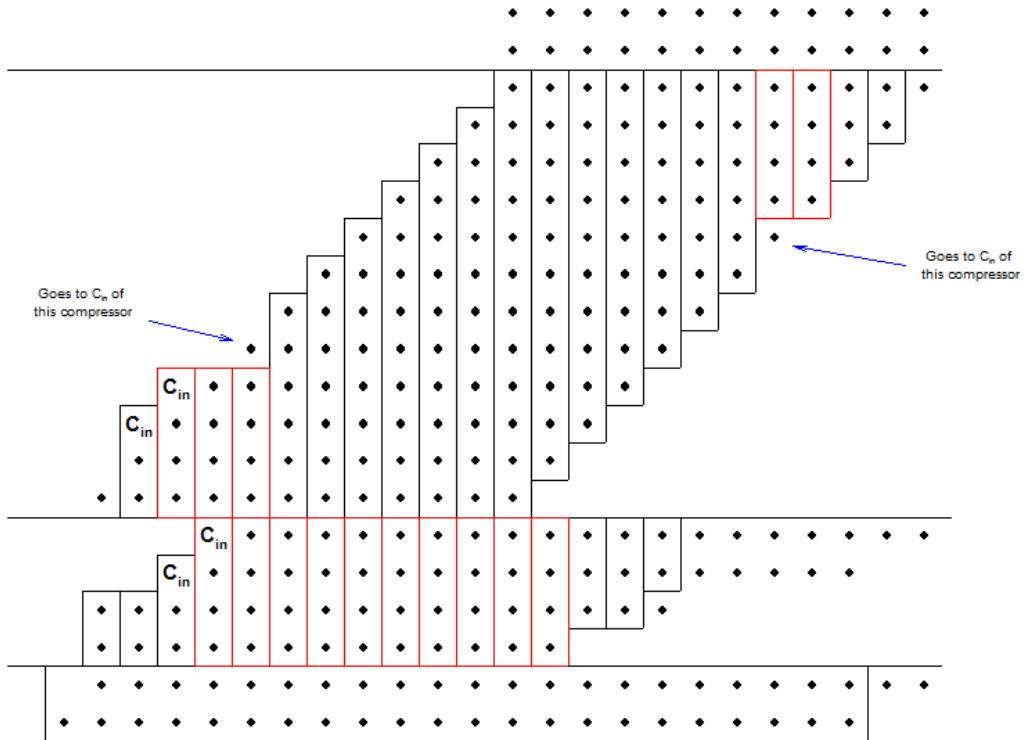


Figure 6.3. Proposed 12x12 Hybrid ABACUS Multiplier

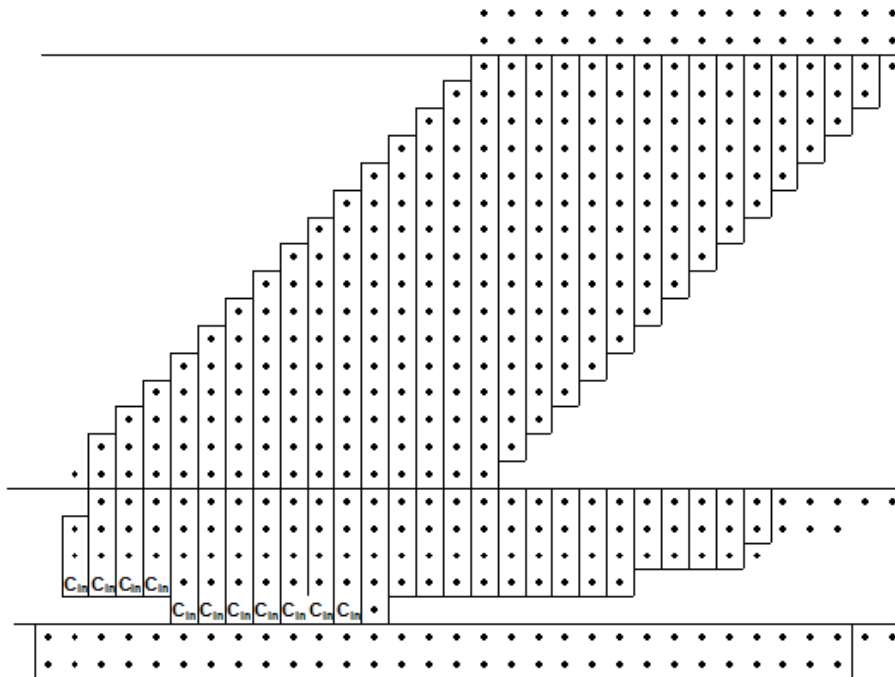


Figure 6.4. Proposed 16x16 Hybrid ABACUS Multiplier_Version-I

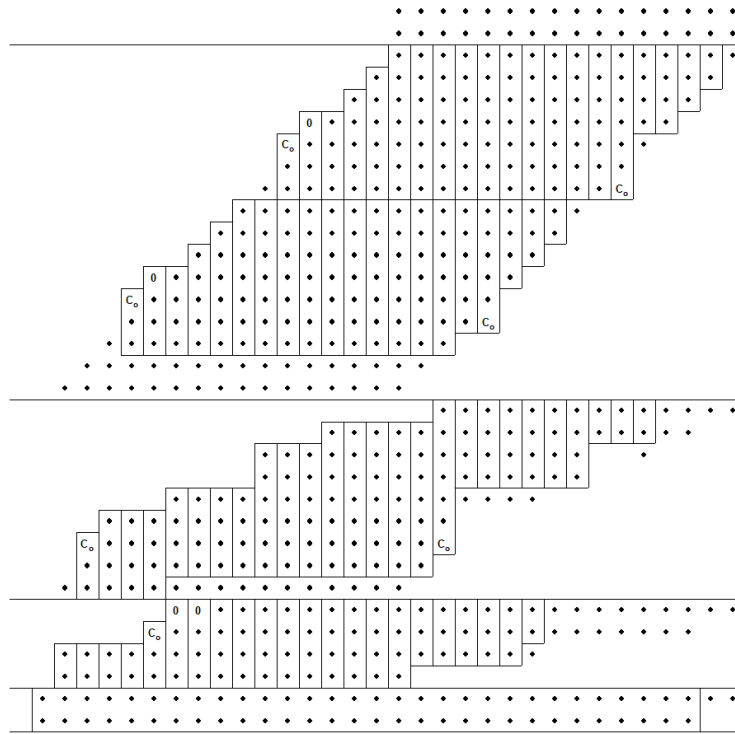


Figure 6.5. Proposed 16x16 Hybrid ABACUS Multiplier Version-II

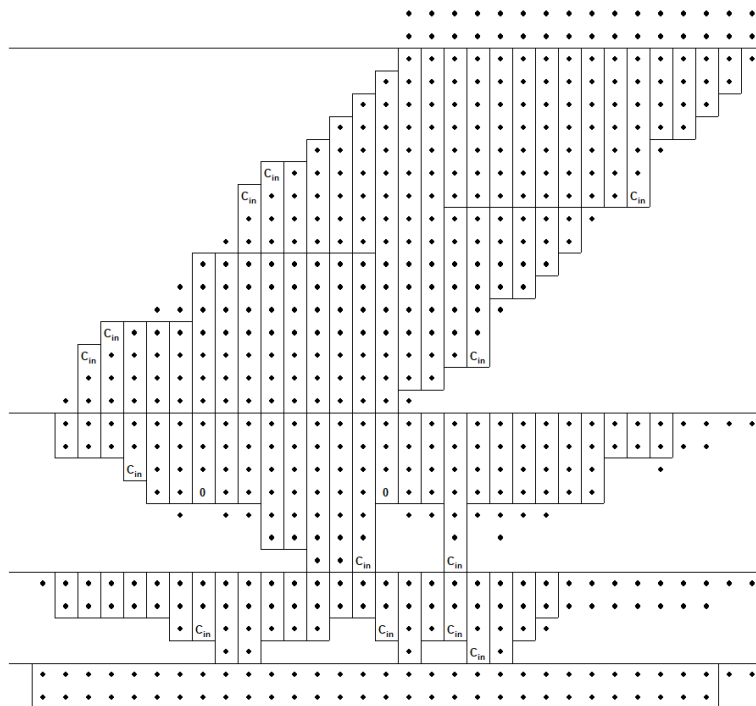


Figure 6.6. Proposed 16x16 Hybrid ABACUS Multiplier Version-III

6.2.2 INACCURATE ABACUS

Although accurate computing was an actively researched topic in the mid-90s, now it is not commonly used in data processing applications. Perhaps, accurate multiplication is outdated as it results in bulky hardware design, which works rarely at nominal ratings. Nowadays, when enormous data processing is required with minimum delay, errors are considered as an integral part of a system and error correction is an essential part of the design process [140]. For example, in N-bit RCA worst-case carry length is N-bit wide, whereas actually which is actually expected nearly $\log(N)$.

With a little compromise on the accuracy of the results, a huge chunk of energy can be off-loaded. However, this compromise without error correction circuits is only possible for certain applications, i.e. image/video processes circuits, redundant data mining, noise processing, and so on. This conflict between the reliability of the data and energy efficiency causes significant design challenges: “The straightforward application of approximate adders in a multiplier *may not* be efficient in terms of trading off accuracy for savings in energy and area” [141].

Extending the concept of approximate multiplication with recently proposed ABACUS perhaps results in a novel solution. As mentioned in [141], approximate results can be generated either at the transistor level, circuit level or at algorithms level; therefore, an enormous potential exists in these three domains to be investigated for ABACUS architecture. For example, approximate counters can be used in ABACUS instead of bulky unsaturated counters and/or even algorithm functionality can be changed such that it results in energy efficient output with errors. Comparison and validation with Wallace architecture become challenging in such cases as approximate computing has a different range of metrics, i.e. signal to noise ratio (SNR), error rate (ER) and error significance (ES), and so on.

6.2.3 MINIMIZING SIMULATION TIME FOR LARGE MULTIPLIERS

Using the generalized equations (3.1) and (3.2) for (3,2) counter based Wallace implementation and (3.5) and (3.6) for compressors based implementations respectively, a number of stages have been forecasted for ARC1 to ARC5 which are depicted, in Figure 6.7.

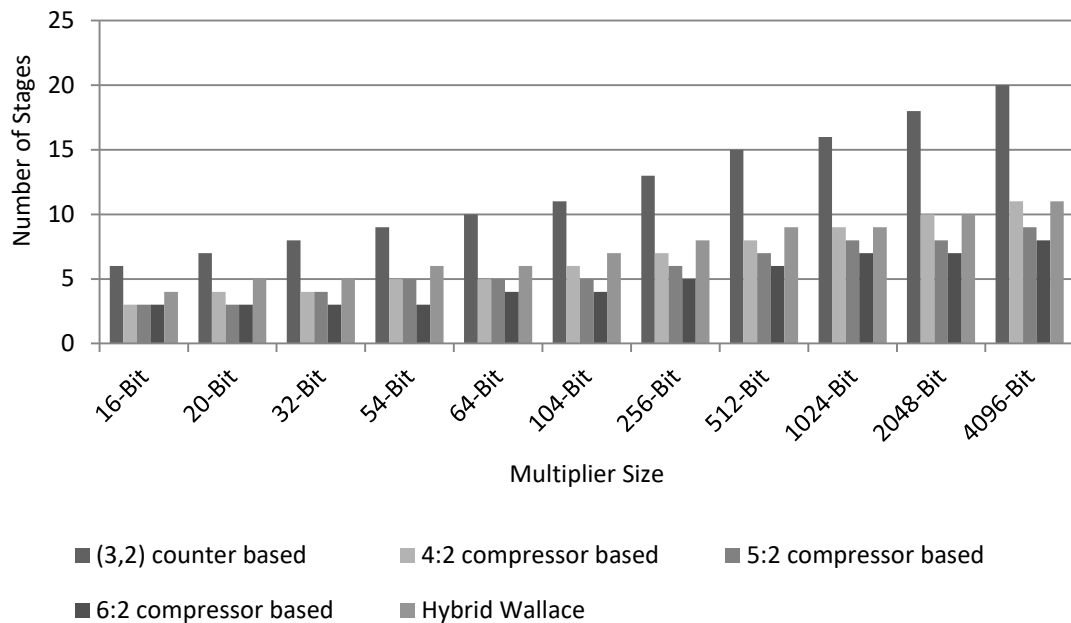


Figure 6.7. Trends of number of stages of Multiplier Architecture

Figure 6.7 shows that the growth rate of (3,2) counter based architectures is at its highest level while lowest for 6:2 compressor based architecture. From these trends, it can also be observed that most optimized results for a given architecture for a small size multiplier may differ for a wide multiplier size. For 16-bit multiplier, 4:2 compressor based architecture showed the best result, but it may give a poorer result for wide multiplications (i.e. 54x54 or 104x104), because of extra inverter stages and switching. This directed us to simulate wide multipliers, (one 16 x 16-bit multiplier simulation took ~ 11 days and 200GB, we simulated five different architectures, and each had two different simulation runs), but this is not always possible as we are restricted with time and memory limitations.

Therefore, to minimize simulation the time and space consumption, we took help from the professionals at Rutherford Appleton Laboratory, Harwell Oxford. After having a detailed discussion on implemented DPL based multipliers, they suggested two options:

- 1) Using standard cell design methodology with static timing analysis, digital simulation can be easily achieved at least a couple of orders of magnitude speed-up simulation. This sort of approach would involve creating (or writing out of virtuoso) a Verilog netlist which implements arithmetic circuit using digital standard cells. We can then run static timing analysis (STA) using Primitime/Tempus on this Verilog file to generate accurate delay information in SDF format. This STA tool can directly calculate the critical path through implemented design (or all paths in the design), to give an indication of the worst-case delays in the circuit. Alternatively, the SDF file can be loaded into the digital simulator along with Verilog netlist and a *Verilog test bench*, and design can be simulated, and one can see the delays in the digital simulator. This approach relies on having a set of standard cell libraries with liberty timing models for the cell delay characteristics, so does not readily apply to the custom designed cells generated in the DPL for this study.
- 2) Another suggestion to boost simulations is by using Transistor level static timing analysis, which gives results within minutes for the size of multipliers we are dealing with. This takes in a SPICE netlist for the full circuit, so may be easier to adopt for custom designed cells of DPL. This works directly on the SPICE netlist, identifying the digital circuit elements and directly calculating accurate delays based on reduced complexity digital models of the transistors. This does not enable to do simulations but will enable to quickly and accurately calculate the critical path. This method is also used commercially for full custom digital data paths. This may have some complex setup issues that the Cadence does not understand. In this case, we end up having to mark the cells so that Cadence can understand their functionality.

6.2.4 MODELLING OF POWER AND DELAY

For adequate analysis of diverse nature of architectures, it is not feasible to perform low circuit-level simulations; instead, it is better to model the architecture to predict its power and performance. As mentioned before, modeling is only possible when architectures under test have common low-level circuit styles, modules and gate library [137]. In addition, modeling is more reasonable if there is regularity in the design. In this study, ARC1 to ARC4 are regular structures, whereas ARC5 and hybrid ABACUS are not regular as they are based on handpicked compression blocks. Power modeling is possible by characterizing the switching capacitance. After obtaining key attribute, the activity of circuits through comprehensive simulations using CAD tool [24], [139], the power can be modelled and forecasted. To avoid detailed simulation runs to find activity method proposed by [138], which attributes power contribution to each block in architecture irrespective of inter-connectivity of these blocks, needs to be investigated to justify.

6.2.5 CIRCUIT LEVEL OPTIMIZATION: GATE RESIZING

At circuit level optimization, gate resizing can be done on non-critical paths which considerably reduce circuit PDP. Perhaps, a slack can be found where “0” means that it can either cannot be resized because it affects the worst path or it is already at minimum size [142]. This requires a very detailed analysis, which is not possible without automation.

REFERENCES

- [1] R. R. Harmon and N. Auseklis, "Sustainable IT services: Assessing the impact of green computing practices," in *Portland International Conference on Management of Engineering Technology, 2009. PICMET 2009*, 2009, pp. 1707–1717.
- [2] L. Tung, "DeepMind AI helps Google slash datacenter energy bills," *ZDNet*. [Online]. Available: <http://www.zdnet.com/article/deepmind-ai-helps-google-slash-datacenter-energy-bills/>. [Accessed: 02-Oct-2017].
- [3] P. Kurp, "Green computing," *Commun. ACM*, vol. 51, no. 10, p. 11, Oct. 2008.
- [4] M. Mills, "THE CLOUD BEGINS WITH COAL," Digital Poer Group, US, Aug. 2013.
- [5] P. Ranganathan, "Recipe for efficiency: principles of power-aware computing," *Commun. ACM*, vol. 53, no. 4, p. 60, Apr. 2010.
- [6] B. Walsh, "The Surprisingly Large Energy Footprint of the Digital Economy [UPDATE]," *Time*.
- [7] L. A. Barroso, "The Price of Performance," *Queue*, vol. 3, no. 7, pp. 48–53, Sep. 2005.
- [8] G. Goth, "Chipping away at greenhouse gases," *Commun. ACM*, vol. 54, no. 2, p. 13, Feb. 2011.
- [9] N. Bansal, T. Kimbrel, and K. Pruhs, "Speed scaling to manage energy and temperature," *J. ACM*, vol. 54, no. 1, p. 1–es, Mar. 2007.
- [10] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS digital design," *IEEE J. Solid-State Circuits*, vol. 27, no. 4, pp. 473–484, Apr. 1992.
- [11] S. Kim, J. Kim, and S. Y. Hwang, "New path balancing algorithm for glitch power reduction," *IEE Proc. - Circuits Devices Syst.*, vol. 148, no. 3, pp. 151–156, Jun. 2001.
- [12] R. Zimmermann and W. Fichtner, "Low-power logic styles: CMOS versus pass-transistor logic," *IEEE J. Solid-State Circuits*, vol. 32, no. 7, pp. 1079–1090, Jul. 1997.
- [13] J. Mori *et al.*, "A 10 ns 54 times;54 b parallel structured full array multiplier with 0.5 μm CMOS technology," *IEEE J. Solid-State Circuits*, vol. 26, no. 4, pp. 600–606, Apr. 1991.
- [14] K. Yano, T. Yamanaka, T. Nishida, M. Saito, K. Shimohigashi, and A. Shimizu, "A 3.8-ns CMOS 16 times;16-b multiplier using complementary pass-transistor logic," *IEEE J. Solid-State Circuits*, vol. 25, no. 2, pp. 388–395, Apr. 1990.
- [15] N. Ohkubo *et al.*, "A 4.4 ns CMOS 54 times;54-b multiplier using pass-transistor multiplexer," *IEEE J. Solid-State Circuits*, vol. 30, no. 3, pp. 251–257, Mar. 1995.
- [16] M. Hanawa, K. Kaneko, T. Kawashimo, and H. Maruyama, "A 4.3 ns 0.3 μm CMOS 54/spl times/54 b multiplier using precharged pass-transistor logic," in *1996 IEEE International Solid-State Circuits Conference. Digest of TEchnical Papers, ISSCC*, 1996, pp. 364–365.
- [17] H. Makino, Y. Nakase, H. Suzuki, H. Morinaka, H. Shinohara, and K. Mashiko, "An 8.8-ns 54 times;54-bit multiplier with high speed redundant binary architecture," *IEEE J. Solid-State Circuits*, vol. 31, no. 6, pp. 773–783, Jun. 1996.
- [18] S. K. Hsu *et al.*, "A 110 GOPS/W 16-bit multiplier and reconfigurable PLA loop in 90-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 41, no. 1, pp. 256–264, Jan. 2006.
- [19] I. S. Abu-Khater, A. Bellaouar, and M. I. Elmasry, "Circuit techniques for CMOS low-power high-performance multipliers," *IEEE J. Solid-State Circuits*, vol. 31, no. 10, pp. 1535–1546, Oct. 1996.

- [20] C.-H. Chang, J. Gu, and M. Zhang, "Ultra low-voltage low-power CMOS 4-2 and 5-2 compressors for fast arithmetic circuits," *IEEE Trans. Circuits Syst. Regul. Pap.*, vol. 51, no. 10, pp. 1985–1997, Oct. 2004.
- [21] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits, A Design Perspective, 2nd 2002*. Prentice Hall, Englewood Cliffs, NJ.
- [22] M. Suzuki *et al.*, "A 1.5-ns 32-b CMOS ALU in double pass-transistor logic," *IEEE J. Solid-State Circuits*, vol. 28, no. 11, pp. 1145–1151, 1993.
- [23] A. Wang, A. P. Chandrakasan, and S. V. Kosonocky, "Optimal supply and threshold scaling for subthreshold CMOS circuits," in *Proceedings IEEE Computer Society Annual Symposium on VLSI. New Paradigms for VLSI Systems Design. ISVLSI 2002*, 2002, pp. 5–9.
- [24] F. Ercan, "POWER-DELAY OPTIMIZED VLSI THRESHOLD DETECTION CIRCUITS AND THEIR USE IN PARALLEL INTEGER MULTIPLICATION," Middle East Technical University Northern Cyprus Campus, 2015.
- [25] T. T. Hoang, M. Sjalander, and P. Larsson-Edefors, "A high-speed, energy-efficient two-cycle multiply-accumulate (MAC) architecture and its application to a double-throughput MAC unit," *IEEE Trans. Circuits Syst. Regul. Pap.*, vol. 57, no. 12, pp. 3073–3081, 2010.
- [26] H. Mora-Mora, J. Mora-Pascual, J. L. Sánchez-Romero, and J. M. García-Chamizo, "Partial product reduction by using look-up tables for M×N multiplier," *Integr. VLSI J.*, vol. 41, no. 4, pp. 557–571, Jul. 2008.
- [27] S. Albers, "Energy-efficient algorithms," *Commun. ACM*, vol. 53, no. 5, p. 86, May 2010.
- [28] F. J. Pollack, "New Microarchitecture Challenges in the Coming Generations of CMOS Process Technologies (Keynote Address)(Abstract Only)," in *Proceedings of the 32Nd Annual ACM/IEEE International Symposium on Microarchitecture*, Washington, DC, USA, 1999, p. 2–.
- [29] C. S. Wallace, "A Suggestion for a Fast Multiplier," *IEEE Trans. Electron. Comput.*, vol. EC-13, no. 1, pp. 14–17, Feb. 1964.
- [30] L. Dadda, "Some schemes for parallel multipliers," *Alta Freq.*, vol. 34, no. 5, pp. 349–356, 1965.
- [31] D. A. Patterson, *Computer architecture: a quantitative approach*. Elsevier, 2011.
- [32] C. C. Stearns and P. H. Ang, "Yet another multiplier architecture," in *IEEE Proceedings of the Custom Integrated Circuits Conference*, 1990, p. 24.6/1-24.6/4.
- [33] G. Goto, T. Sato, M. Nakajima, and T. Sukemura, "A 54 times;54-b regularly structured tree multiplier," *IEEE J. Solid-State Circuits*, vol. 27, no. 9, pp. 1229–1236, Sep. 1992.
- [34] G. Goto *et al.*, "A 4.1-ns compact 54 times;54-b multiplier utilizing sign-select Booth encoders," *IEEE J. Solid-State Circuits*, vol. 32, no. 11, pp. 1676–1682, Nov. 1997.
- [35] J. K. Omura and J. L. Massey, "Computational method and apparatus for finite field arithmetic," US4587627 A, 06-May-1986.
- [36] Y. Oowaki *et al.*, "A sub-10-ns 16 x16 multiplier using 0.6 nm CMOS technology," *IEEE J. Solid-State Circuits*, vol. 22, no. 5, pp. 762–767, Oct. 1987.
- [37] P. Kulkarni, P. Gupta, and M. D. Ercegovic, "Trading Accuracy for Power in a Multiplier Architecture," *J. Low Power Electron.*, vol. 7, no. 4, pp. 490–501, Dec. 2011.
- [38] D. Kelly, B. Phillips, and S. Al-Sarawi, "Approximate signed binary integer multipliers for arithmetic data value speculation," in <http://www.ecsi.org/sites/default/files/rs5.1.pdf>, 2009.
- [39] M. J. Schulte and E. E. Swartzlander, "Truncated multiplication with correction constant [for DSP]," in *Proceedings of IEEE Workshop on VLSI Signal Processing*, 1993, pp. 388–396.

- [40] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-Inspired Imprecise Computational Blocks for Efficient VLSI Implementation of Soft-Computing Applications," *IEEE Trans. Circuits Syst. Regul. Pap.*, vol. 57, no. 4, pp. 850–862, Apr. 2010.
- [41] E. J. King and E. E. Swartzlander, "Data-dependent truncation scheme for parallel multipliers," in *Conference Record of the Thirty-First Asilomar Conference on Signals, Systems and Computers (Cat. No.97CB36136)*, 1997, vol. 2, pp. 1178–1182 vol.2.
- [42] C. F. Law, S. S. Rofail, and K. S. Yeo, "A low-power 16 times;16-b parallel multiplier utilizing pass-transistor logic," *IEEE J. Solid-State Circuits*, vol. 34, no. 10, pp. 1395–1399, Oct. 1999.
- [43] S. Abed, B. J. Mohd, Z. Al-bayati, and S. Alouneh, "Low power Wallace multiplier design based on wide counters," *Int. J. Circuit Theory Appl.*, vol. 40, no. 11, pp. 1175–1185, Nov. 2012.
- [44] S. D. Pezaris, "A 40-ns 17-Bit by 17-Bit Array Multiplier," *IEEE Trans. Comput.*, vol. C-20, no. 4, pp. 442–447, Apr. 1971.
- [45] F. S. Lee *et al.*, "A high-speed LSI GaAs 8x8 bit parallel multiplier," *IEEE J. Solid-State Circuits*, vol. 17, no. 4, pp. 638–647, Aug. 1982.
- [46] Y. Harata, Y. Nakamura, H. Nagase, M. Takigawa, and N. Takagi, "A high-speed multiplier using a redundant binary adder tree," *IEEE J. Solid-State Circuits*, vol. 22, no. 1, pp. 28–34, Feb. 1987.
- [47] R. Sharma, A. D. Lopez, J. A. Michejda, S. J. Hillenius, J. M. Andrews, and A. J. Studwell, "A 6.75 ns 16 times;16 bit multiplier in single-level-metal CMOS technology," *IEEE J. Solid-State Circuits*, vol. 24, no. 4, pp. 922–927, Aug. 1989.
- [48] M. R. Santoro and M. A. Horowitz, "SPIM: a pipelined 64*64-bit iterative multiplier," *IEEE J. Solid-State Circuits*, vol. 24, no. 2, pp. 487–493, Apr. 1989.
- [49] M. Nagamatsu, S. Tanaka, J. Mori, K. Hirano, T. Noguchi, and K. Hatanaka, "A 15-ns 32 times;32-b CMOS multiplier with an improved parallel structure," *IEEE J. Solid-State Circuits*, vol. 25, no. 2, pp. 494–497, Apr. 1990.
- [50] E. Hokenek, R. K. Montoye, and P. W. Cook, "Second-generation RISC floating point with multiply-add fused," *IEEE J. Solid-State Circuits*, vol. 25, no. 5, pp. 1207–1213, Oct. 1990.
- [51] P. J. Song and G. D. Micheli, "Circuit and architecture trade-offs for high-speed multiplication," *IEEE J. Solid-State Circuits*, vol. 26, no. 9, pp. 1184–1198, Sep. 1991.
- [52] J. Fadavi-Ardekani, "M*N Booth encoded multiplier generator using optimized Wallace trees," *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol. 1, no. 2, pp. 120–125, Jun. 1993.
- [53] F. Lu and H. Samueli, "A 200 MHz CMOS pipelined multiplier-accumulator using a quasi-domino dynamic full-adder cell design," *IEEE J. Solid-State Circuits*, vol. 28, no. 2, pp. 123–132, Feb. 1993.
- [54] V. G. Oklobdzija and D. Villeger, "Improving multiplier design by using improved column compression tree and optimized final adder in CMOS technology," *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol. 3, no. 2, pp. 292–301, 1995.
- [55] T. Hanyu and M. Kameyama, "A 200 MHz pipelined multiplier using 1.5 V-supply multiple-valued MOS current-mode circuits with dual-rail source-coupled logic," *IEEE J. Solid-State Circuits*, vol. 30, no. 11, pp. 1239–1245, Nov. 1995.
- [56] V. G. Oklobdzija, D. Villeger, and S. S. Liu, "A method for speed optimized partial product reduction and generation of fast parallel multipliers using an algorithmic approach," *IEEE Trans. Comput.*, vol. 45, no. 3, pp. 294–306, Mar. 1996.

- [57] Y. Hagihara *et al.*, "A 2.7 ns 0.25 μm^2 CMOS 54-bit multiplier," in *1998 IEEE International Solid-State Circuits Conference. Digest of Technical Papers, ISSCC. First Edition (Cat. No.98CH36156)*, 1998, pp. 296–297.
- [58] Y. Kim, B.-S. Song, J. Grosspietsch, and S. F. Gillig, "A carry-free 54-bit multiplier using equivalent bit conversion algorithm," *IEEE J. Solid-State Circuits*, vol. 36, no. 10, pp. 1538–1545, Oct. 2001.
- [59] R. Lin, "Reconfigurable parallel inner product processor architectures," *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol. 9, no. 2, pp. 261–272, Apr. 2001.
- [60] A. Cilaro *et al.*, "High Speed Speculative Multipliers Based on Speculative Carry-Save Tree," *IEEE Trans. Circuits Syst. Regul. Pap.*, vol. 61, no. 12, pp. 3426–3435, Dec. 2014.
- [61] A. Momeni, J. Han, P. Montuschi, and F. Lombardi, "Design and Analysis of Approximate Compressors for Multiplication," *IEEE Trans. Comput.*, vol. 64, no. 4, pp. 984–994, Apr. 2015.
- [62] E. E. J. Swartzlander, "Merged Arithmetic," *IEEE Trans. Comput.*, vol. C-29, no. 10, pp. 946–950, Oct. 1980.
- [63] A. D. Booth, "A Signed Binary Multiplication Technique," *Q. J. Mech. Appl. Math.*, vol. 4, no. 2, pp. 236–240, Jan. 1951.
- [64] O. T. C. Chen, S. Wang, and Y.-W. Wu, "Minimization of switching activities of partial products for designing low-power multipliers," *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol. 11, no. 3, pp. 418–433, Jun. 2003.
- [65] T. K. Callaway and E. E. Swartzlander, "Power-delay characteristics of CMOS multipliers," in *Proceedings 13th IEEE Symposium on Computer Arithmetic*, 1997, pp. 26–32.
- [66] K. Tsoumanis, S. Xydis, C. Efstathiou, N. Moschopoulos, and K. Pekmestzi, "An Optimized Modified Booth Recoder for Efficient Design of the Add-Multiply Operator," *IEEE Trans. Circuits Syst. Regul. Pap.*, vol. 61, no. 4, pp. 1133–1143, Apr. 2014.
- [67] Y. He and C. H. Chang, "A New Redundant Binary Booth Encoding for Fast n -Bit Multiplier Design," *IEEE Trans. Circuits Syst. Regul. Pap.*, vol. 56, no. 6, pp. 1192–1201, Jun. 2009.
- [68] Y.-H. Seo and D. Kim, "A New VLSI Architecture of Parallel Multiplier \times Accumulator Based on Radix-2 Modified Booth Algorithm," *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol. 18, no. 2, pp. 201–208, Feb. 2010.
- [69] S. R. Kuang and J. P. Wang, "Design of Power-Efficient Configurable Booth Multiplier," *IEEE Trans. Circuits Syst. Regul. Pap.*, vol. 57, no. 3, pp. 568–580, Mar. 2010.
- [70] H. Jiang, J. Han, F. Qiao, and F. Lombardi, "Approximate Radix-8 Booth Multipliers for Low-Power and High-Performance Operation," *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2638–2644, Aug. 2016.
- [71] D. Villeger and V. G. Oklobdzija, "Evaluation of Booth encoding techniques for parallel multiplier implementation," *Electron. Lett.*, vol. 29, no. 23, pp. 2016–2017, Nov. 1993.
- [72] F. Mo and R. K. Brayton, "Placement Based Multiplier Rewiring for Cell-based Designs," in *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design*, Piscataway, NJ, USA, 2008, pp. 430–433.
- [73] R. S. Waters and E. E. Swartzlander, "A Reduced Complexity Wallace Multiplier Reduction," *IEEE Trans. Comput.*, vol. 59, no. 8, pp. 1134–1137, Aug. 2010.
- [74] R. Lin, "A Regularly Structured Parallel Multiplier with Low-power Non-binary-logic Counter Circuits," *VLSI Des.*, vol. 12, no. 3, pp. 377–390, 2001.

- [75] P. C. H. Meier, R. A. Rutenbar, and L. R. Carley, "Exploring multiplier architecture and layout for low power," in *Proceedings of Custom Integrated Circuits Conference*, 1996, pp. 513–516.
- [76] A. Muhtaroglu, "ABACUS: A novel array multiplier-accumulator architecture for low energy applications," in *2010 International Conference on Energy Aware Computing (ICEAC)*, 2010, pp. 1–4.
- [77] F. Ercan and A. Muhtaroglu, "Power-delay analysis of an ABACUS parallel integer multiplier VLSI implementation," in *5th International Conference on Energy Aware Computing Systems Applications*, 2015, pp. 1–4.
- [78] T.-Y. Chang and M.-J. Hsiao, "Carry-select adder using single ripple-carry adder," *Electron. Lett.*, vol. 34, no. 22, pp. 2101–2103, Oct. 1998.
- [79] S. Perri, P. Corsonello, F. Pezzimenti, and V. Kantabutra, "Fast and energy-efficient Manchester carry-bypass adders," *IEE Proc. - Circuits Devices Syst.*, vol. 151, no. 6, p. 497, 2004.
- [80] T. Kilburn, D. B. G. Edwards, and D. Aspinall, "Parallel addition in digital computers: a new fast 'carry' circuit," *Proc. IEE - Part B Electron. Commun. Eng.*, vol. 106, no. 29, pp. 464–466, Sep. 1959.
- [81] M. Lehman and N. Burla, "Skip Techniques for High-Speed Carry-Propagation in Binary Arithmetic Units," *IRE Trans. Electron. Comput.*, vol. EC-10, no. 4, pp. 691–698, Dec. 1961.
- [82] S. Turrini, "Optimal group distribution in carry-skip adders," in *Proceedings of 9th Symposium on Computer Arithmetic, 1989*, 1989, pp. 96–103.
- [83] S. Knowles, "A family of adders," in *14th IEEE Symposium on Computer Arithmetic, 1999. Proceedings*, 1999, pp. 30–34.
- [84] J. P. Fishburn, "A depth-decreasing heuristic for combinational logic; or how to convert a ripple-carry adder into a carry-lookahead adder or anything in-between," in *27th ACM/IEEE Design Automation Conference, 1990. Proceedings*, 1990, pp. 361–364.
- [85] T. Vo and P. P. Gelsinger, "Optimally partitioned regenerative carry lookahead adder," US4737926 A, 12-Apr-1988.
- [86] J. B. Kuo, H. J. Liao, and H. P. Chen, "A BiCMOS dynamic carry lookahead adder circuit for VLSI implementation of high-speed arithmetic unit," *IEEE J. Solid-State Circuits*, vol. 28, no. 3, pp. 375–378, Mar. 1993.
- [87] T. Lynch and E. E. Swartzlander, "A spanning tree carry lookahead adder," *IEEE Trans. Comput.*, vol. 41, no. 8, pp. 931–939, Aug. 1992.
- [88] B. K. Hwang, "Carry lookahead adder," US5951631 A, 14-Sep-1999.
- [89] B. Desoete and A. De Vos, "A reversible carry-look-ahead adder using control gates," *Integr. VLSI J.*, vol. 33, no. 1–2, pp. 89–104, Dec. 2002.
- [90] R. P. Brent and H. T. Kung, "A Regular Layout for Parallel Adders," *IEEE Trans. Comput.*, vol. C-31, no. 3, pp. 260–264, Mar. 1982.
- [91] B. Parhami, *Computer arithmetic: algorithms and hardware designs*, 2nd ed. New York: Oxford University Press, 2010.
- [92] H. Ling, "High-Speed Binary Adder," *IBM J. Res. Dev.*, vol. 25, no. 3, pp. 156–166, Mar. 1981.
- [93] "IEEE Xplore Abstract - Variants of an improved carry look-ahead adder." [Online]. Available: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=2261&queryText=variants+of+>

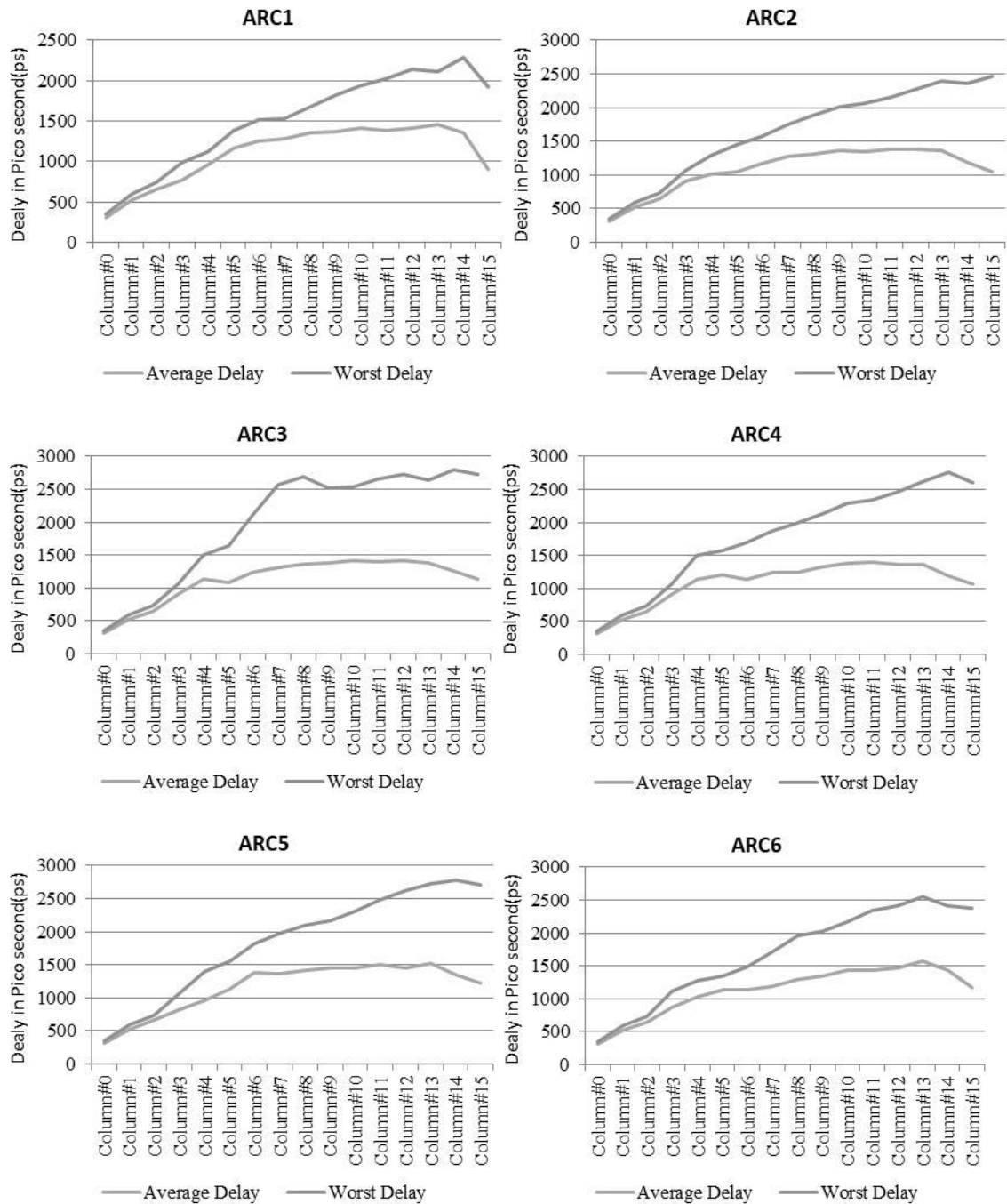
- an+improved+carry+look+ahead&newsearch=true&searchField=Search_All. [Accessed: 13-Jul-2015].
- [94] H. T. Vergos and C. Efstathiou, "Efficient modulo adder architectures," *Integr. VLSI J.*, vol. 42, no. 2, pp. 149–157, Feb. 2009.
- [95] O. J. Bedrij, "Carry-Select Adder," *IRE Trans. Electron. Comput.*, vol. EC-11, no. 3, pp. 340–346, Jun. 1962.
- [96] Y. He, C.-H. Chang, and J. Gu, "An area efficient 64-bit square root carry-select adder for low power applications," in *IEEE International Symposium on Circuits and Systems, 2005. ISCAS 2005*, 2005, p. 4082–4085 Vol. 4.
- [97] B. Ramkumar and H. M. Kittur, "Low-Power and Area-Efficient Carry Select Adder," *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol. 20, no. 2, pp. 371–375, Feb. 2012.
- [98] A. Tyagi, "A reduced-area scheme for carry-select adders," *IEEE Trans. Comput.*, vol. 42, no. 10, pp. 1163–1170, Oct. 1993.
- [99] G. A. Ruiz and M. Granda, "An area-efficient static CMOS carry-select adder based on a compact carry look-ahead unit," *Microelectron. J.*, vol. 35, no. 12, pp. 939–944, Dec. 2004.
- [100] Y. Kim and L.-S. Kim, "A low power carry select adder with reduced area," in *The 2001 IEEE International Symposium on Circuits and Systems, 2001. ISCAS 2001*, 2001, vol. 4, pp. 218–221 vol. 4.
- [101] J. C. Leininger and G. P. Taylor, "Carry save adder," US4110832 A, 29-Aug-1978.
- [102] M. S. Rashid and A. Muhtaroglu, "Novel Multiplier design for Data Rendering," in *13th IEEE BioCAS 2017*, Turin, Italy, October 19-21.
- [103] A. Pishvaie, G. Jaberipur, and A. Jahanian, "Redesigned CMOS (4; 2) compressor for fast binary multipliers," *Can. J. Electr. Comput. Eng.*, vol. 36, no. 3, pp. 111–115, Summer 2013.
- [104] P. J. Song and G. D. Micheli, "Circuit and architecture trade-offs for high-speed multiplication," *IEEE J. Solid-State Circuits*, vol. 26, no. 9, pp. 1184–1198, Sep. 1991.
- [105] A. Svoboda, "Adder With Distributed Control," *IEEE Trans. Comput.*, vol. C-19, no. 8, pp. 749–751, Aug. 1970.
- [106] I. T. Ho and T. C. Chen, "Multiple Addition by Residue Threshold Functions and Their Representation by Array Logic," *IEEE Trans. Comput.*, vol. C-22, no. 8, pp. 762–767, Aug. 1973.
- [107] E. E. Swartzlander, "Parallel Counters," *IEEE Trans Comput*, vol. 22, no. 11, pp. 1021–1024, Nov. 1973.
- [108] D. Zhang, G. A. Jullien, W. C. Miller, and E. Swartzlander, "Arithmetic for digital neural networks," in *[1991] Proceedings 10th IEEE Symposium on Computer Arithmetic*, 1991, pp. 58–63.
- [109] M. Mehta, V. Parmar, and E. Swartzlander, "High-speed multiplier design using multi-input counter and compressor circuits," in *10th IEEE Symposium on Computer Arithmetic, 1991. Proceedings*, 1991, pp. 43–50.
- [110] D. Zhang and M. I. Elmasry, "VLSI compressor design with applications to digital neural networks," *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol. 5, no. 2, pp. 230–233, Jun. 1997.
- [111] G. Deng and C. Chen, "Binary Multiplication Using Hybrid MOS and Multi-Gate Single-Electron Transistors," *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol. 21, no. 9, pp. 1573–1582, Sep. 2013.

- [112] S. Asif and Y. Kong, "Design of an algorithmic Wallace multiplier using high speed counters," in *2015 Tenth International Conference on Computer Engineering Systems (ICCES)*, 2015, pp. 133–138.
- [113] W. J. Stenzel, W. J. Kubitz, and G. H. Garcia, "A Compact High-Speed Parallel Multiplication Scheme," *IEEE Trans. Comput.*, vol. C-26, no. 10, pp. 948–957, Oct. 1977.
- [114] O. Kwon, K. Nowka, and E. E. Swartzlander, "A 16-Bit by 16-Bit MAC Design Using Fast 5:3 Compressor Cells," *J. VLSI Signal Process. Syst. Signal Image Video Technol.*, vol. 31, no. 2, pp. 77–89, Jul. 2002.
- [115] S. Dormido and M. A. Canto, "Synthesis of Generalized Parallel Counters," *IEEE Trans. Comput.*, vol. C-30, no. 9, pp. 699–703, Sep. 1981.
- [116] N. H. E. Weste and D. M. Harris, *CMOS VLSI design: a circuits and systems perspective*, 4th ed. Boston: Addison Wesley, 2011.
- [117] M. E. Robinson and E. Swartzlander, "A reduction scheme to optimize the Wallace multiplier," in *Proceedings International Conference on Computer Design. VLSI in Computers and Processors (Cat. No.98CB36273)*, 1998, pp. 122–127.
- [118] A. Dandapat, "A 1.2-ns 16×16 -Bit Binary Multiplier Using High Speed Compressors," *Int. J. Electr. Electron. Eng.*, vol. 4, no. 3, 2010.
- [119] S.-F. Hsiao, M.-R. Jiang, and J.-S. Yeh, "Design of high-speed low-power 3-2 counter and 4-2 compressor for fast multipliers," *Electron. Lett.*, vol. 34, no. 4, pp. 341–343, Feb. 1998.
- [120] D. Radhakrishnan and A. P. Preethy, "Low power CMOS pass logic 4-2 compressor for high-speed multiplication," in *Proceedings of the 43rd IEEE Midwest Symposium on Circuits and Systems (Cat.No.CH37144)*, 2000, vol. 3, pp. 1296–1298 vol.3.
- [121] M. Santoro and M. Horowitz, "A Pipelined 64x64b Iterative Array Multiplier," *1988 IEEE Int. Solid-State Circuits Conf. 1988 ISSCC Dig. Tech. Pap.*, pp. 36–37, 290, 1988.
- [122] O. Kwon, E. E. Swartzlander, and K. Nowka, "A 16-Bit x 16-Bit MAC Design Using Fast 5:2 Compressors," in *Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures, and Processors*, Washington, DC, USA, 2000, p. 235–.
- [123] J. Gu and C.-H. Chang, "Low voltage, low power (5:2) compressor cell for fast arithmetic circuits," in *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03)*, 2003, vol. 2, p. II-661-4 vol.2.
- [124] K. Prasad and K. K. Parhi, "Low-power 4-2 and 5-2 compressors," in *Signals, Systems and Computers, 2001. Conference Record of the Thirty-Fifth Asilomar Conference on*, 2001, vol. 1, pp. 129–133.
- [125] J. Liang, J. Han, and F. Lombardi, "New Metrics for the Reliability of Approximate and Probabilistic Adders," *IEEE Trans. Comput.*, vol. 62, no. 9, pp. 1760–1771, Sep. 2013.
- [126] R. Lin, "Reconfigurable inner product processor architecture implementing square recursive decomposition of partial product matrices," US6718465 B1, 06-Apr-2004.
- [127] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, and K. Roy, "IMPACT: Imprecise Adders for Low-power Approximate Computing," in *Proceedings of the 17th IEEE/ACM International Symposium on Low-power Electronics and Design*, Piscataway, NJ, USA, 2011, pp. 409–414.
- [128] K. A. C. Bickerstaff, M. Schulte, and E. E. Swartzlander, "Reduced area multipliers," in *International Conference on Application-Specific Array Processors*, 1993, pp. 478–489.
- [129] A. G. M. Strollo and D. D. Caro, "Booth folding encoding for high performance squarer circuits," *IEEE Trans. Circuits Syst. II Analog Digit. Signal Process.*, vol. 50, no. 5, pp. 250–254, May 2003.

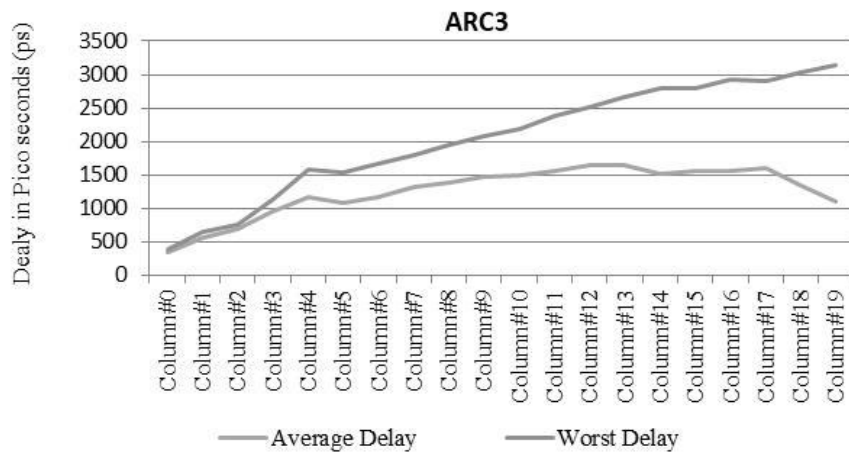
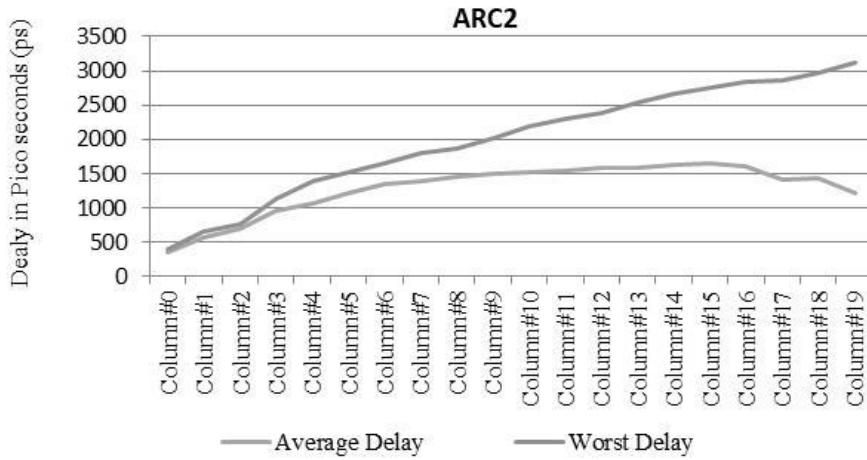
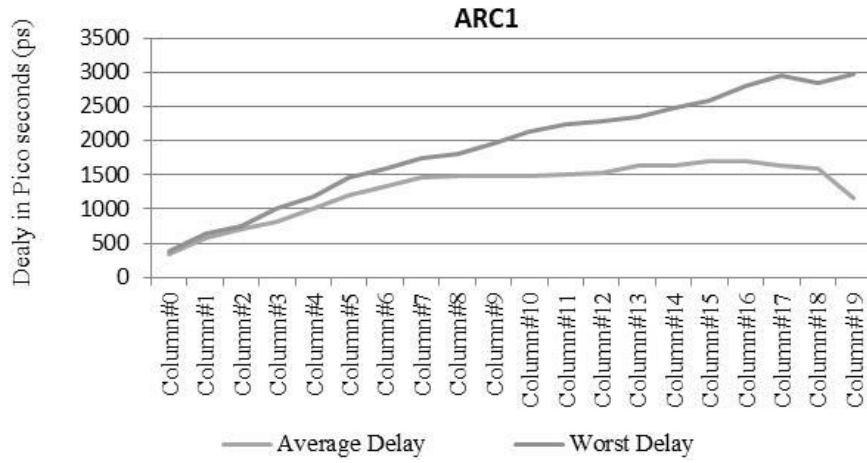
- [130] C. Nagendra, M. J. Irwin, and R. M. Owens, "Area-time-power tradeoffs in parallel adders," *IEEE Trans. Circuits Syst. II Analog Digit. Signal Process.*, vol. 43, no. 10, pp. 689–702, Oct. 1996.
- [131] M. Nachtigal, H. Thapliyal, and N. Ranganathan, "Design of a reversible single precision floating point multiplier based on operand decomposition," in *2010 10th IEEE Conference on Nanotechnology (IEEE-NANO)*, 2010, pp. 233–237.
- [132] D. GURDUR, "ARCHITECTURAL ENERGY-DELAY ASSESSMENT OF ABACUS MULTIPLIER WITH RESPECT TO OTHER MULTIPLIERS," Middle East Technical University Northern Cyprus Campus, Mersin-10 Turkey., 2013.
- [133] B. Hoppe, G. Neuendorf, D. Schmitt-Landsiedel, and W. Specks, "Optimization of high-speed CMOS logic circuits with analytical models for signal delay, chip area, and dynamic power dissipation," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 9, no. 3, pp. 236–247, Mar. 1990.
- [134] K. C. Bickerstaff, E. E. Swartzlander, and M. J. Schulte, "Analysis of column compression multipliers," in *15th IEEE Symposium on Computer Arithmetic, 2001. Proceedings*, 2001, pp. 33–39.
- [135] Y. Oowaki, K. Numata, K. Tsuchiya, K. Tsuda, A. Nitayama, and S. Watanbe, "A 7.4ns CMOS 16 #215; 16 multiplier," in *1987 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*, 1987, vol. XXX, pp. 52–53.
- [136] M. Alioto and G. Palumbo, "Power-delay optimization of D-latch/MUX source coupled logic gates," *Int. J. Circuit Theory Appl.*, vol. 33, no. 1, pp. 65–86, Jan. 2005.
- [137] Srinivas Devadas, "A Survey of Optimization Techniques Targeting Low Power VLSI Circuits," 1995, pp. 242–247.
- [138] T. Sato, M. Nagamatsu, and H. Tago, "Power and performance simulator: ESP and its application for 100 MIPS/W class RISC design," in *Proceedings of 1994 IEEE Symposium on Low Power Electronics*, 1994, pp. 46–47.
- [139] P. E. Landman and J. M. Rabaey, "Power estimation for high level synthesis," in *1993 European Conference on Design Automation with the European Event in ASIC Design*, 1993, pp. 361–366.
- [140] G. L. Shaw and G. Palm, Eds., *Brain theory: reprint volume*. Singapore ; New Jersey: World Scientific, 1988.
- [141] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," 2013, pp. 1–6.
- [142] C. H. Tan and J. Allen, "Minimization of power in VLSI circuits using transistor sizing, input ordering, and statistical power estimation," in *Proceedings of the 1994 International Workshop on Low Power Design*, 1994, pp. 75–80.

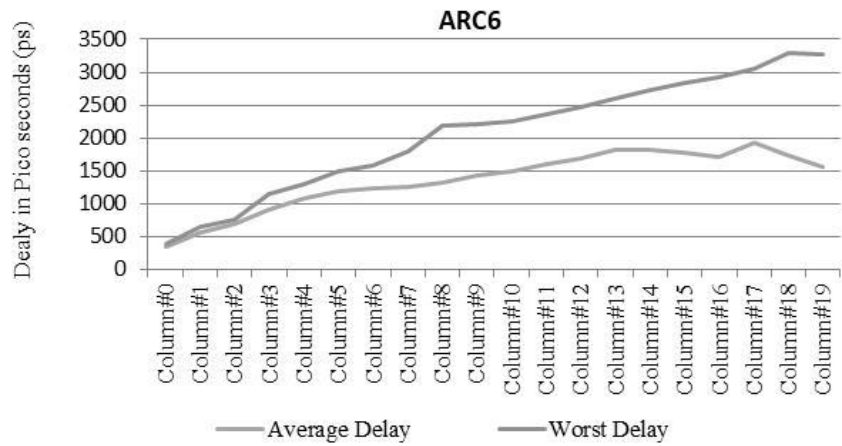
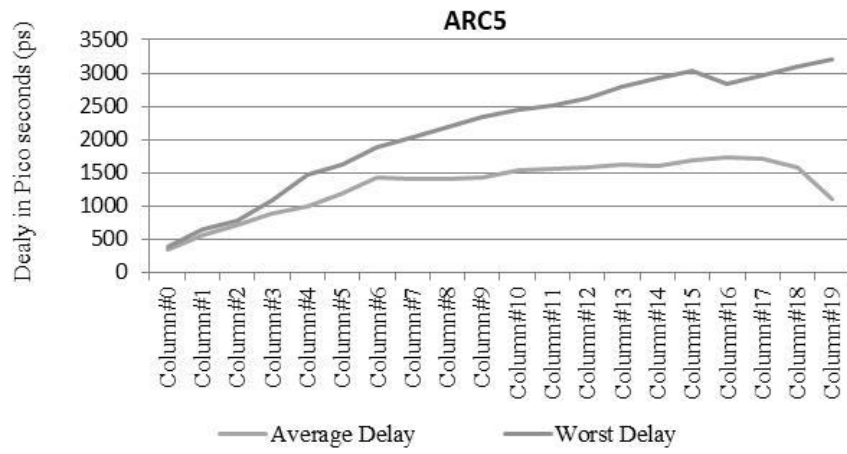
APPENDIX A

Signal arrival profile at outputs of 8-Bits multiplier

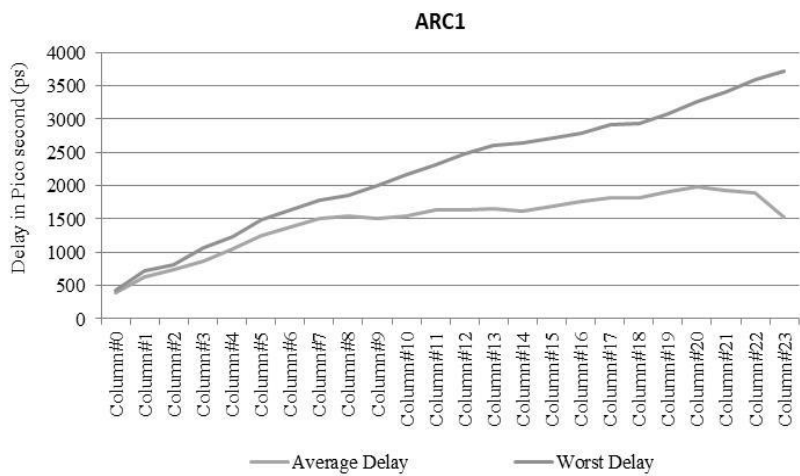


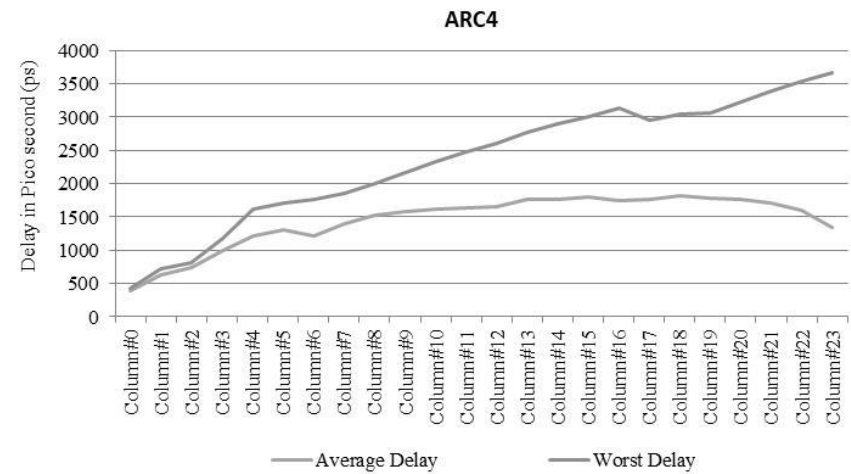
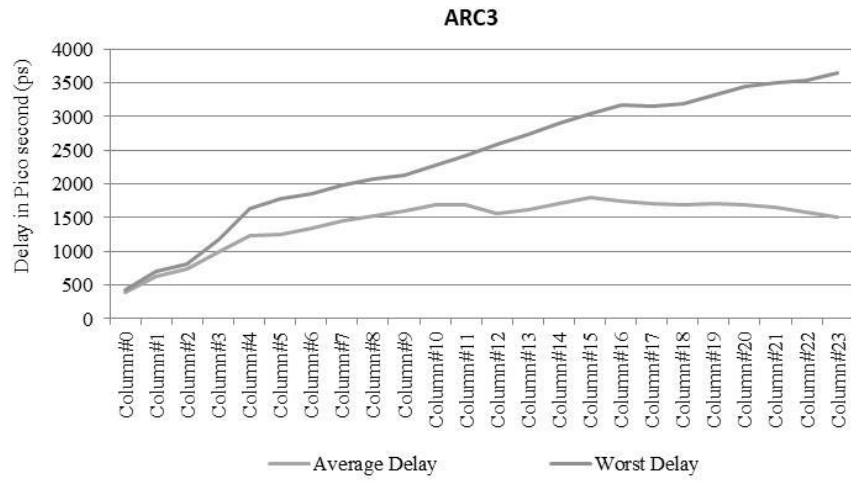
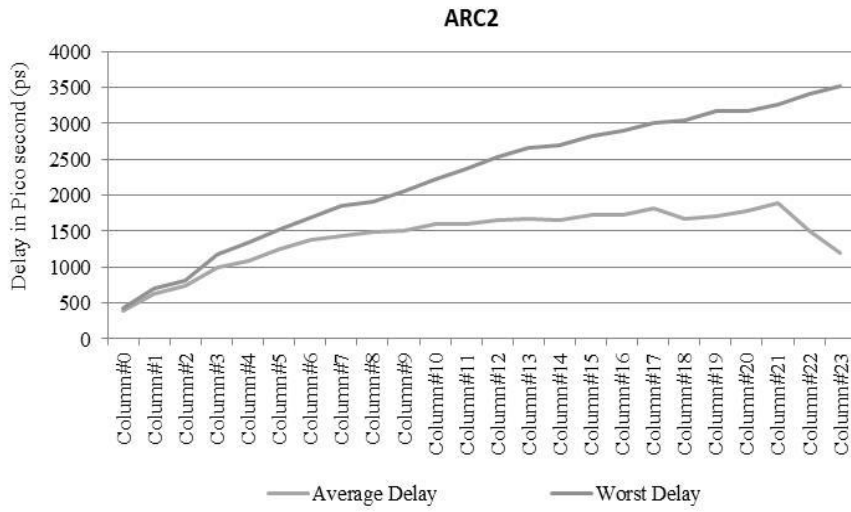
Signal arrival profile at outputs of 10-Bits multiplier

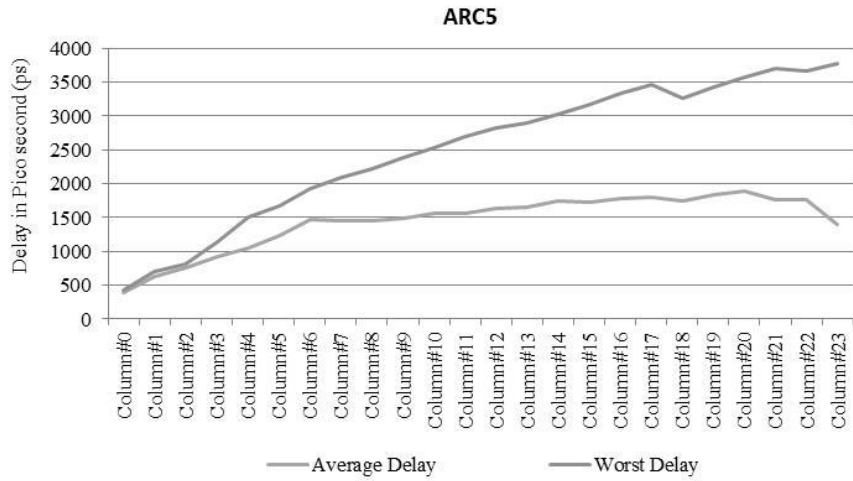




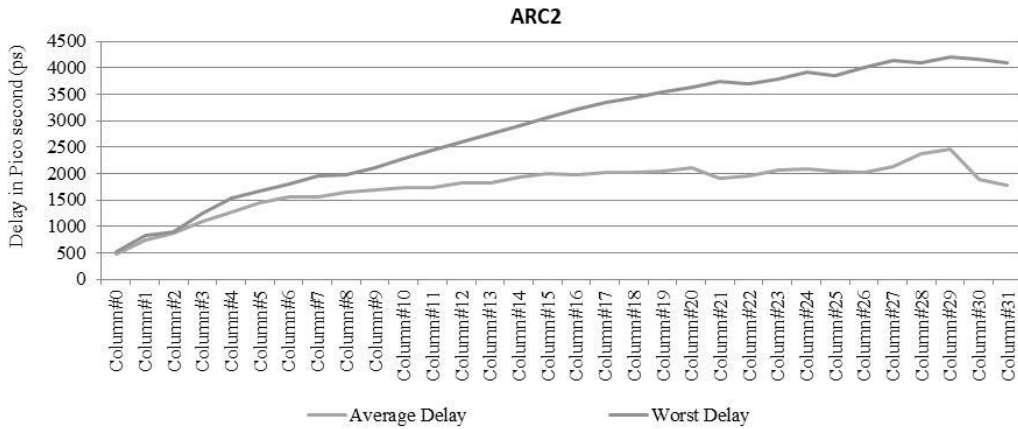
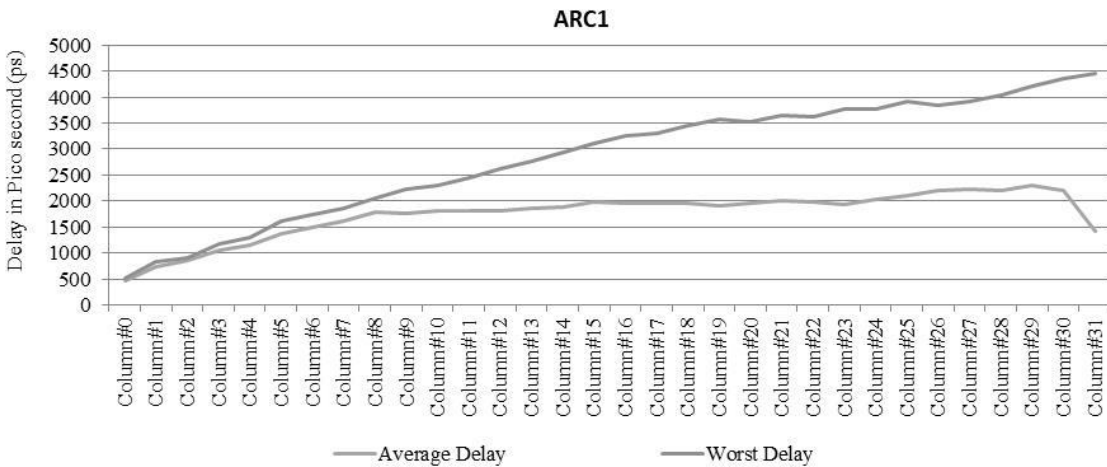
Signal arrival profile at outputs of 12-Bits multiplier

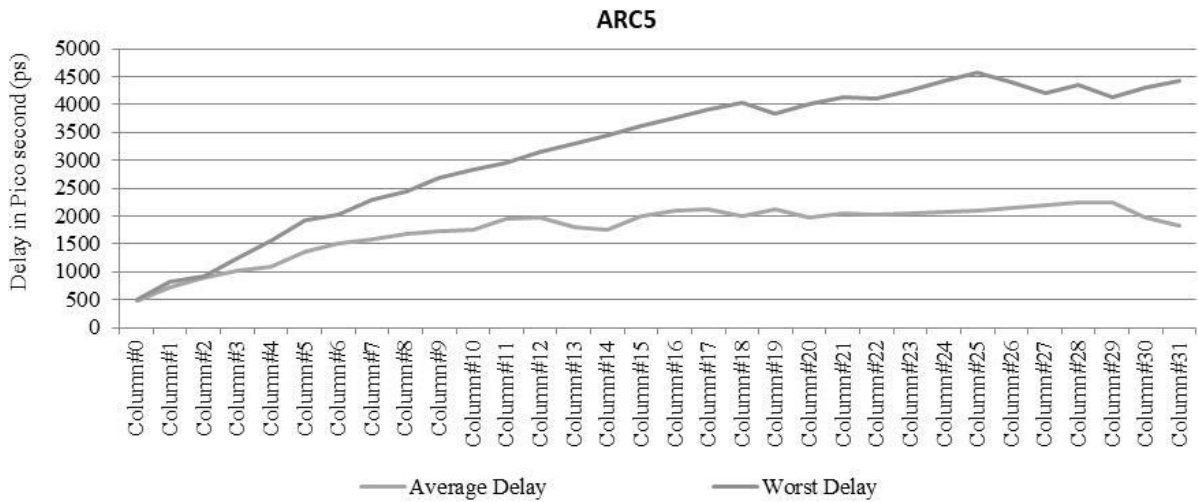
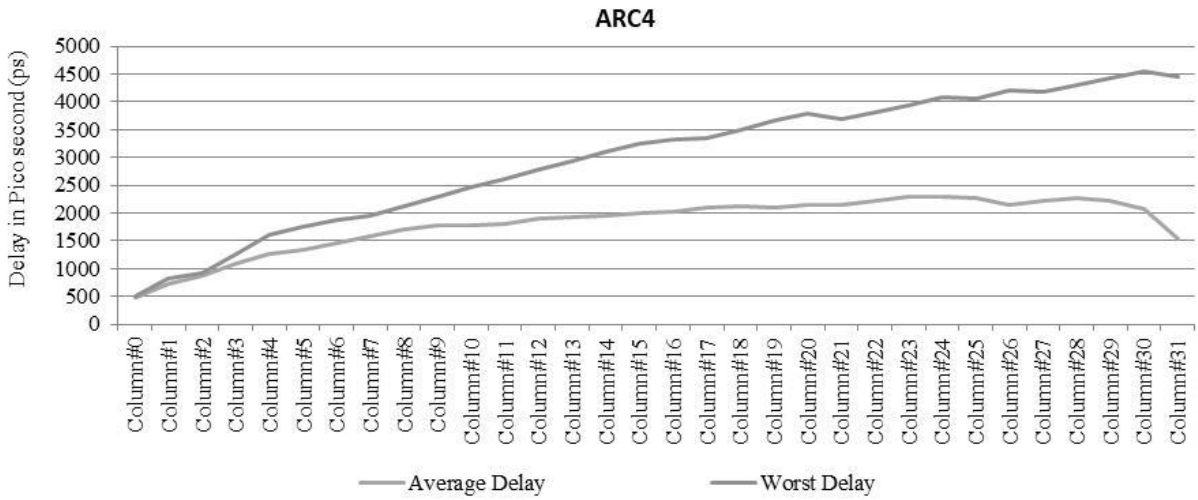
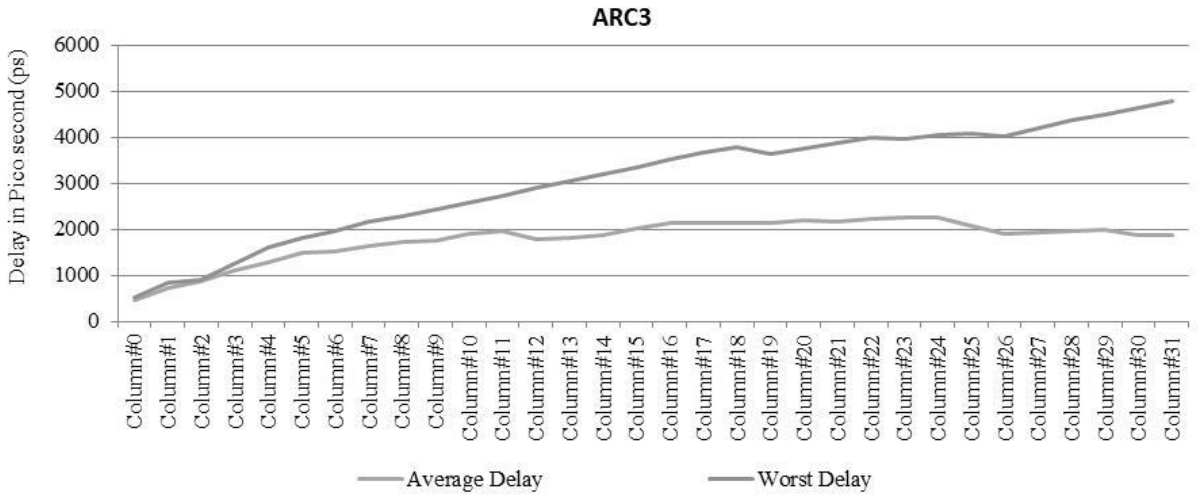






Signal arrival profile at outputs of 16-Bits multiplier





LIST OF PUBLICATIONS

Conference:

M. Rashid, A.Muhtaroglu, 'Power Delay Product Optimized Hybrid Full Adder Circuits', in *International Artificial Intelligence and Data Processing Symposium '17*, 2017, Malatya, Turkey. (**IEEE conference proceedings**)

M. Rashid, A.Muhtaroglu, 'A Novel Multiplier Design for Data Rendering', *13th IEEE BioCAS, October 19-21 in Turin, Italy*, 2017 (**IEEE, ISSC conference proceedings**)

M. Rashid, A.Muhtaroglu 'New Energy-Aware Evaluation Metric for Compression Circuits', in *the 10th International Conference on Electrical and Electronics Engineering*, 2017 Bursa, Turkey. (**IEEE conference proceedings**)