Wright State University
CORE Scholar

Browse all Theses and Dissertations | Theses and Dissertations

2022

Low-Power, Low-Cost, & High-Performance Digital Designs : Multi-bit Signed Multiplier design using 32nm CMOS Technology

N V Vijaya Krishna Boppana
Wright State University

Follow this and additional works at: https://corescholar.libraries.wright.edu/etd_all

Part of the Engineering Commons

Repository Citation
Boppana, N V Vijaya Krishna, "Low-Power, Low-Cost, & High-Performance Digital Designs : Multi-bit Signed Multiplier design using 32nm CMOS Technology" (2022). *Browse all Theses and Dissertations*. 2643.
https://corescholar.libraries.wright.edu/etd_all/2643

This Dissertation is brought to you for free and open access by the Theses and Dissertations at CORE Scholar. It has been accepted for inclusion in Browse all Theses and Dissertations by an authorized administrator of CORE Scholar. For more information, please contact library-corescholar@wright.edu.

LOW-POWER, LOW-COST, & HIGH-PERFORMANCE DIGITAL DESIGNS: MULTI-BIT SIGNED MULTIPLIER DESIGNS USING 32NM CMOS TECHNOLOGY

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

By

N V VIJAYA KRISHNA BOPPANA

M.S.Eg., Wright State University, USA, 2014

B.E., Andhra University, India, 2011

2022

Wright State University

Copyright © 2022 All Rights Reserved

N V Vijaya Krishna Boppana @ Wright State University

2022

WRIGHT STATE UNIVERSITY

GRADUATE SCHOOL

July 27, 2022

I HEREBY RECOMMEND THAT THE DISSERTATION PREPARED UNDER MY SUPERVISION BY N V Vijaya Krishna Boppana ENTITLED Low-Power, Low-Cost, & High-Performance Digital Designs: Multi-bit Signed Multiplier Designs using 32nm CMOS Technology BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Doctor of Philosophy.

Saiyu Ren, Ph.D.
Dissertation Director

Michael Saville, Ph.D., P.E.
Chair, Electrical Engineering

Barry Milligan, Ph.D.
Dean of the Graduate School

Committee on Final Examination:

Raymond E. Siferd, Ph.D.

Henry Chen, Ph.D.

Marian K. Kazimierczuk, Ph.D.

Yan Zhuang, Ph.D.

Abstract

Boppana, N V Vijaya Krishna. Ph.D., Department of Electrical Engineering, Wright State University, 2022. “Low-Power, Low-Cost, & High-Performance Digital Designs: Multi-bit Signed Multiplier design using 32nm CMOS Technology”

Binary multipliers are ubiquitous in digital hardware. Digital multipliers along with the adders play a major role in computing, communicating, and controlling devices. Multipliers are used majorly in the areas of digital signal and image processing, central processing unit (CPU) of the computers, high-performance and parallel scientific computing, machine learning, physical layer design of the communication equipment, etc. The predominant presence and increasing demand for low-power, low-cost, and high-performance digital hardware led to this work of developing optimized multiplier designs. Two optimized designs are proposed in this work. One is an optimized 8 x 8 Booth multiplier architecture which is implemented using 32nm CMOS technology. Synthesis (pre-layout) and post-layout results show that the delay is reduced by 24.7% and 25.6% respectively, the area is reduced by 5.5% and 15% respectively, the power consumption is reduced by 21.5% and 26.6% respectively, and the area-delay-product is reduced by 28.8% and 36.8% respectively when compared to the performance results obtained for the state-of-the-art 8 x 8 Booth multiplier designed using 32nm CMOS technology with 1.05 V supply voltage at 500 MHz input frequency. Another is a novel radix-8 structure with 3-bit grouping to reduce the number of partial products along with the effective partial product reduction schemes for 8 x 8, 16 x 16, 32 x 32, and 64 x 64 signed multipliers. Comparing the performance results of the (synthesized, post-layout) designs of sizes 32 x 32, and 64 x 64 based on the simple novel radix-8 structure with the estimated performance measurements for the optimized Booth multiplier design presented in this work, reduction in delay by (2.64%, 0.47%) and (2.74%, 18.04%) respectively, and reduction in area-delay-product by (12.12%, -5.17%) and (17.82%, 12.91%) respectively can be observed. With the use of the higher radix structure, delay, area, and power consumption can be further reduced. Appropriate adder deployment, further exploring the optimized grouping or compression strategies, and applying more low-power design techniques such as power-gating, multi-Vt MOS transistor utilization, multi-VDD domain creation, etc., help, along with the higher radix structures, realizing the more efficient multiplier designs.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Acknowledgements

First and foremost, I would like to acknowledge and thank my dissertation advisor and dissertation committee chair Dr. Saiyu Ren. I would also like to especially thank Dr. Saiyu Ren for understanding, supporting, and encouraging my strong desire for an independent and broad range of study/research. Thanks to be with me for the last 9+ years, since the beginning of my first master’s career at Wright State University, for listening to my ideas, for the initial years of support and encouragement towards my teaching career, for appreciating and expressing exciting views on my teaching and research at various occasions, and for spending time on reviewing my work, documents, and articles and providing the feedback. I would also like to thank my dissertation committee members, Dr. Ray Siferd, Dr. Henry Chen, Dr. Marian K. Kazimierczuk, and Dr. Yan Zhuang, and the external observer, Dr. Tarun Goswami, for spending their valuable time and effort to review my work and provide feedback.

I would like to thank all the faculty members, Dr. Saiyu Ren, Dr. Ray Siferd, Dr. John Marty Emmert, Dr. Henry Chen, and Dr. LaVern A. Starman, who taught VLSI-related subjects in my master’s in Electrical Engineering career. I would also like to thank all the faculty members, Dr. Jack S. Jean, Dr, Pradeep Misra, Dr. Ronald Taylor, Dr. Yan Zhuang, Instructor Dennis Hance, Dr. Yong Pei, Dr. John Nehrbass, Dr. Nikolaos G. Bourbakis, and Dr. Jeff Clark, who taught other subjects in my student career, Ph.D. in EE and second masters in CEG.

I would like to especially thank Dr. Raymond E. Siferd for the encouragement provided and I always keep in mind the valuable suggestions. I would also like to especially thank Dr. Marian K. Kazimierczuk for sharing some of his life experiences, achievements, and suggestions related to studies and research, which I keep in mind forever.

A special thanks to a very important person, Mike VanHorn (Senior Computer Systems Administrator) in my student, teaching, and research career. I cannot express in words how thankful I am to Mike (a very knowledgeable, humble, great human being, and an invaluable asset to WSU) for the last 9+ years of continuous help and support on numerous occasions in fixing any hardware and software-related issues. I would like to thank Simon A. (Tony) Tritschler, laboratory manager, for the help he provided me being a lab instructor and a research student.

I would like to thank the EE department chairs, Dr. Kefu Xue (before 2014) and Dr. Brian D. Rigling (2014-2017), and the current dean of the CECS, Dr. Brian D. Rigling. I would also like to thank especially the EE department staff, including academic advisors, Dr. Ryan Hamilton, Lori Luckner, Vickie Slone, Nickey Brown, Elizabeth Anne Generas, and Amanda Ellen Steward. A special thanks to Vickie Slone and Amanda Ellen Steward for their help in the final semesters of teaching and completion of the dissertation. I would also like to thank the other staff members at Wright State University, Donna J. Harris, Kimberly Ann Brumbaugh (DSO), Gwana Snell, Scott, and many others for their help. I would like to thank Erika Gilles (Associate Director of Academic Affairs, Graduate School) for the assistance provided at various occasions and for performing the format check of this dissertation document.

Thanks to all the faculty members, Dr. Henry Chen, Dr. Saiyu Ren, Dr. John M. Emmert, Instructor Dennis Hance, Dr. Gorman, Dr. Kefu Xue, Dr. Mike Saville, Dr. Pradeep Misra, Dr. Robert Allen Myers, and Dr. Marian K. Kazimierczuk, for whom I taught the labs for, for the trust, support, and appreciation. A very special thanks to my students whom I taught labs/classes for. Thanks for the cooperation provided in the lab, for making my teaching career so exciting and enjoyable, and for the love and appreciation. I will cherish the memories, accumulated over the years, forever.

Finally, I would like to thank all my family members for the unparalleled love and immense support. And, I would like to also thank the friends and well-wishers who were/are/will be always loving me and being with me.

1 Introduction

1.1 Digital Multiplier

Addition and Multiplication are the most often used elemental components of digital computing devices such as DSP, microprocessor, etc. Multiplier is the most used computer arithmetic after addition and subtraction. DSPs use multipliers for frequently used computationally intensive applications such as filtering: finite impulse response (FIR) filter, infinite impulse response (IIR) filter, and adaptive filters of types of least mean squares (LMS) filter and recursive least mean squares (RLS) filter; convolution: linear and circular convolution, Fast Fourier Transform (FFT), audio/video codecs etc. High performance computer hardware, CPUs, and GPUs, for scientific computing rely majorly on use of these fundamental digital arithmetic. Digital signal processors spend most of the time multiplying and requires more chip area of multipliers to meet the performance requirements. Multipliers often contributes towards critical path delay which in turn effects the throughput in case of pipelined designs and consumes more power in applications such as multimedia and DSP. Demand for low power consuming portable computing and communication devices such as smart watches, IoT devices, mobile phones, laptops, PCs etc., comprise of signal processing algorithms and other multiplication intense algorithms, has been increasing. Specifically, the global market for wireless portable medical devices is going to see a huge growth in next five years. According to the market research report [1], "The Global Portable Medical Devices

Market is estimated to be USD 38.1 Bn in 2021 and is expected to reach USD 68.24 Bn by 2027, growing at a CAGR of 10.2%.", and according the report [2], "Global portable medical devices market will reach $99.89 billion by 2030, growing by 9.8% annually over 2020-2030 owing to the rise in demand for portable medical devices, increase in geriatric population, growing incidences of chronic diseases, increasing government support, rising R&D investment and technological advancement.". "The global wearable medical devices market size was valued at USD 16.6 billion in 2020. It is expected to expand at a compound annual growth rate (CAGR) of 26.8% from 2021 to 2028. The growth of industries such as home healthcare and remote patient monitoring devices is anticipated to influence market growth. In addition, increasing focus on fitness and a healthy lifestyle orientation are also expected to impact the market." [3] with the revenue forecast in 2028 as USD 111.9 billion. The importance of need for inventing multiplier algorithms is supported by the statement "At least one good reason for studying multiplication and division is that there is an infinite number of ways of performing these operations and hence there is an infinite number of PhDs (or expenses-paid visits to conferences in the USA) to be won from inventing new forms of multiplier." by Alan Clements in the year 1986.

1.2 Research motivation and objective

As the usage of the digital hardware is getting increased, digital hardware become ubiquitous and a part of the gadgets, appliances, or vehicles used by and for human. With the advancements in chip manufacturing technologies such as 10nm, 7nm, and 5nm nodes, the density of the transistors is increasing and hence the power dissipation. The chip manufacturing technologies are getting matured, and the node sizes are tending to approach the atomic sizes. With billions of people using multiple digital devices and a huge number of digital devices deployed, for the direct use or for the indirect use, for the people, and the advancements of new technologies such as digital health, Inter-of-Things (IOTs), etc., the demand for the digital devices with a combination of one or more performance attributes and cost attributes such as low-area, low-power, and high-speed has been increasing. Low-power digital designs are in great demand for biomedical signal processing. This work focuses on finding the solutions for low-cost and high-speed designs at algorithmic level instead of exploring for an advanced node technology.

The objective of this work is to designing low-power, low-cost, and high-speed signed integer multipliers. The research is started with the best algorithm, Booth algorithm, for the signed number multiplication either to optimize the existing algorithm or to find new architectures. The objective includes designing the optimized algorithms using 32nm CMOS technology, performing synthesis, generating the post-layout, and comparing the performance of the modified or proposed designs with the state-of-the-art designs.

2 Literature Review

2.1 Digital Multiplier

This and the following subsection of this chapter presents the summary of the literature study on multipliers with an emphasis on signed Booth multiplier algorithms. An extensive work on optimization of multiplication circuits has been performed [4], [5] and is continuing. Standard binary multiplication uses repeated shift and accumulate routine. Therefore, the mechanisms to improve the multiplier speed involves dealing with the combination of the following process: partial product generation (PPG) & Partial product reduction (PPR), and acceleration of the accumulation of the shifted partial products (PP). Smaller number of PPs require lesser number of resources to build and hence reduces the design complexity, design area, and time & power required for the accumulation process, i.e., minimization of power-delay product (PDP) and power-delay-area (PDA) product which is also known as energy-area product (EAP). According to [5], study of various implementations of shift/add multiplications leads to a conclusion to come up with two ways to improve the speed of intrinsic multi-operand addition: **high radix multipliers** to reduce the number of operands to be added, and **tree and array multipliers** to compose multi-operand adders to minimize latency and/or maximize the throughput.

The following notation adapted from [5] is used in this discussion of multiplication algorithms:

X Multiplicand $x_{n-1}x_{n-2} \dots x_1x_0$

(2.1)

Y Multiplier $y_{n-1}y_{n-2} \cdots y_1y_0$

(2.2)

P Product $(x \times y)$ $p_{2n-1}p_{2n-2} \cdots p_1p_0$

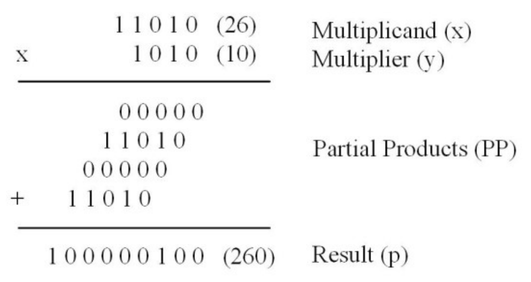
(2.3)

2.2 Multiplier Architectures

Various binary multiplier architectures such as array and tree multipliers for unsigned multiplication and the optimized booth algorithms for signed multiplication are discussed in this section.

2.2.1 Array Multiplier

An array multiplier is a combinational circuit in the shape of parallelogram used for multiplying two binary numbers, multiplicand (x) and multiplier (y), by using an array of full adders (FA) and half adders (HA) employed in simultaneous addition of the product terms generated by an array of AND gates. An example of binary multiplication is shown in Figure 1.



```
          1 1 0 1 0  (26)     Multiplicand (x)
x           1 0 1 0  (10)     Multiplier (y)
-----------------------
          0 0 0 0 0
        1 1 0 1 0             Partial Products (PP)
      0 0 0 0 0
+   1 1 0 1 0
-----------------------
    1 0 0 0 0 0 1 0 0 (260)   Result (p)
```

Figure 1. Example of a binary multiplication

An array multiplier has a simple structure and is designed by placing the FA and HA blocks horizontally, row-wise, and vertically, column-wise. Hence, the structure is more compatible to modify to pipeline structure, smaller, and requires less design time. Rows of PPs and adders are treated as stages. All stages work parallelly by processing the respective partial products before each adder while carry-out propagating to the next row. Critical path is the limiting factor in non-pipelined structure to achieve high speed or throughput. As highlighted in Figure 2 with bold line, critical path has vertical and horizontal parts with similar delay contributions in both directions by the gate delays and adder delays. The worst-case delay of the structure is proportional to the width of the multiplier i.e., for an n-bit by n-bit array multiplier the delay is nearly equal to the two times the number of full adders in a row or a slanted column. Hence, the width of an array multiplier limits the speed.

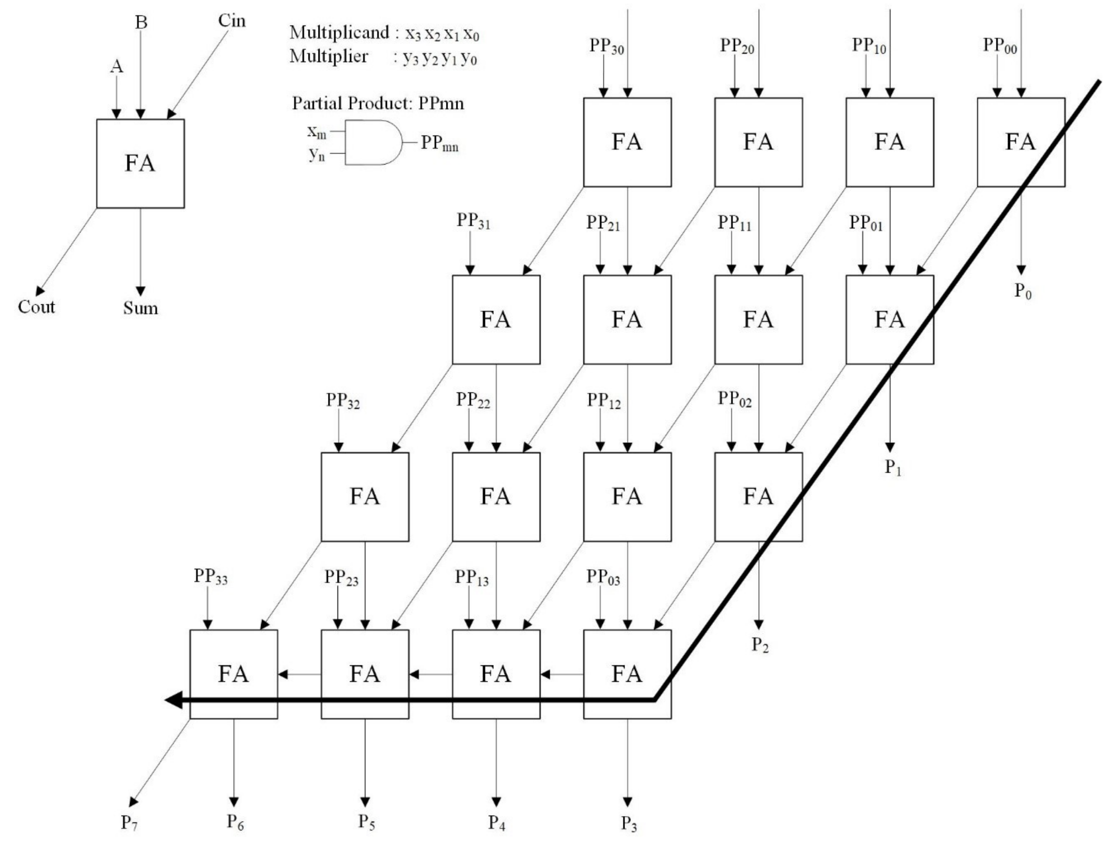


Figure 2. 4-bit x 4-bit Array Multiplier

2.2.2 Tree Multiplier [6]

Unlike the generation of PPs and accumulating the PPs in a regular array multiplier, tree structure shown in Figure 3 deploys number of full adders, each one as a 3:2 compressor, to reduce the three input bits to two output bits: sum bit and carry out bit. A full adder acts like a compressor or as an encoder by converting three binary inputs to two encoded binary output with a compression factor of 1.5. The advantage of employing the 3:2 compressors in the tree structure is that it does not involve longer carry propagation along multiple stages and hence this process is faster than the conventional way of multiplication. Summands are grouped in each step to reduce, and the process of grouping and compression continues until only two numbers remain. Total time of adding to the last step is proportional to the logarithm of the number of summands.

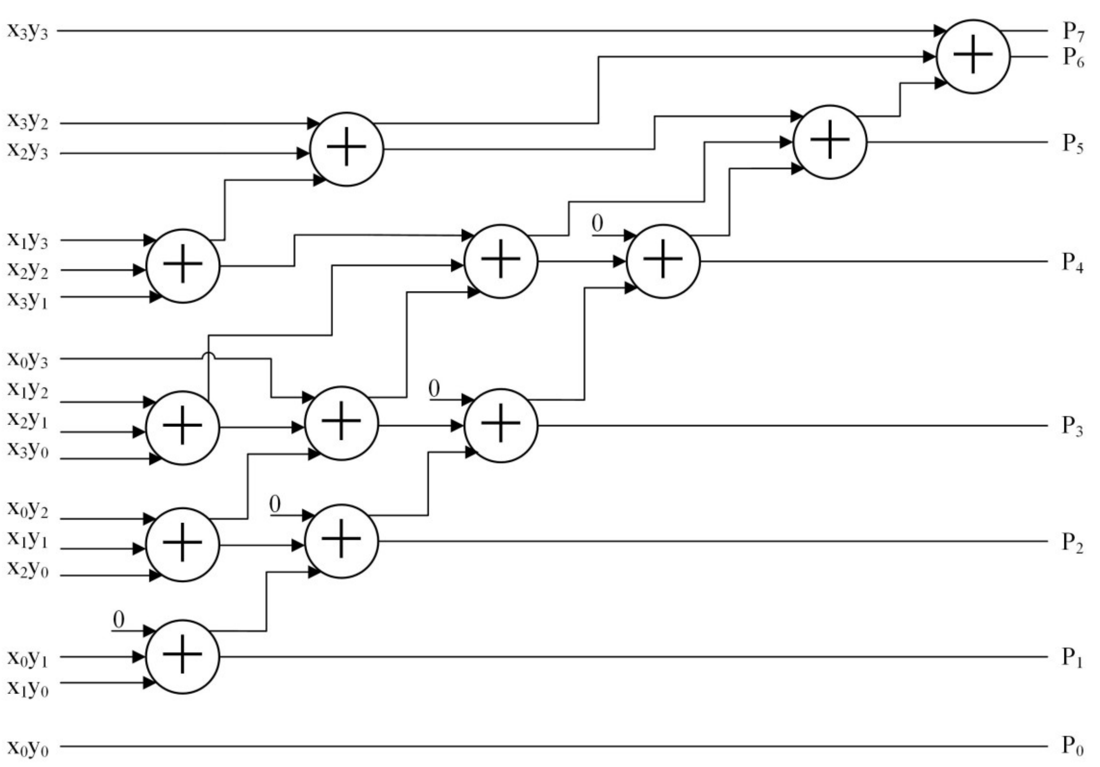


Figure 3. 4-bit x 4-bit Tree Multiplier [6]

2.2.2.1 Wallace Tree Multiplier (WTM) [7]

Wallace tree structure using carry-save adders to sum the multiplicand-multiples in parallel is shown in Figure 4. A carry save adder takes three binary inputs and yields two binary outputs. The advantage of the Wallace tree multiplier is its tree like structure with the carry save adders which ensures less delay due to the reduced number of logic levels. The disadvantage of this structure is its complex structure to design the layout and very high hardware requirement.

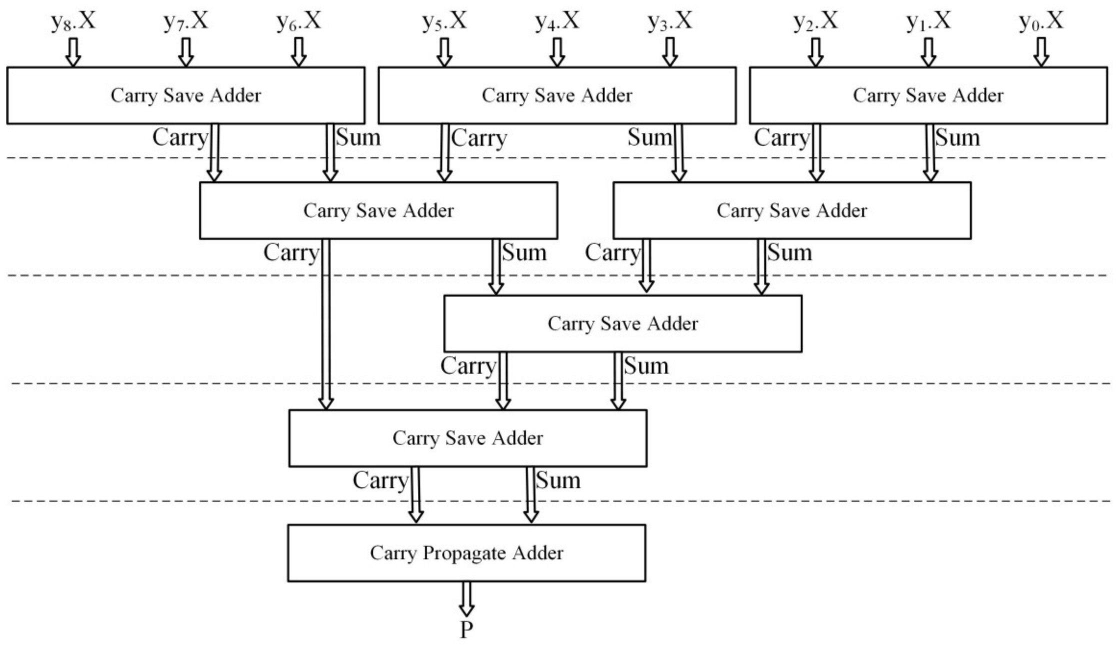


Figure 4. Wallace Tree Multiplier using Carry Save Adders [6]

The essential feature of Wallace tree architecture is to find the final product using least number of possible steps. Hence, at each step maximum number of bits, shown as solid dots in [8], are covered in each step vertically to compress using half adder to encode two bits and full adder to encode three bits. This process is illustrated in Figure 5. Any bits uncovered, groups of two or three, are transferred to the next stage without any further processing. As mentioned earlier, half adder acts like a 2:2 compressor and full adder like a 3:2 compressor. The steps are continued until only two or a smaller number of bits remaining at each bit position at the end of the completion of a step. All the compressions are carried out in parallel at each step. At each step, the lone bit on the least significant bit (LSB) positions or on the most significant bit (MSB) positions are transferred down to the next step without any processing.

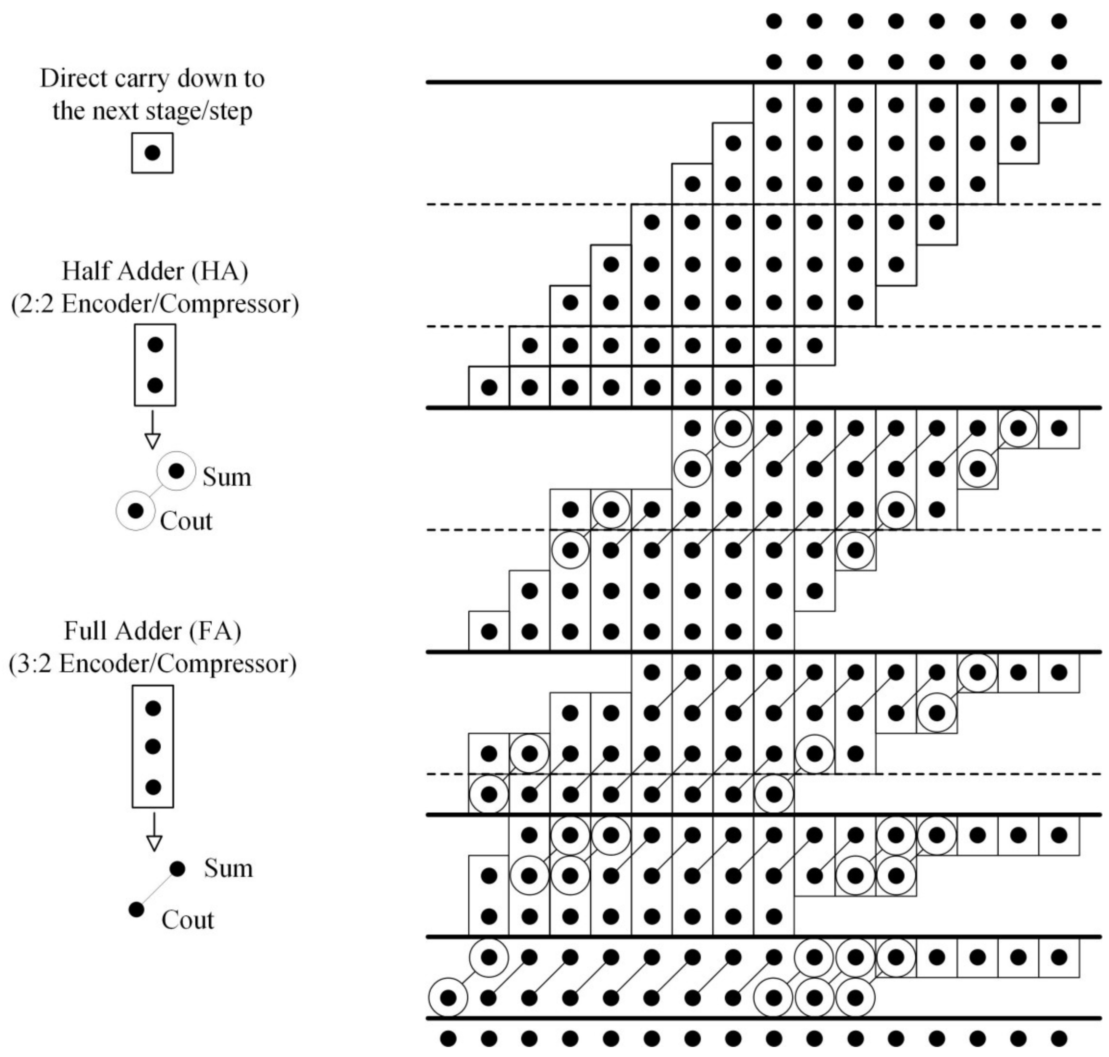


Figure 5. Dot Diagram for an 8-bit x 8-bit Wallace Tree Multiplier [8]

Sum and carry out output bits obtained from the full adder in the current step are depicted using a diagonal line joining two solid dots in the next stage. Similarly, the two output bits of the half adder are shown in the next stage using an inclined solid line of joining two circles enclosing the respective bit of a partial product, solid dot. Recursive equations used to determine the height of the j^{th} reduction stage, ω_j, where j starts from 0, are shown below [9] using equations (2.4) and (2.5).

$$\omega_0 = N$$

(2.4)

$$\omega_{j+1} = 2 \cdot \left\lfloor \frac{\omega_j}{3} \right\rfloor + \omega_j \; mod \; 3$$

(2.5)

In the dot diagram shown in Figure 5 for an 8 x 8 Wallace multiplier, four reduction stages can be observed with heights of 6, 4, 3, and 2 respectively. And, the total digital hardware required for the multiplication includes 64 AND gates, 1 OR gate, 38 3:2 compressors, 15 2:2 compressors, and 10-bit carry propagation adder (CPA). The number of 2:2 compressors required are either equal or greater than N and often much greater than N. Given the number of bits of operands, N, and calculating the number of stages of reduction to reduce the PP matrix from N-rows to 2-rows, S, the number of 3:2 compressors and the size of the final CPA adder can be determined by the following criteria [10] shown using equations (2.6) to (2.10).

$3 \le N \le 5$

$$\text{(3,2) counters} = N^2 - 4 \cdot N + 3 + S$$

(2.6)

$$\text{CPA length} = 2 \cdot N - 2 - S$$

(2.7)

$5 < N$

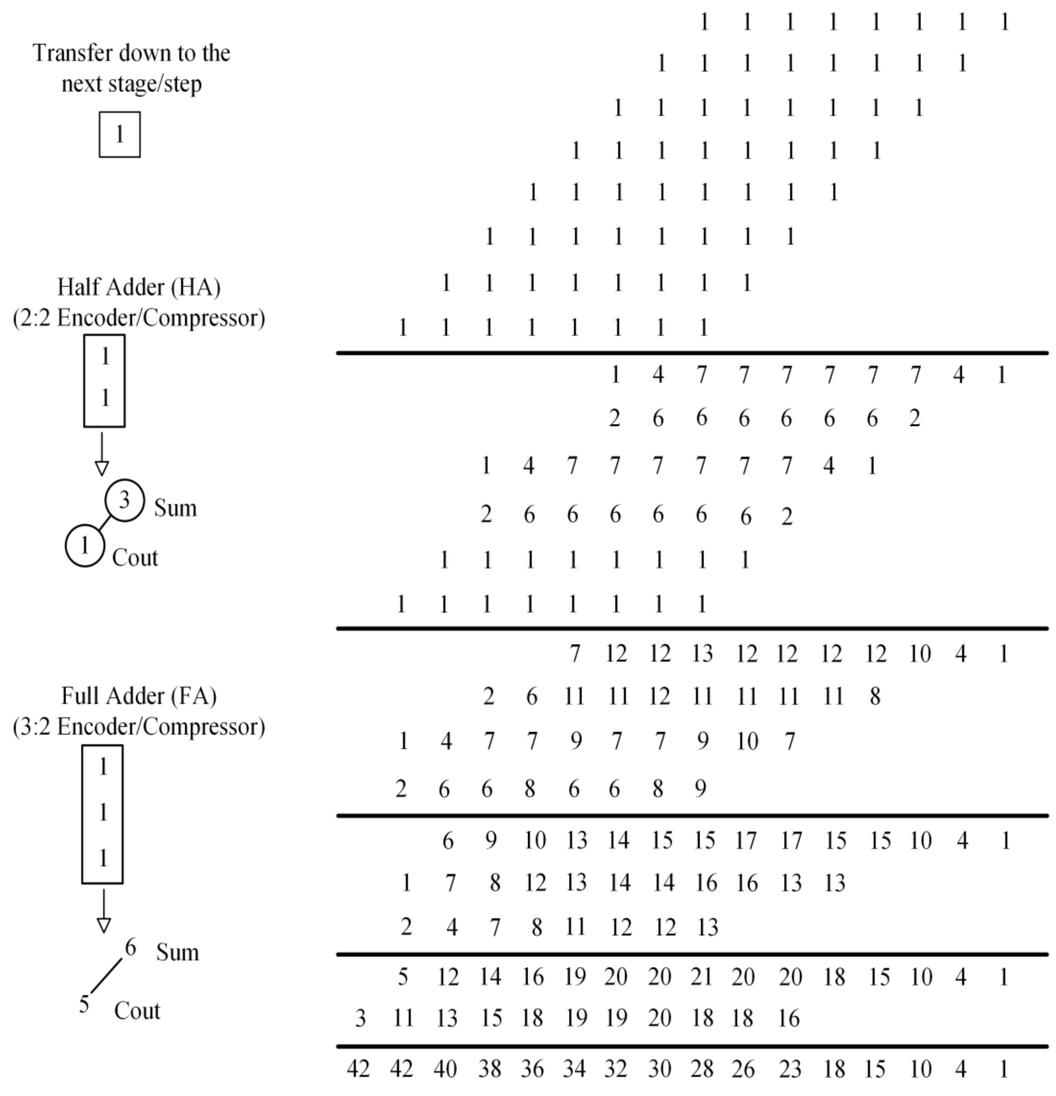
$$\text{(3,2) counters} = N^2 - 4 \cdot N + 2 + S$$

(2.8)

or

$$(3,2)\text{ counters} = N^2 - 4 \cdot N + 1 + S \tag{2.9}$$

$$\text{CPA length} = 2 \cdot N - 1 - S \tag{2.10}$$



Transfer down to the next stage/step
1
Half Adder (HA) (2:2 Encoder/Compressor)
Sum
Cout
Full Adder (FA) (3:2 Encoder/Compressor)
Sum
Cout

Figure 6 Delay Diagram of an 8 x 8 Wallace Multiplier with RCA as a Final Adder [8]

Discussing about the delay estimation, all N^2 partial products are generated in parallel by exercising the bit-by-bit multiplication using a simple two input AND gate. Hence, the delay contributed by PP generation is of $O(1)$ complexity. Delay estimation for the Wallace multiplier is performed in [8] considering a nine gate full adder, comprised of only 2-input AND, 2-input OR, and inverter gates, as 3:2 compressor and a half adder, comprised of four 2-input gates, as 2:2 compressor while considering that all the two input standard cells having nearly equivalent area, gate count, and a delay of 1. Assuming the simultaneous arrival of all the input signals to the compressors, the delays of the sum, LSB output bit, and carry out, MSB output bit, signals for the full adder as 3:2 compressor are 6 gate delays and 5 delays respectively, and the delays of sum and carry-out output signals of the half adder, as 2:2 compressor, are 3 and 1 gate delays respectively. Similarly, the delay estimation for the Dadda tree multiplier discussed in the following sub-section relies on the gate count and delay consideration made above. The following delay diagram for an 8 by 8 Wallace multiplier with an RCA as final adder is presented in [8], shown in Figure 6, with detailed explanation.

A low power and scalable counter-based modular Wallace tree (CBMW) multiplier is presented in [11]. Partial products are reduced using a power efficient sequential 7:3 counter, composed of multiplexer and XOR, and by applying multibit addition in a single column. Power consumption is reduced by deploying a single 7:3 counter to perform the partial product reduction in each column. Only a single 7:3 counter used per each stage of partial product reduction applying inputs serially. 7:3 counter presented in [12], which is an efficient 7:3 counter than the ones presented in [13], [14], [15], and [16], is used. The performance of the CBMW multiplier is compared in terms of power, delay, total cell area, and PDP against the other variants such as conventional WTM, Reduced Complexity Wallace (RCW) tree multiplier [17], Counter-Based Wallace (CBW) tree multiplier [18] [16], etc.,.

Ever increasing use of the Booth-encoding in reduction of number of partial products led to the work in [19] to perform the performance comparison between RCW multiplier and radix-4 Booth-Reduced Complexity Wallace (R4B-RCW) multiplier and the synthesis results clearly show that the RCW performs significantly better than the R4B-RCW multiplier in terms of both speed and power consumption.

2.2.2.2 Dadda Tree Multiplier [20]

Dadda tree binary multiplier design is an optimized scheme invented by computer scientist, Luigi Dadda, in 1965 to compute the multiplication of the unsigned fixed-point numbers. Like Wallace tree structure, Dadda tree multiplier is also a column compression multiplier consists of three stages: Partial product matrix formation in stage 1, reduction of partial product matrix height to 2 in stage 2, and accumulation of these final two rows using carry propagation adder in the final stage.

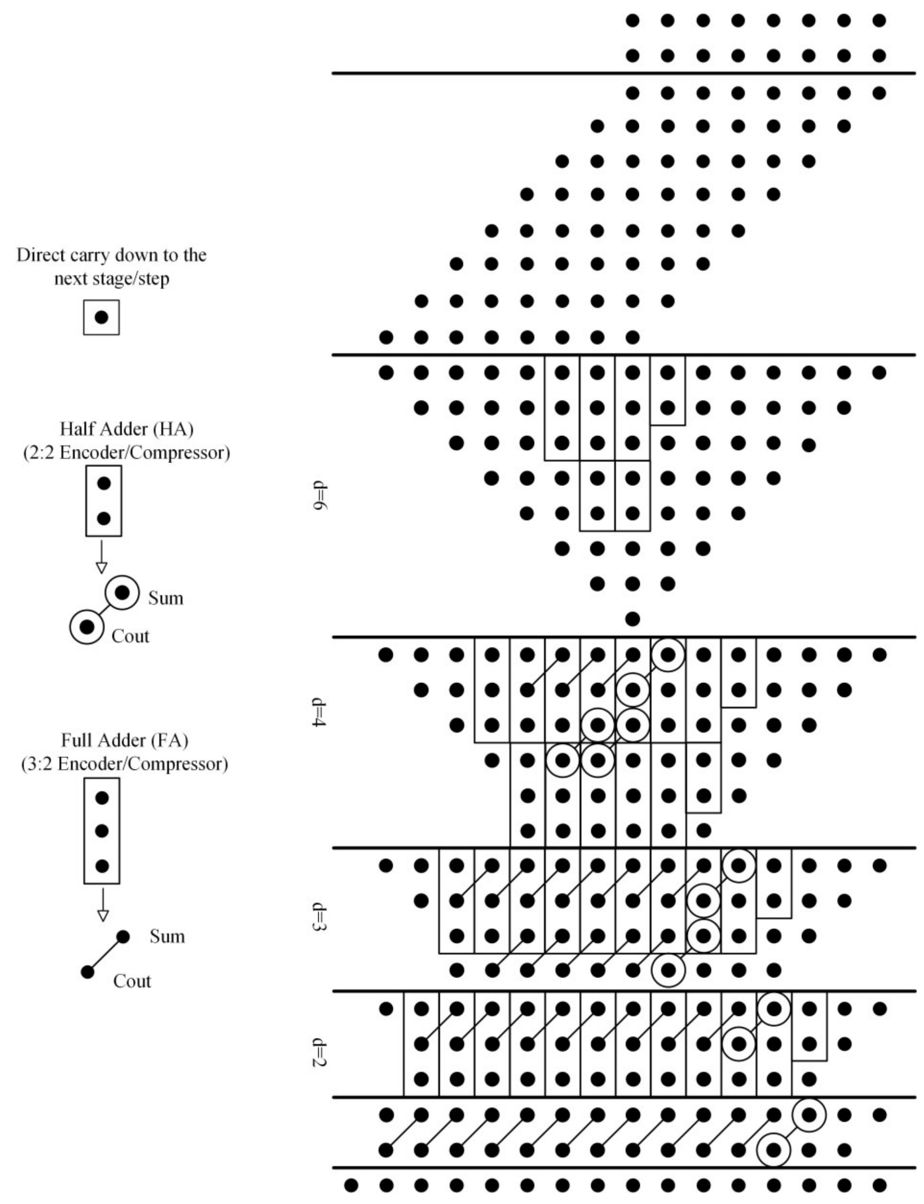

Direct carry down to the next stage/step
Half Adder (HA) (2:2 Encoder/Compressor)
Sum
Cout
Full Adder (FA) (3:2 Encoder/Compressor)
Sum
Cout
d=6
d=4
d=3
d=2

Figure 7. Dot Diagram for an 8-bit x 8-bit Dadda Tree Multiplier [8]

Dadda tree multiplier has same number of reduction levels as Wallace tree multiplier with variation of matrix height at different levels. The number of full adders in both the tree structures is nearly the same. However, Wallace adder uses more of the full adders and a greater number of half adders in the reduction levels leading to a shorter final CPA adder compared to the Dadda multiplier. Dadda multiplier use a minimal number of compressors in each level of compression to achieve the required compression and the recursive reduction procedure is as follows [21] [8]:

1. Starting from $d_1 = 2$ as the final stage of reduction and Nas the height of the original PP matrix, calculate d_j, the height of the matrix at j^{th} level of reduction from the bottom, using $d_j = \left\lfloor \frac{3}{2} \cdot d_{j-1} \right\rfloor$. Repeat calculating the d_j until reaching the largest matrix of at j^{th} level where $d_j < N < d_{j+1}$.

2. Starting from the highest j^{th} stage from the end, matrix in each stage needed to be reduced using (3,2) and (2,2) counters to the desired heights calculated in the previous step. Reduction should be performed on only columns with dots greater than the required stage height, d_j. Carries coming from the least significant (3,2) and (2,2) counters are needed to consider as dots while reducing.

3. Repeat the reduction procedure in step two on each stage until reaching the final stage of height $d_1 = 2$.

The dot diagram for 8 by 8 Dadda multiplier is shown in Figure 7. All three stages involving PP matrix generation, matrix reduction, and final CPA additions are shown in the figure. In stage 2 of reduction, each of four levels of reduction are labeled with the respective matrix height reduction requirement as 6, 4, 3, and 2 from the top to the bottom level. Digital hardware required to build the Dadda multiplier includes 64 AND gates in the first stage, 35 3:2 compressors and 7 2:2 compressors in the second stage, and a 14-bit CPA adder in the final stage. Given N as the number of bits of operands, the number of (3,2) counters, (2,2) counters, and the size of the final CPA adder are determined as follows:

$$(3,2) \text{ counters} = N^2 - 4 \cdot N + 3$$

(2.11)

$$(2,2) \text{ counters} = N - 1$$

(2.12)

$$\text{CPA length} = 2 \cdot N - 2$$

(2.13)

The article [8], includes the discussion of delay estimation methodology, concluded about the delay comparison based on the closed examination performed between Dadda and Wallace multipliers that the general assumption of slightly faster response of Wallace multiplier due to the smaller final stage adder is incorrect. The delay calculation diagram is depicted in Figure 8 and the results presented in [8] by performing detailed analysis on both tree structures varying the operand sizes with both RCA and CLA are shown in Table 2.1 and concluded that the delay and complexity of Dadda multiplier is less compared to Wallace multiplier.

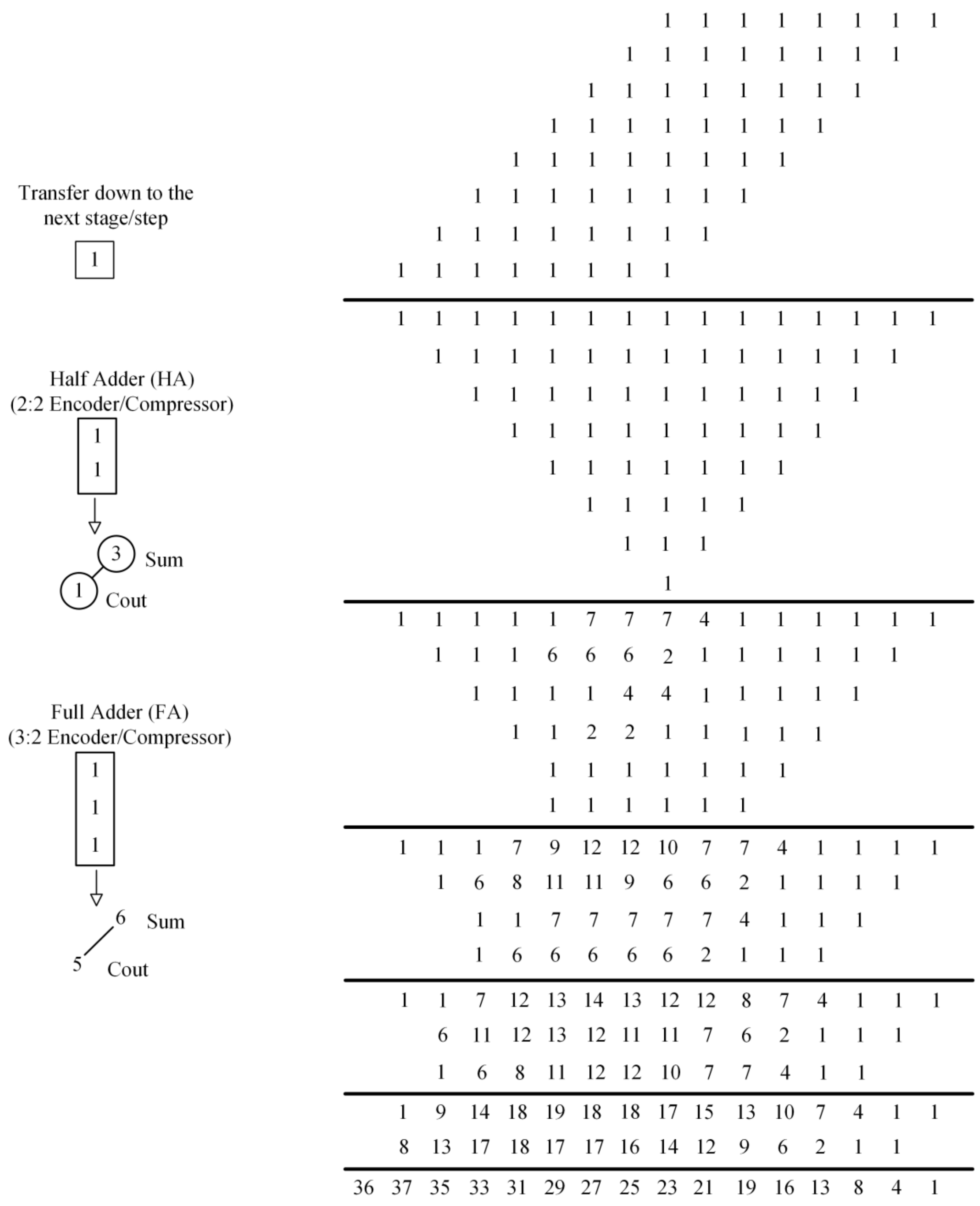

Transfer down to the next stage/step
Half Adder (HA) (2:2 Encoder/Compressor)
Sum
Cout
Full Adder (FA) (3:2 Encoder/Compressor)
Sum
Cout
36 37 35 33 31 29 27 25 23 21 19 16 13 8 4 1

Figure 8. Delay Diagram of an 8 x 8 Dadda Multiplier with RCA as Final Adder [8]

Table 2.1. Delay and Complexity Comparisons for various sizes of Dadda and Wallace Multipliers with RCA and CLA presented in [8]

Multiplier Size	Delay		Complexity	
	Dadda	Wallace	Dadda	Wallace
with RCAs				
4 x 4	19 (100%)	21 (111%)	104 (100%)	104 (100%)
8 x 8	37 (100%)	42 (114%)	528 (100%)	552 (105%)
16 x 16	69 (100%)	77 (112%)	2336 (100%)	2476 (106%)
32 x 32	133 (100%)	145 (109%)	9792 (100%)	10283 (105%)
with CLAs				
4 x 4	15 (100%)	18 (120%)	120 (100%)	112 (93%)
8 x 8	29 (100%)	31 (107%)	573 (100%)	582 (102%)
16 x 16	43 (100%)	45 (105%)	2440 (100%)	2557 (105%)
32 x 32	54 (100%)	56 (104%)	10013 (100%)	10475 (105%)

Signed Radix-2^m parallel multipliers with two new sign extension techniques to improve the energy efficiency with smaller design area presented in [22] with partial product compression performed in both Wallace and Dadda styles deploying two variants, one employing RCA-less optimization (NR) and the second one employing optimized sign extension without intermediary ripple carry adders (NR-SO) for multipliers of operand widths 8, 16, 32, and 64.

An efficient signed carry-save multiplier (CSM) with modified square root carry-select adder (MSCA), instead of conventional vector-merging adder (CVMA), for the vector-merging addition and improved full adder (IFA), in place of regular full adder, is presented in [23]. 8-bit and 16-bit wide multipliers are designed, synthesized, and made comparisons with the state-of-the-art designs to show that there is a remarkable improvement in the performance metrics, critical path delay (CPD), power, PDP, area, and ADP.

2.2.3 Booth Multiplier

A signed multiplier technique is presented in [4] by Andrew D. Booth hence the name Booth multiplier. Booth multiplier architecture works mainly on the partial product generation using the Booth encoding scheme in Table 2.2. Though the encoding scheme looks like using three bits of the multiplier, the effective number of bits used at any stage of partial product generation are two and hence the encoding scheme is a radix-4 scheme.

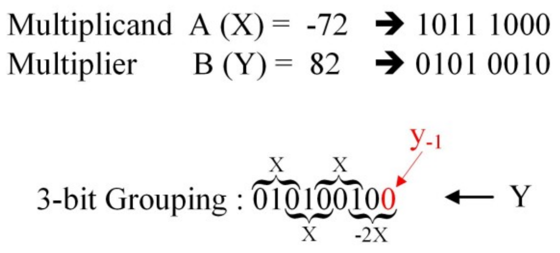

Multiplicand A (X) = -72 → 1011 1000
Multiplier B (Y) = 82 → 0101 0010
y_{-1}
X
X
3-bit Grouping : 010100100
← Y
X
-2X

Figure 9 3-bit grouping performed in radix-4 Booth multiplier

The grouping, shown in Figure 10, and encoding starts from the LSB side and proceeds towards the MSB side of the multiplier. First 3-bit grouping on the LSB side includes a padded bit-0 at -1th location, a pseudo index, and the next two bits include the LSB bits at 0th and 1st bit positions of the multiplier. The second 3-bit grouping include the MSB bit of the previous 3-bit grouping as an LSB bit and the next two bits include the adjacent bit values proceeding towards the MSB side, i.e., bits at indexes 2 and 3 respectively. This grouping process continues until the end of the multiplier bits. Even number of multiplier bits result in complete grouping of all the bits. Since the effective number of bits used per group are two and the number of groups equal to half the total number of multiplier bits.

Table 2.2 Radix-4 Booth encoding scheme [4]

y_{2i+1}	y_{2i}	y_{2i-1}	Partial Product
0	0	0	0
0	0	1	$+X$
0	1	0	$+X$
0	1	1	$+2X$
1	0	0	$-2X$
1	0	1	$-X$
1	1	0	$-X$
1	1	1	0

A high-speed parallel Booth multiplier with new modified Booth encoding (MBE) scheme to achieve better performance than the traditional MBE schemes, modified PPA, and a new addition algorithm, multiple-level conditional-sum adder (MLCSMA), to perform the final addition is presented in [24]. One of the two common methods of generating partial products in the first stage is using radix-4 MBE [4] [25] which reduces the number of partial products by a factor of two. Area and delay profiles of new MBE scheme proposed is compared with several other existing MBE schemes and can be observed that the proposed scheme is faster while requiring moderate design area. Partial product reduction tree (PPRT) is used to effectively sum up all the partial products generated. Also, examined and concluded that the parallel multiplier constructed using Three-Dimensional-reduction-Method (TDM) [26] [27] with MBE is faster with smaller area. Partial product reduction using 4:2 compressor is faster compared to using Wallace tree and Carry-save tree which uses 3:2 compressor, full adder, as a basic element, whereas, the TDM outperforms the 4:2 compressor in speed. New MLCSMA presented and used is constructed using conditional-sum adder (CSMA) and the conditional-carry adder (CCA) and showed an improvement of up to 25% performing final addition when designed in 350nm technology at supply voltage of 3.3 V. Since the tree based CSMA is a very regular structure with the performance compared to CLA adder. The hybrid adder structure retains the speed from by using the conventional CSMA [28] and saves area using CCA [29].

Modified Booth algorithms performing speed critical wide operand multiplications with very high radix structure accompanied by deployment of reduced area adder trees is presented in [30] resulting in large increase in speed with reasonable design area. Another modified Booth algorithm with optimized radix-4 Booth encoders for partial product generation and effective use of (3,2), (5,3), and (7,4) compressors for partial product reduction in vertical direction is presented in [31].

2.2.4 Baugh-Wooley Two's Complement Signed Multiplier [32]

A high-speed two's complement m-bit by n-bit parallel array multiplier for signed multiplication, also known as Baugh-Wooley multiplier, is presented in 1973 [32] with the focus on the solving the problems caused by the sign bits in signed number multiplication using the most common two's complement representation. Conventional two's complement multiplier contains partial products with both the positive and negative signs. However, the Baugh-Wooley multiplier has an advantage over the conventional two's complement multiplier is that the signs of all the partial product bits are positive. Conventional two's complement binary multiplication, shown in Figure 10, multiplies the multiplicand $X = (x_{m-1}x_{m-2} \dots x_1x_0)$ with multiplier $Y = (y_{n-1}y_{n-2} \dots x_1x_0)$ resulting in the $m + n$ bit product $P = (p_{m+n-1}p_{m+n-2} \dots p_1p_0)$. Product is the result of the sum of the partial product bits formed by AND each of the multiplicand bit with the multiplier bit.

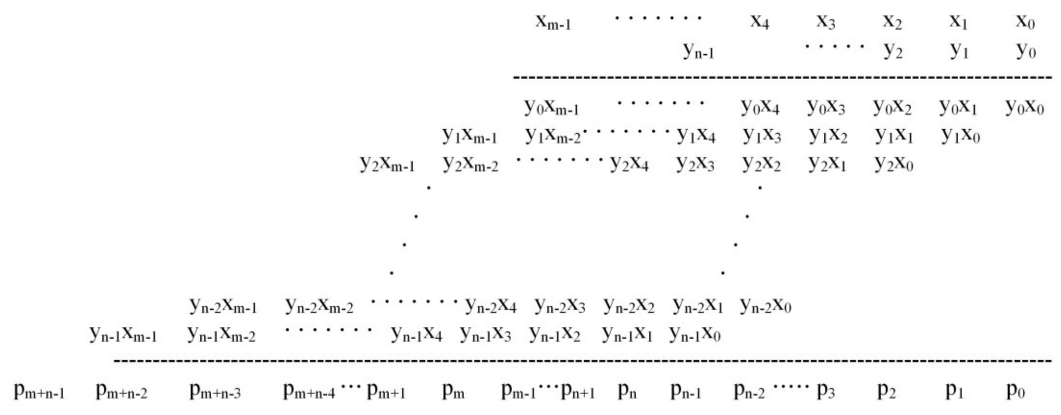
Let the values of the X and Y be X_v and Y_v which are given by the following equations (2.14) and (2.15) included in [32].

$$X_v = -x_{m-1}2^{m-1} + \sum_{i=0}^{m-2} x_i 2^i$$

(2.14)

$$Y_v = -y_{n-1}2^{n-1} + \sum_{j=0}^{n-2} y_j 2^j$$

(2.15)



x_{m-1} .... x_4 x_3 x_2 x_1 x_0
y_{n-1} ..... y_2 y_1 y_0
y_0x_{m-1} .... y_0x_4 y_0x_3 y_0x_2 y_0x_1 y_0x_0
y_1x_{m-1} y_1x_{m-2} .... y_1x_4 y_1x_3 y_1x_2 y_1x_1 y_1x_0
y_2x_{m-1} y_2x_{m-2} .... y_2x_4 y_2x_3 y_2x_2 y_2x_1 y_2x_0
$y_{n-2}x_{m-1}$ $y_{n-2}x_{m-2}$ .... $y_{n-2}x_4$ $y_{n-2}x_3$ $y_{n-2}x_2$ $y_{n-2}x_1$ $y_{n-2}x_0$
$y_{n-1}x_{m-1}$ $y_{n-1}x_{m-2}$ .... $y_{n-1}x_4$ $y_{n-1}x_3$ $y_{n-1}x_2$ $y_{n-1}x_1$ $y_{n-1}x_0$
p_{m+n-1} p_{m+n-2} p_{m+n-3} p_{m+n-4} ... p_{m+1} p_m p_{m-1} ... p_{n+1} p_n p_{n-1} p_{n-2} ..... p_3 p_2 p_1 p_0

Figure 10 Conventional two's complement binary multiplication [32]

Let the value of the product P be P_v, and is represented by the equation (2.16).

$$P_v = -p_{m+n-1}2^{m+n-1} + \sum_{i=0}^{m+n-2} p_i 2^i = X_v Y_v$$

$$= \left(-x_{m-1}2^{m-1} + \sum_{i=0}^{m-2} x_i 2^i \; Y_v\right)\left(-y_{n-1}2^{n-1} + \sum_{j=0}^{n-2} y_j 2^j\right)$$

$$= \left(x_{m-1}x_{n-1}2^{m+n-2} + \sum_{i=0}^{m-2}\sum_{j=0}^{n-2} x_i y_j 2^{i+j}\right)$$

$$- \left(\sum_{j=0}^{n-2} x_{m-1}y_j 2^{m-1+j} + \sum_{i=0}^{m-2} y_{n-1}x_i 2^{n-1+i}\right)$$

(2.16)

The above equation has partial products needed to be added and subtracted. The partial products with negative signs must be two's complemented to perform the addition instead of subtraction. Assuming the magnitude of a two's complement number Z is Z_v. Value of the negation of the two's complement number $Z = (z_{k-1}, \cdots, z_0)$ is as follows:

$$-Z_v = 2's\ complement\ of\ Z$$

$$= 1's\ complement\ of\ Z + 1$$

$$= -\overline{z_{k-1}}2^{k-1} + \sum_{i=0}^{k-2} \overline{z_i} 2^i + 1$$

(2.17)

Therefore, the subtraction terms in the product equation,

$$2^{m-1}\left(-0\cdot 2^{n}+0\cdot 2^{n-1}+\sum_{j=0}^{n-2} y_j x_{m-1} 2^{j}\right)$$

(2.18)

and

$$2^{n-1}\left(-0\cdot 2^{m}+0\cdot 2^{m-1}+\sum_{i=0}^{m-2} y_{n-1} x_i 2^{i}\right)$$

(2.19)

are replaced with

$$2^{m-1}\left(-1\cdot 2^{n}+1\cdot 2^{n-1}+\left(\sum_{j=0}^{n-2} \overline{y_j x_{m-1}} 2^{j}\right)+1\right)$$

(2.20)

and

$$2^{n-1}\left(-1\cdot 2^{m}+1\cdot 2^{m-1}+\left(\sum_{i=0}^{m-2} \overline{y_{n-1} x_i} 2^{i}\right)+1\right)$$

(2.21)

respectively. Thus, the last two partial product rows, shown in Figure 11,

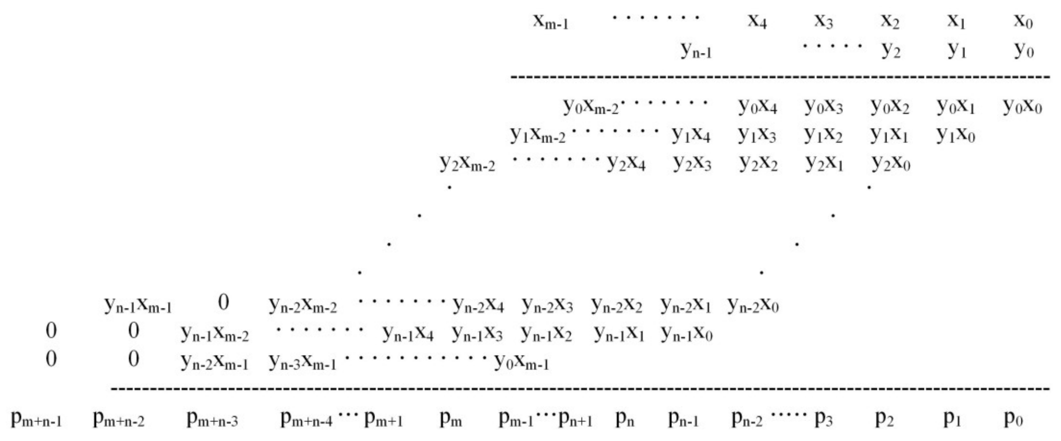


Figure 11 Positive and Negative segregation of PP bits [32]

which need to be subtracted,

$$0\ \ 0\ \ y_{n-1}x_{m-2}\ \ y_{n-1}x_{m-3}\ \ \cdots\ \ y_{n-1}x_0 \tag{2.22}$$

and

$$0\ \ 0\ \ y_{n-2}x_{m-1}\ \ y_{n-3}x_{m-1}\ \ \cdots\ \ y_0x_{m-1} \tag{2.23}$$

are replaced by

$$1\ \ 1\ \ \overline{y_{n-1}x_{m-2}}\ \ \overline{y_{n-1}x_{m-3}}\ \ \cdots\ \ \overline{y_{n-1}x_0} \tag{2.24}$$

and

$$1\ \ 1\ \ \overline{y_{n-2}x_{m-1}}\ \ \overline{y_{n-3}x_{m-1}}\ \ \cdots\ \ \overline{y_0x_{m-1}} \tag{2.25}$$

respectively. The additional “1”s are added to the p_{m-1} column and p_{n-1} column. And, the non-uniformity occurred in the last two rows due to the need for using NAND instead of AND to form the partial product bits can be avoided by using the following equivalences (2.26) and (2.27),

$$\begin{cases} 0, & for\ y_{n-1} = 0 \\ 2^{n-1}\left(-2^m + 2^{m-1} + \left(\sum_{i=0}^{m-2} \overline{x_i}2^i\right) + 1\right), & for\ y_{n-1} = 1 \end{cases}$$

(2.26)

and

$$\begin{cases} 0, & for\ x_{m-1} = 0 \\ 2^{m-1}\left(-2^n + 2^{n-1} + \left(\sum_{j=0}^{n-2} \overline{y_j}2^j\right) + 1\right), & for\ x_{m-1} = 1 \end{cases}$$

(2.27)

Following the equivalences (2.26) and (2.27), the above equations (2.20) and (2.21)can be rewritten as follows:

$$2^{n-1}\left(-2^m + 2^{m-1} + \overline{y_{n-1}}2^{m-1} + y_{n-1} + \left(\sum_{i=0}^{m-2} y_{n-1}\overline{x_i}2^i\right)\right)$$

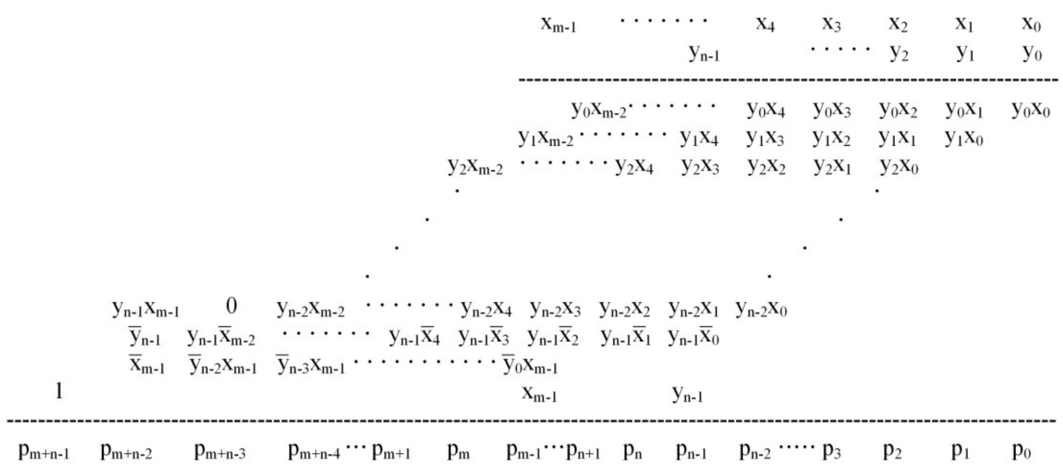
(2.28)

and

$$2^{m-1}\left(-2^n + 2^{n-1} + \overline{x_{m-1}}2^{n-1} + x_{m-1} + \left(\sum_{j=0}^{n-2} x_{m-1}\overline{y_j}2^j\right)\right)$$

(2.29)

A simplified proof for Baugh-Wooley two's complement parallel array multiplier is presented in [33]. After achieving the uniformity at the last two rows and adding the constant terms, the rewritten partial product array with complete uniformity is shown in Figure 12.



x_{m-1} ······· x_4 x_3 x_2 x_1 x_0
y_{n-1} ····· y_2 y_1 y_0
y_0x_{m-2} ······ y_0x_4 y_0x_3 y_0x_2 y_0x_1 y_0x_0
y_1x_{m-2} ······· y_1x_4 y_1x_3 y_1x_2 y_1x_1 y_1x_0
y_2x_{m-2} ······· y_2x_4 y_2x_3 y_2x_2 y_2x_1 y_2x_0
$y_{n-1}x_{m-1}$ 0 $y_{n-2}x_{m-2}$ ······· $y_{n-2}x_4$ $y_{n-2}x_3$ $y_{n-2}x_2$ $y_{n-2}x_1$ $y_{n-2}x_0$
$\overline{y}_{n-1}$ $y_{n-1}\overline{x}_{m-2}$ ······· $y_{n-1}\overline{x}_4$ $y_{n-1}\overline{x}_3$ $y_{n-1}\overline{x}_2$ $y_{n-1}\overline{x}_1$ $y_{n-1}\overline{x}_0$
$\overline{x}_{m-1}$ $\overline{y}_{n-2}x_{m-1}$ $\overline{y}_{n-3}x_{m-1}$ ··········· $\overline{y}_0x_{m-1}$
1 x_{m-1} y_{n-1}
p_{m+n-1} p_{m+n-2} p_{m+n-3} p_{m+n-4} ··· p_{m+1} p_m p_{m-1}···p_{n+1} p_n p_{n-1} p_{n-2} ····· p_3 p_2 p_1 p_0

Figure 12 Baugh-Wooley algorithm with all positive PP bits [32]

A new high speed digital multiplier using modified pairwise and parallel addition algorithms is presented in [34] to improve the speed and the simulation results show that the speed and delay performance is two folds better than the conventional linear array multipliers. The delay and power consumption for this design is significantly better than the designed and optimized Baugh-Wooley multiplier.

2.2.5 Vedic Multiplier

Vedic multiplier, which is based on Urdhva – Tiryakbhyam (UT) (vertical and crosswise) formulae, with fast adders (carry save adder, Brenk-Kung adder, and carry select adder) and by deploying compressors in place of full-adders and half adders to minimize the power-delay-product (PDP) is proposed in [35]. Multipliers with operand width of 8-bit and 16-bit are designed, synthesized, and made proper comparisons with the synthesis results obtained from designing and synthesizing Ripple carry based Vedic multiplier (RCVM) [36], Carry Save array multiplier (CSAM) [37], compressor-based Vedic multiplier (CVM) [38], and modified Booth encoded Wallace tree multiplier (MBWM) [39]. Results comparison clearly shows that optimized Vedic multiplier presented shows significant improvement in delay and PDP with the area and power tradeoff. Another optimized Vedic multiplier with adaptable Manchester carry chain (MCC) adder, implemented with adaptable clocking scheme to be suitable to extend the use towards wider multipliers, for low power-delay product is presented in [40].

3 Optimization of radix-4 8 x 8 Booth multiplier

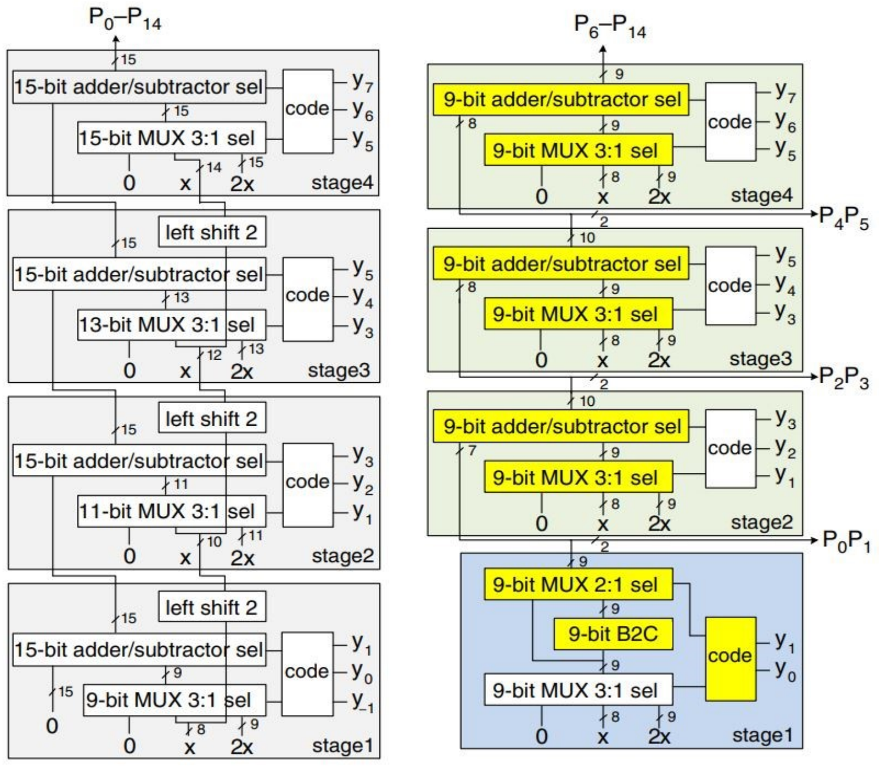
The discussion in the following chapter is substantially drawn from [41] [42] [43] where we first reported the development and evaluation of this technique.

3.1 Introduction

The main motivation for optimization came from the observations made while designing the conventional Booth multiplier. Conventional Booth multiplier architecture is impressive for reducing the number of partial products to half, but the implementation needs further optimization to achieve low-cost and high-performance modified signed multiplier with Booth encoding. The improvements presented in [42] for an 8 x 8 Booth multiplier dealt with the unnecessary usage of full width adder and multiplier hardware, optimization of B2C, removal of adder in the first stage, and replacing the two-input encoder with three input encoders in the first stage. More details about the work in [42] are discussed in the sub-section 1.1 of this document. Further optimized 8 x 8 Booth multiplier is presented in [41] by majorly focusing on two aspects, (1) improving the speed by the execution of square root carry select adders with carry look ahead block in parallel to reduce the number of stages of addition, and (2) optimization of Booth encoder along with the B2C with bubble pushing and deploying a simple hardware by the fusion of the encoder logic and the multiplexer logic at every stage. Complete discussion about the implementation and results are included in the section 3.3 of this document.

3.2 Low power-delay-product radix-4 8 x 8 Booth multiplier [42]

Booth encoding scheme applied at each stage of the conventional Booth multiplier architecture shown in Figure 13(a) uses one of the five distinct operations, to result in a partial product in the respective stage. The five distinct operations include: (1) all 0's, (2) direct use of multiplicand, X, (3) multiplicand left shifted by one bit position, 2X, (4) two's complement of X, and (5) right shift the two's complemented X by one bit position, -2X. Except the two's complement operation to generate the -X as partial product at the respective stage, all other operations involve trivial and fast parallel computation of shift.



(a) (b)

Figure 13. (a) Conventional Booth Multiplier [4], (b) optimized low PDP Booth Multiplier in [43] [42]

Design shown in Figure 13(a) avoid two's complement operation by deploying the adder-subtractor and switching the addition and subtraction operation relying on the MSB bit of the 3-bit group. But, it takes an additional hardware and power consumption due to the deployment of the number of XOR's at one of the multibit inputs of the adder at every stage. This leads to a significant power and area over head. Left shift operations are performed by hardwiring the input connections to the multi-bit 3-to-1 multiplexer to avoid the actual physical hardware. Moving forward to the next stage, which takes an adjacent group of 3-bits of multiplier to perform the encoding scheme, the possible partial product output is formed from one of the above mentioned five distinct values which are left shifted by two-bit positions because of the two-bit left shift to form 3-bit group. Hence, there is no need to generate the five possible values in any other stage but to be selected using the multiplexers at each stage while hardwiring the inputs to eliminate the inclusion of the left shift block. The optimized 8 x 8 Booth multiplier is proposed and presented in [43] [42], shown in Figure 13(b), minimized the encoder in the first stage from a 3-bit encoder to 2-bit encoder based on the fact that the LSB bit, y_{-1}, of the 3-bit group is always '0', the first stage adder-subtractor block is replaced with an optimized B2C, and the sizes of the 15-bit adder-subtractor blocks at each stage are reduced to 9-bit for eliminating the unnecessary computational cost at the LSB bits and sign bits at the MSB side. Optimized B2C presented in [42] is represented using equation (3.1) assuming the total number of bits of the word to be 2's complemented are odd where n is even and [a_n a_{n-1} … a_2 a_1 a_0] as input vector and [ac_n ac_{n-1} … a_2 a_1 a_0] as the two's complemented output

$$ac_0 = a_0$$

$$ac_1 = \overline{a_0} \oplus \overline{a_1}$$

$$ac_2 = \overline{a_2} \oplus (\overline{a_1} \cdot \overline{a_0}) = a_2 \oplus \overline{(\overline{a_1} \cdot \overline{a_0})}$$

$$ac_3 = \overline{a_3} \oplus (\overline{a_2} \cdot \overline{a_1} \cdot \overline{a_0}) = a_3 \oplus \overline{\left(a_2 + \overline{(\overline{a_1} \cdot \overline{a_0})}\right)}$$

$$\dots$$

$$ac_{n-1} = \overline{a_{n-1}} \oplus \overline{\left(a_{n-2} + \overline{\left(\overline{a_{n-3}} \cdot \overline{\left(a_{n-4} + \overline{\left(\overline{a_{n-5}} \cdot \cdots + \overline{(\overline{a_1} \cdot \overline{a_0})}\right)}\right)}\right)}\right)}$$

$$ac_n = a_n \oplus \overline{\left(\overline{a_{n-2}} \cdot \overline{\left(a_{n-3} + \overline{\left(\overline{a_{n-4}} \cdot \overline{\left(\overline{a_{n-5}} + \cdots + \overline{(\overline{a_1} \cdot \overline{a_0})}\right)}\right)}\right)}\right)}$$

(3.1)

3.3 Low-cost and high-performance radix-4 8 x 8 Booth multiplier [41]

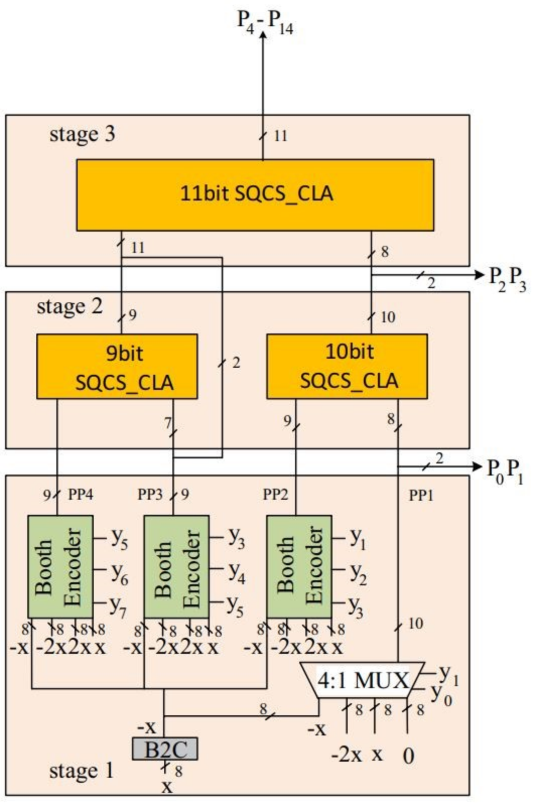


Figure 14. Further optimized low-cost and high-performance 8 x 8 Booth Multiplier with parallel additions in [41]

Booth multiplier architecture is further optimized in [41], shown in Figure 14, by implementing the following strategies: (1) replacing the adder-subtractor blocks in all stages with only adders and using the B2C output to feed the -X and -2X inputs to generate the partial products in all stages where necessary to reduce the power consumption and design area, (2) sequential addition of partial products is replaced with parallel addition, number of stages of addition are reduced to two, to improve the speed while keeping the area of partial product reduction hardware almost same, (3) encoded partial products are generated directly using the speed optimized encoder block with inherent multiplexer logic, (4) B2C block in the worst delay path is further optimized to reduce the power consumption and design area, and (5) Square root carry select adders with carry look ahead blocks are used to improve the speed performance with a power and area tradeoff.

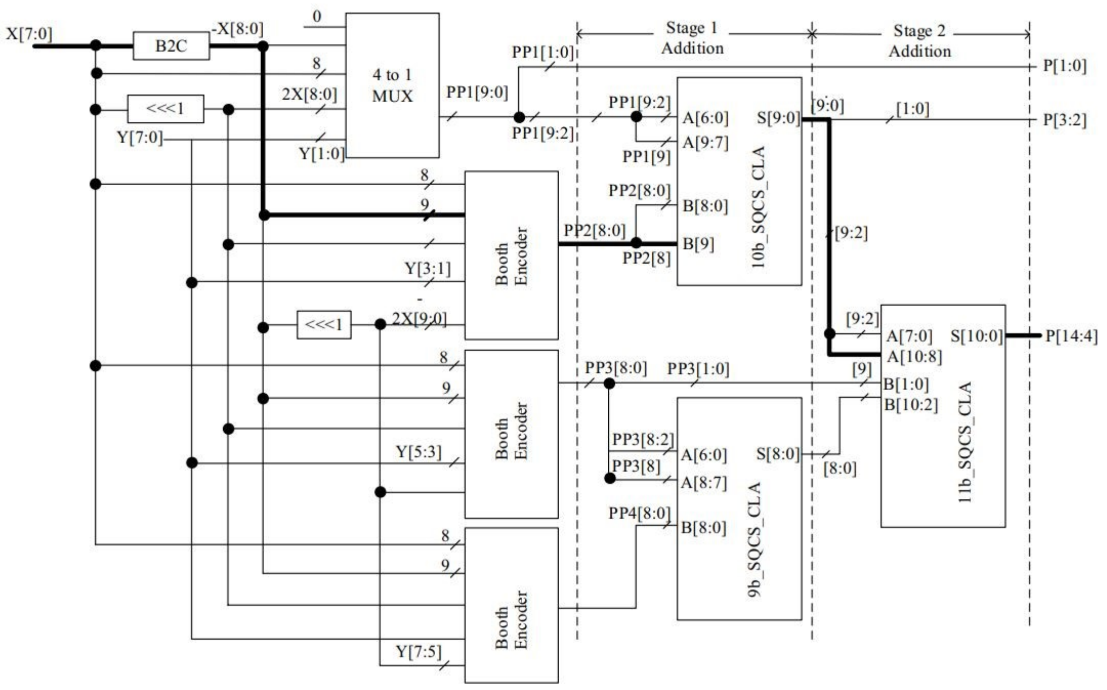


Figure 15 Radix-4 8x8 signed Booth multiplier with parallel encoding and additions in [41]

The anatomy of the radix-4 8 x 8 signed Booth multiplier with parallel adders is presented in [41] is illustrated in Figure 15. This parallel addition with careful grouping has higher performance advantage compared to employing the series addition of the partial products in [42] [43] [44] [45] [46]. The worst delay path includes B2C, Booth encoder, 10-bit SQCS with CLA, and 11-bit SQCS with CLA. Hence, all the blocks in worst delay path are optimized for speed. The explanation about each sub-optimization strategy used is included in the following sub-sections. Layout of the modified 8x8 Booth multiplier with parallel encoding scheme and parallel partial product reduction, which is synthesized at 500 MHz clock frequency and performed PnR, using Synopsys Design Compiler (DC) and IC compiler (ICC) respectively, is shown in Figure 16. Total chip area of the layout shown in the following figure is 2058.566 $(\mu\mathrm{m})^2$ with the standard cell utilization factor of 57.32%. Hence, the total cell area is nearly 1180 $(\mu\mathrm{m})^2$ with total number of standard cells used are 358.

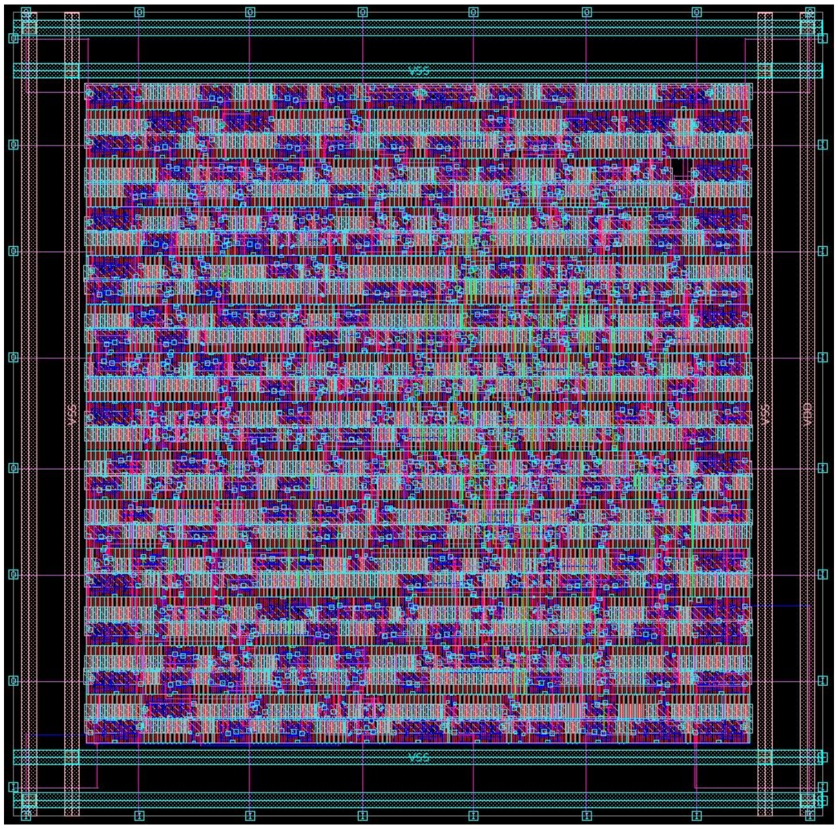


Figure 16 Layout of the modified Radix-4 8x8 signed Booth multiplier with parallel encoding and additions [41]

3.3.1 Booth encoder optimization & partial product generation

Pre-computation stage first encoder for the first partial product, PP1, generation is reduced to a two-input encoder, 4 to 1 multiplexer, since the LSB bit, y_{-1}, is always '0' and can be ignored. The rest of the encoding uses y_1 and y_0 bits of the multiplicand to generate the partial products based on the encoding scheme shown in Table 3.1. The operation of choosing one of the four possible partial products is performed using a 9-bit wide 4 to 1 multiplexer, shown in Figure 17. The multiplexer takes four inputs, 0, X, -X, and -2X, with two LSB bits of the multiplier as selection inputs. The negative value of the multiplicand, -X, is generated by using the B2C and by feeding -2X, output from B2C, as an input after arithmetic left shift is performed just by hardwiring while avoiding the actual physical hardware. Discussion about the optimized B2C is included in a later sub-section.

Table 3.1 First stage encoding scheme [41]

y_1	y_0	y_{-1}	Partial Product
0	0	0	0
0	1	0	$+X$
1	0	0	$-2X$
1	1	0	$-X$

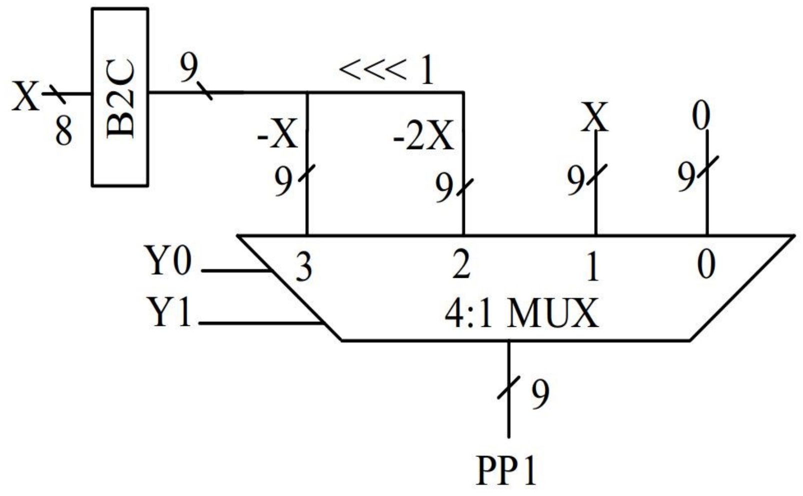


Figure 17 First stage PP generation in [41]

The three encoders needed for the generation of the next three partial products, PP2, PP3, and PP4, in the pre-computation stage run in parallel, shown in Figure 15, and are built on a same logic. The optimized encoder logic presented in [41], achieved by applying bubble pushing and Boolean logic minimization techniques, is shown in Figure 18 (b), and is compared to the encoder presented in [43] [42]. Typical Booth encoder takes in three input bits and yields three selection output bits, S_2, S_1, and S_0, at each stage to fed to the multiplexer to choose one of the eight possible values based on the booth encoder strategy. Where in the optimized encoder results in four output bits, P, Q, R, and S, to choose between {X, 2X, -X, -2X} without the need for an exclusive multiplexer since the simplified multiplexer functionality is fused with the encoder to generate the partial products with reduced delay and reduced utilization of the design space. The Boolean logic used to generate the P, Q, R, and S scalars is shown in the Figure 18 (b) and the Boolean equation representation of the scalars as follows:

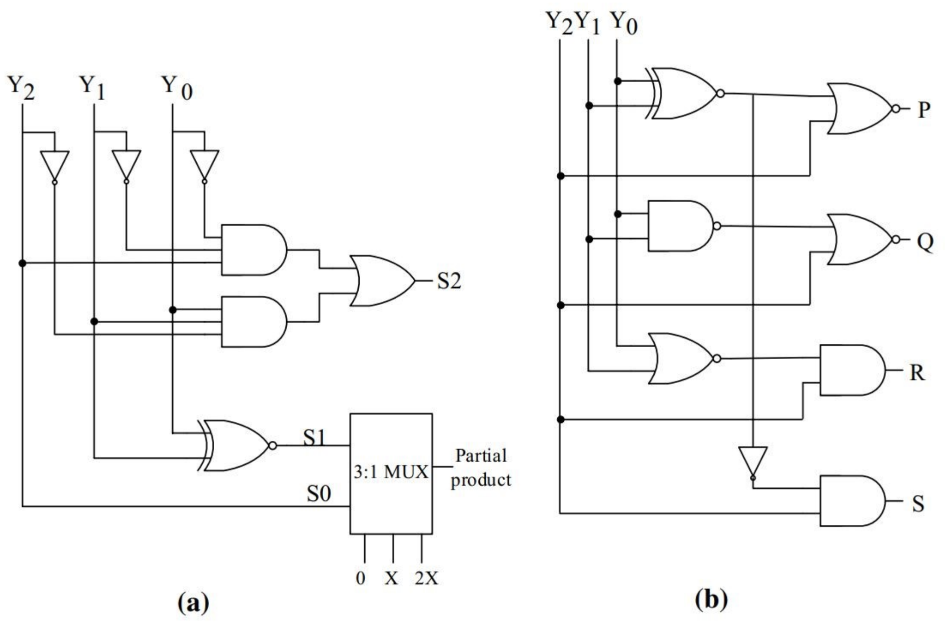


Figure 18 (a) Booth encoder in [43] [42], (b) optimized Booth encoder in [41]

$$P = \overline{(y_0 \ominus y_1) + y_2}$$

(3.2)

$$Q = \overline{\overline{(y_0 \cdot y_1)} + y_2}$$

(3.3)

$$R = \overline{(y_0 + y_1)} \cdot y_2$$

(3.4)

$$S = (y_0 \oplus y_1) \cdot y_2$$

(3.5)

The fused encoder and multiplexer logic presented in [41] for the generation of the 2nd, 3rd, and 4th partial products is shown in Figure 19 (a) and is compared with the conventional Booth encoder and the multiplexer, shown in Figure 19 (b), used to generate the partial products at each stage. P, Q, R, and S scalar outputs from the optimized encoder are used to select the X, 2X, -2X, and -X vectors respectively to generate the partial product, PP, and the Boolean representation of the selection hardware is represented using the following equation (3.6):

$$PP = (P \cdot X) + (Q \cdot 2X) + \big(R \cdot (-2X)\big) + \big(S \cdot (-X)\big)$$

$$= \overline{\overline{(P \cdot X)} \cdot \overline{(Q \cdot 2X)} \cdot \overline{\big(R \cdot (-2X)\big)} \cdot \overline{\big(S \cdot (-X)\big)}}$$

(3.6)

Selection inputs P, Q, R, and S are enabled with the 3-bit input vectors {001, 010}, {011}, {100}, and {101, 110} respectively. In case of the input vectors ‘000’ and ‘111’, all the output bits of the encoder are 0’s and hence the fifth possible partial product of all 0’s is assigned to the PP output vector.

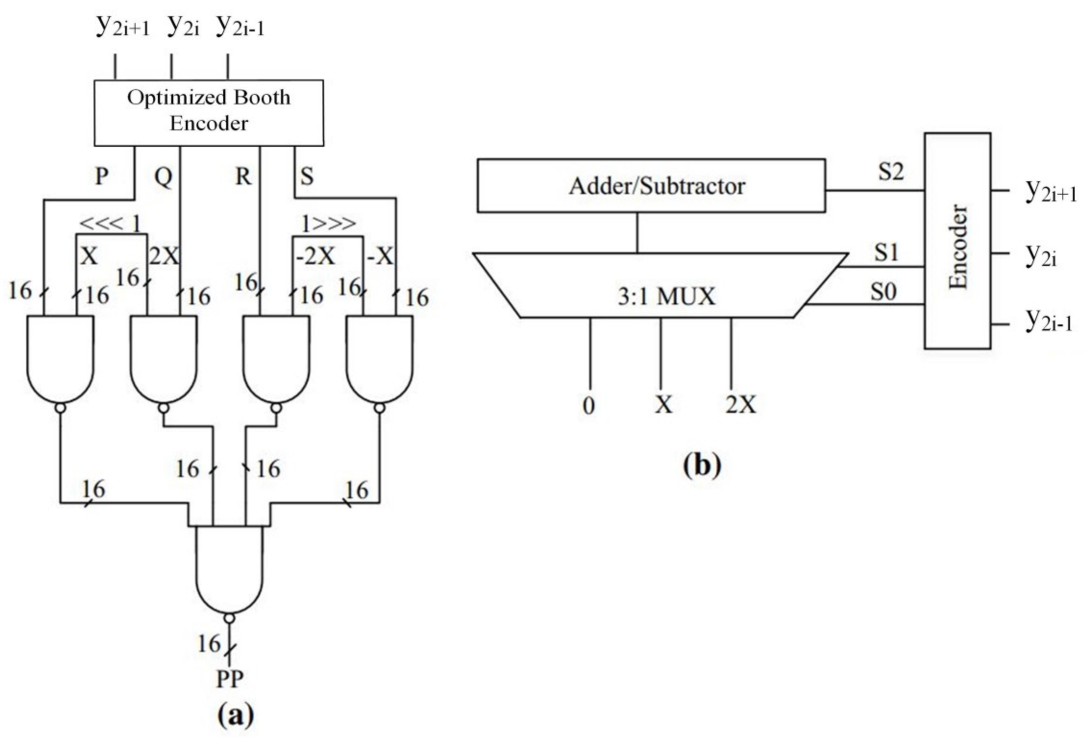
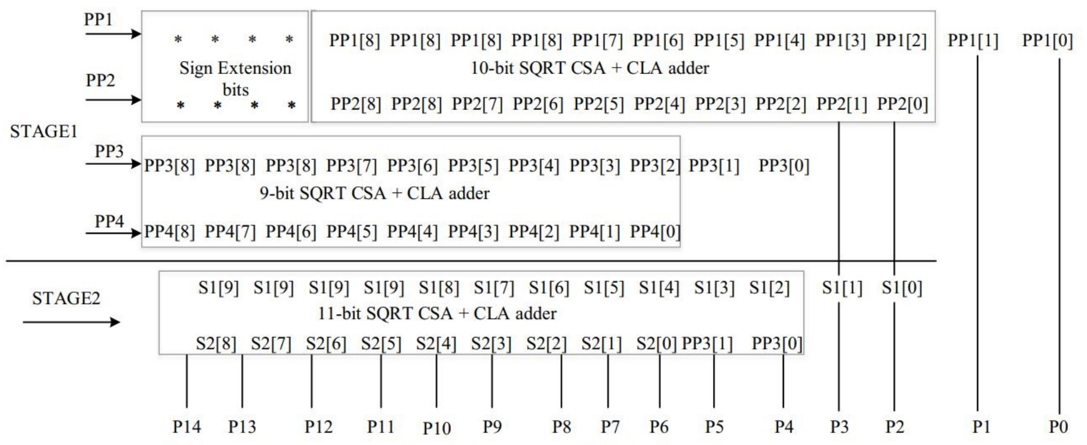


Figure 19 (a) 2nd, 3rd, and 4th PP generation in [41], (b) PP generation in [43] [42]

3.3.2 Reduction of Partial Products using Two-Stage Parallel Addition

As mentioned above, the three partial products generated are summed to have the final product. Square root (SQRT) carry select adder with carry look ahead (CSA-CLA) blocks are used to increase the computation speed with the power and area trade-off. Carry select blocks works on a simple and very effective way to increase the speed by precomputing the possible sum outputs with the assumption of ‘0’ carry-in and ‘1’ carry-in. By the time of actual carry-in signal arrival at a particular block, two possible sum values are ready to be selected in parallel by the multiplexer. Including CLA logic in sub-block further improves the speed performance by the very quick carry out generation based on the multi-bit propagate and generate chain. The parallel addition structure is carefully designed following the grouping of the partial products illustrated in Figure 20. All the partial products, PP1, PP2, PP3, and PP4, generated are of 9-bit wide, PP[9:0], hence a 9-bit B2C is enough instead of a full width, 15-bit, B2C. PP1[8], PP2[8], and PP3[8] are repeated as extension bits for keeping the sign of the partial product. PP1[0] and PP1[1] are the final product bits P[0] and P[1] respectively. 10-bit SQRT CS-CLA with 3-bit CLA, 3-bit CS-CLA, and 4-bit CS-CLA sub-blocks, shown in Figure 21 (b), is used to sum the partial products PP1[11:2] and PP2[9:0], where PP1[11:9] are the sign extension bits formed by repeating the sign indicator bit, PP1[8]. A 9-bit SQRT CS-CLA with 2-bit CLA, 3-bit CS-CLA, and 4-bit CS-CLA sub-blocks, shown in Figure 21 (a), is used to sum the partial products PP3[10:2] and PP4[8:0], where PP3[10:9] are the sign extension bits formed by repeating the sign bit, PP3[8]. At each partial product row to be summed in stage 1, the $(n+1)^{th}$ bit indicates the sign of that partial products and that bit value is copied across the proceeding MSB bits until reaching the bit index of 14, with the indexing begin at 0, with reference to the indices of the final product bits. Like the 10-bit adder, 9-bit adder takes the first partial product with a starting index of 2. The remaining LSB bits of the first input vector PP3, PP3[0] and PP[1], are carried down to the second stage addition to fill in the gap to form a rectangular group structure for forming a SQRT CS-CLA. These two adders, 9-bit adder and 10-bit adder, run in parallel for the stage 1 reduction resulting in two intermittent output vectors, S1 and S2, to be further reduced, summed, in the final stage, stage 2.


PP1
PP2
STAGE1
PP3
PP4
STAGE2
* * * *
Sign Extension bits
* * * *
PP1[8] PP1[8] PP1[8] PP1[8] PP1[7] PP1[6] PP1[5] PP1[4] PP1[3] PP1[2] PP1[1] PP1[0]
10-bit SQRT CSA + CLA adder
PP2[8] PP2[8] PP2[7] PP2[6] PP2[5] PP2[4] PP2[3] PP2[2] PP2[1] PP2[0]
PP3[8] PP3[8] PP3[8] PP3[7] PP3[6] PP3[5] PP3[4] PP3[3] PP3[2] PP3[1] PP3[0]
9-bit SQRT CSA + CLA adder
PP4[8] PP4[7] PP4[6] PP4[5] PP4[4] PP4[3] PP4[2] PP4[1] PP4[0]
S1[9] S1[9] S1[9] S1[9] S1[8] S1[7] S1[6] S1[5] S1[4] S1[3] S1[2] S1[1] S1[0]
11-bit SQRT CSA + CLA adder
S2[8] S2[7] S2[6] S2[5] S2[4] S2[3] S2[2] S2[1] S2[0] PP3[1] PP3[0]
P14 P13 P12 P11 P10 P9 P8 P7 P6 P5 P4 P3 P2 P1 P0

Note: * are the sign extension bits which is same as 9th bit for all partial product rows
S1[9:0] are sum bits generated by 10-bit SQRT CSA adder in first stage
S2[8:0] are sum bits generated by 9-bit SQRT CSA adder in first stage

Figure 20 Optimized radix-4 signed 8 x 8 Booth multiplier with two-stage architecture in [41]

The two sum outputs, S1[12:2] and S2[8:0], along with the two partial product bits, PP3[1:0], left unprocessed, from stage 1 are fed to the 11-bit SQRT CS-CLA in the final stage, stage 2. As shown in Figure 21 (c), 11-bit SQRT CS-CLA constitutes a 3-bit CLA block, 4-bit CS-CLA block, and 4-bit CS-CLA block. This parallel addition with two stage structure is reducing the two adder delays compared with the designs presented in [44] [45] [46], and reducing one adder delay if compared to the design in [42].

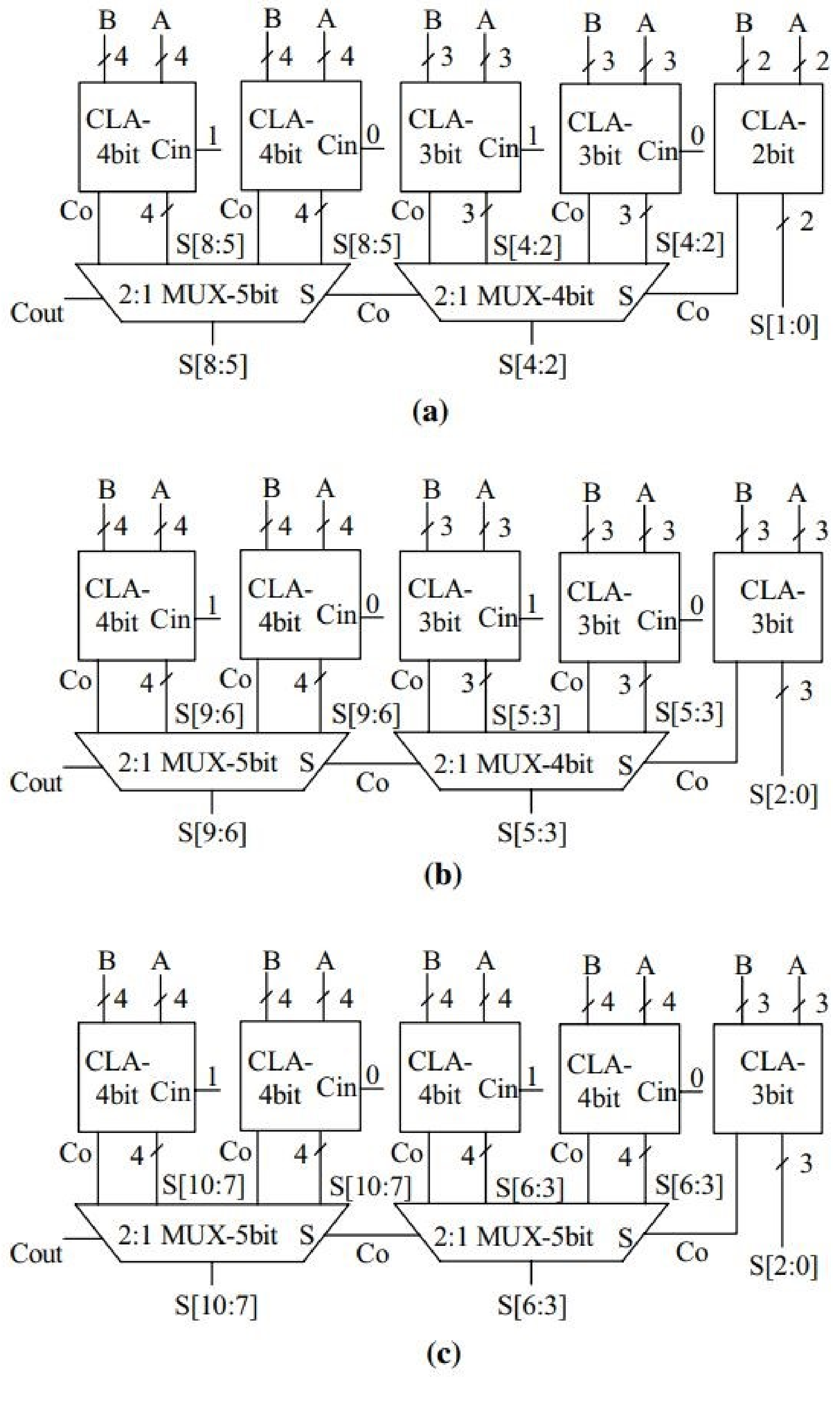


Figure 21 (a) 9-bit SQRT CS-CLA, (b) 10-bit SQRT CS-CLA, (c) 11-bit SQRT CS-CLA [41]

3.3.3 B2C optimization

Binary two's complement block is one of the important blocks needed to be optimized for the reduction in delay since it has the ripple structure. An optimized B2C is presented in [42], discussion is included in the sub-section 1.1, shown in Figure 22 (a), and the design is better optimized in [41], shown in Figure 22 (b), for the reduction of area and power consumption. Based on the widely accepted and observed fact that the inverters at the inputs consume more power due to high switching activity, reduction of area and power consumption is achieved by bubble pushing the inverters at the inputs used in [42]. The Boolean equations for the 9-bit B2C used in [42] and [41], shown in Figure 22, are as follows, where [a_n a_{n-1} … a_2 a_1 a_0] as input vector and [ac_n ac_{n-1} … a_2 a_1 a_0] as the two's complemented output:

$$ac_0 = a_0$$

$$ac_1 = a_0 \oplus a_1$$

$$ac_2 = \overline{a_0 + a_1} \ominus a_2$$

$$ac_3 = \overline{a_0 + a_1 + a_2} \ominus a_3$$

$$ac_4 = \overline{a_0 + a_1 + a_2 + a_3} \ominus a_4$$

$$ac_5 = \overline{((a_0 + a_1 + a_2 + a_3) + a_4)} \ominus a_5$$

$$= \overline{\overline{\overline{a_0 + a_1 + a_2 + a_3}} + a_4} \ominus a_5$$

$$ac_6 = \overline{((a_0 + a_1 + a_2 + a_3) + a_4 + a_5)} \ominus a_6$$

$$= \overline{\overline{\overline{a_0 + a_1 + a_2 + a_3}} + a_4 + a_5} \ominus a_6$$

$$ac_7 = \overline{((a_0 + a_1 + a_2 + a_3) + a_4 + a_5 + a_6)} \ominus a_7$$

$$= \overline{\overline{\overline{a_0 + a_1 + a_2 + a_3}} + a_4 + a_5 + a_6} \ominus a_7$$

$$ac_8 = \overline{((a_0 + a_1 + a_2 + a_3) + a_4 + a_5 + a_6 + a_7) \oplus a_8}$$

$$= (a_0 + a_1 + a_2 + a_3 + a_4 + a_5 + a_6 + a_7) \ominus a_8$$

$$= \overline{\overline{\overline{\overline{\overline{a_0 + a_1 + a_2 + a_3}} + a_4 + a_5 + a_6}} + a_7} \ominus a_8$$

(3.7)

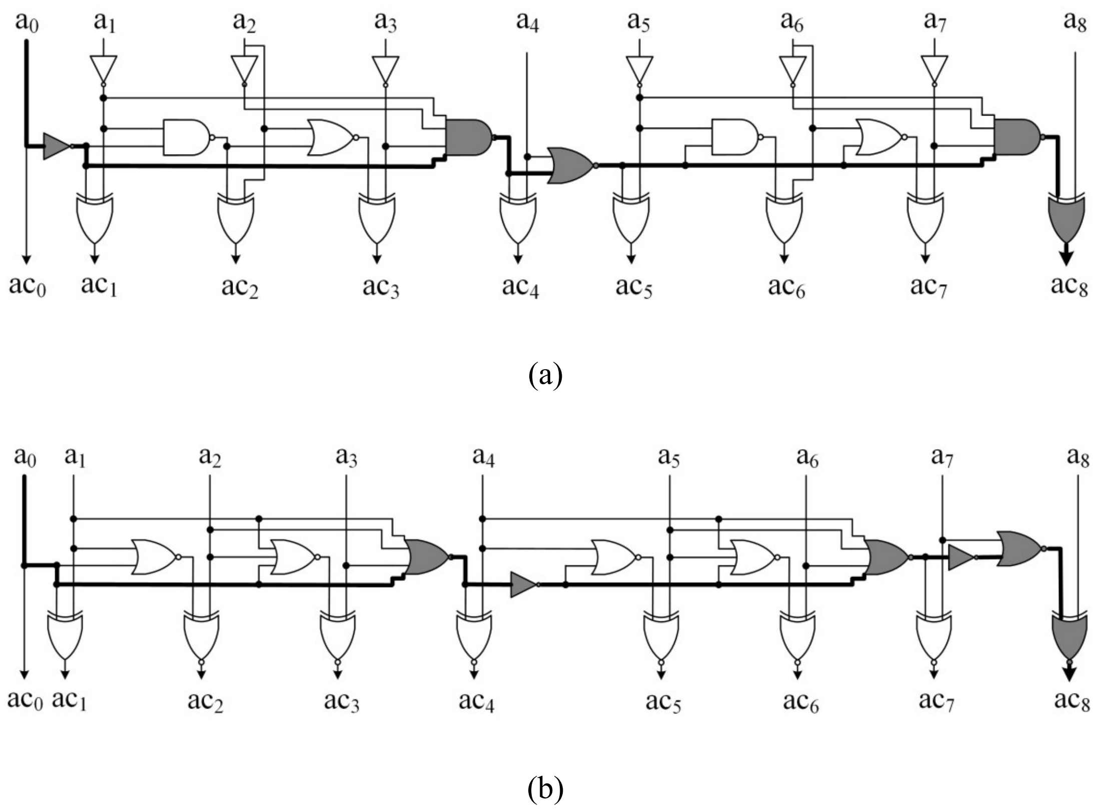


Figure 22. (a) optimized B2C using bubble pushing in [42], (b) better optimized B2C in [41]

The synthesis results for the two B2C designs are compared in the following Table 3.2. Synthesis is performed on the two designs designed in Synopsys 32nm CMOS RVT standard cells at 500 MHz frequency and 1.05 supply voltage with the help of

Synopsys design compiler (DC). Reduction in design area and power consumption can be observed with a very small delay trade-off.

Table 3.2 Comparison of the synthesis (pre-layout) results for the B2C designs [41]

B2C Design	Frequency (MHz)	Area $(\mu\text{m})^2$	Power (μW)	Delay (ns)
Boppana et al. [41]	500	56.41	15.15	0.41
B2C in [42]	500	63.02	16.85	0.4

4 Proposed Multiplier Architecture based on radix-8 (3-bit grouping) structure

Modified booth multiplier's encoding scheme with the deployment of the parallel adders of square root carry select (SQCS) structure, constitutes carry look ahead (CLA) sub-blocks, is presented in [41], and has achieved a significant increase in speed by reducing the worst path delay and a significant decrease in the power consumption with a small reduction in area. As shown in Table 4.1, Booth encoding reduces the number of partial products by half, and the parallel execution of the adders in a binary tree style requires the same number of adders required by the series execution of the adders in summing the partial products while the number of stages required to sum the partial products to compute the final product are reduced to $\log_2 N$ from $\frac{N}{2} - 1$.

Table 4.1 Booth multiplier with binary tree style reduction of partial products using adders

Multiplier size (N)	No. of PPs (N/2)	No. of additions (N/2-1)	Complexity in Hardware (#Adders)	Complexity in Time (PP reduction time)
8	4	2+1 = 3	$O(N) = \frac{N}{2} - 1$	$O(\log N) = \log_2 N$
16	8	4+2+1=7	$O(N) = \frac{N}{2} - 1$	$O(\log N) = \log_2 N$
32	16	8+4+2+1=15	$O(N) = \frac{N}{2} - 1$	$O(\log N) = \log_2 N$
64	32	16+8+4+2+1=31	$O(N) = \frac{N}{2} - 1$	$O(\log N) = \log_2 N$

Though the state-of-the-art modified Booth multiplier architectures use the radix-8 Booth encoding with 3-bit grouping of the multiplier bits, the effective number of multiplier bits, from LSB to MSB, used in partial product generation at any stage are two.

Hence, the Booth multiplier architecture can be called as a radix-4 structure instead of radix-8 structure. The proposed architecture in this work is an attempt to prove that the use of the higher radix, radix-8 in this work, by deploying a non-trivial block, which further reduces the number of partial products to be added compared to the Booth architectures, and area-power-delay aware grouping of the partial products results in achieving a significant advantage, especially for the larger word size multipliers such as 32-bit, 64-bit, etc. The proposed structure first converts the multiplicand and multiplier to its magnitudes, pre-compute the non-trivial functions, NT1, NT2, and NT3, included in Table 4.2 which depicts the separation of the eight functions to four trivial and four non-trivial functions.

Table 4.2 Proposed radix-8 (3-bit) grouping to separate trivial and non-trivial computations

	Trivial	
0 (000)	By-pass	N0 (No-Op)
1 (001)	<< 0 (No shift or addition)	T0
2 (010)	<< 1 (Left shift by 1)	T1
4 (100)	<< 2 (Left shift by 2)	T2
	Non-Trivial	
3 (011)	2x+1x (or) 4x-1x	NT1
5 (101)	4x+1x	NT2
6 (110)	4x+2x	S1 = (NT1) << 1
7 (111)	8x-x (or) 4x+2x+1x	NT3

4.1 8 x 8 signed multiplication using the proposed design :

As seen in Table 4.2, All the trivial functions involve one of the four operations: no operation, all 0's; the magnitude of the multiplicand (X) with no shift, X; X left shifted by 1 bit position, 2X; X left shifted by 2-bit positions, 4X. The complete working of the proposed radix-8 with the grouping of 3-bit multiplier bits for the 8x8 signed multiplier is clearly illustrated using the example-1 and example-2 shown in Figure 23 and Figure 24 respectively. As a first step, the magnitudes of the multiplicand, A, and multiplier, B, are computed as A_p and B_p. Magnitude of the negative number is computed using the 2's complement and the magnitude of the positive number is the number itself. In example 1, the negative multiplicand, A = -72 (1011 1000), is converted to its 2's complement, Ap (0100 1000), which is denoted as X for the ease of representation in later steps. Taking the X as an input, the three non-trivial computations, 3X, 5X, and 7X, are computed by performing the addition operations. 3X is formed by summing the X and 2X, 5X is formed by summing the X and 4X, and 7X is formed using the summation of -X and 8X instead of summing X, 2X, and 4X to complete reduce the number of additions.

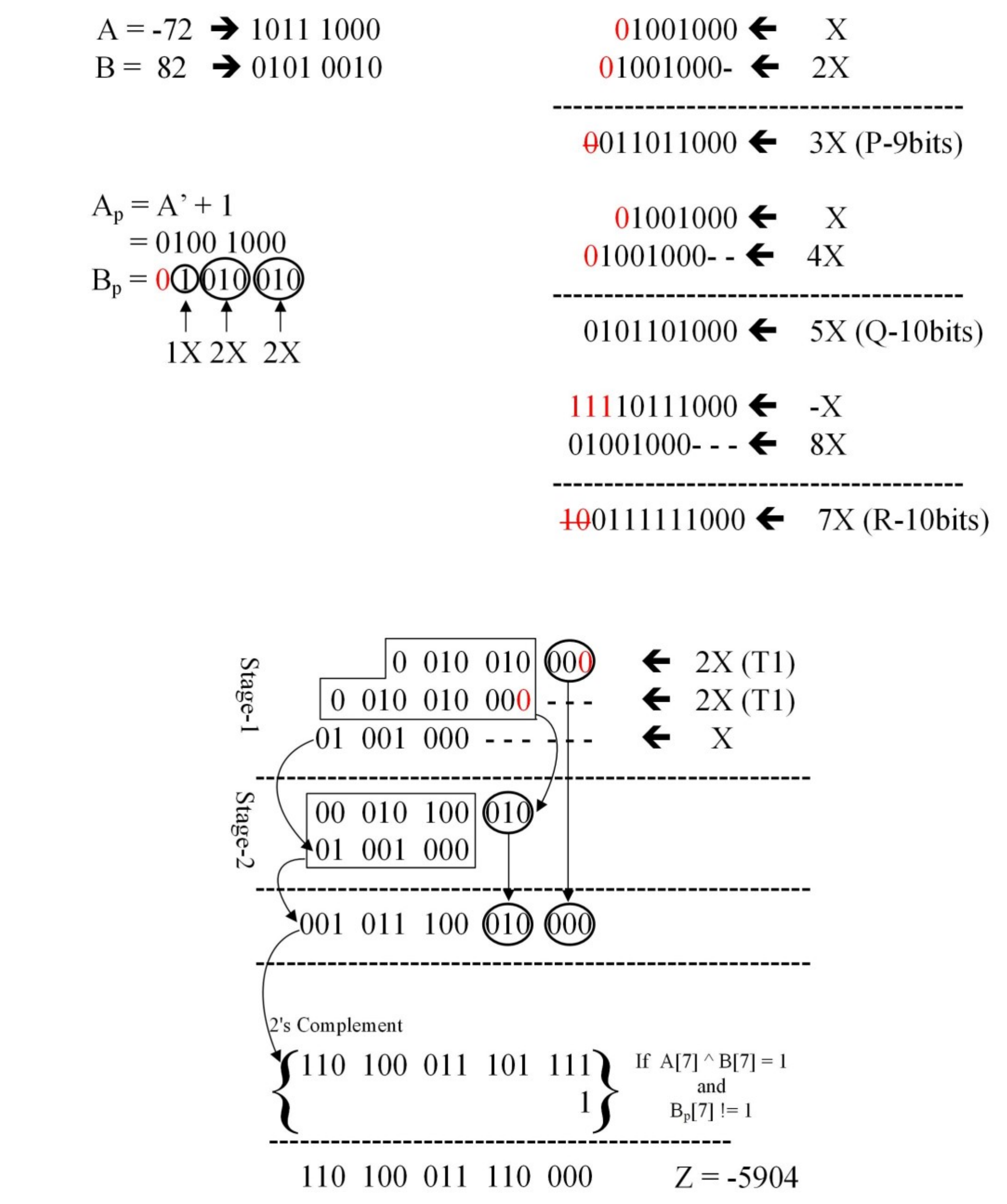


Figure 23 Example-1 calculation of the 8 x 8 signed multiplication using the proposed radix-8 architecture (NOTE: ! - NOT symbol. ^ - XOR symbol)

The first 6-bits of B_p on the LSB side are grouped into two 3-bit groups and the first two partial products in stage 1 are calculated following the computation criteria provided in Table 4.2. Indexing always starts from the LSB side with 0 as start index.

The second partial product needs to be shifted by three bits since the second group multiplier bits start from index 3. The bit, ‘1’, at position 7 of index 6 from the LSB side forms the third partial product of X left shifted by 6-bit positions. If this lone bit is ‘0’, then the third partial product contains all 0’s. The first two partial products enclosed in the L-shaped box are added together and the third partial product is carried to the next stage to perform the addition in stage 2. In two stages the magnitude of the product is computed as 001011100010000. As a final post computation step, the sign of the result should be decided based on the combination of signs of the original inputs, A and B. If the signs of A and B are opposite, $A[7] \oplus B[7] == 1$, and the magnitude of multiplier is not the max value of 128, $\sim B_p[7] == 1$, then the final magnitude obtained is 2’s complement. XOR operation is represented as ^ and NOT operation is represented as ! in all the figures with example illustrations. The sign bit of a multiplier or multiplicand stays same even after performing the 2’s complement in only one case where the actual value is the lowest, -128, i.e., the magnitude value is at its maximum for the 8-bit signed number.

Like the example 1, example 2 also starts with the first of magnitude calculation of both multiplicand, A = 6 (0000 0110), and multiplier, B = -36 (1101 1100), 2’s complement of B, Bp=(00100100) shown in Figure 24. As a next step, the non-trivial computation required are performed. In the proceeding step, the grouping of bits of the multiplier magnitude, B_p, is completed and the three partial products are generated based on the two 3-bit groups and the lone bit at index 6. Required additions are computed in stage 1 and stage 2. The result after performing the summation is the magnitude of the product. The sign of the product is determined based on the signs of the actual inputs. If the signs of the actual inputs are opposite, then the result obtained from the summation at stage 2 is 2’s complemented to find the actual value of the product.

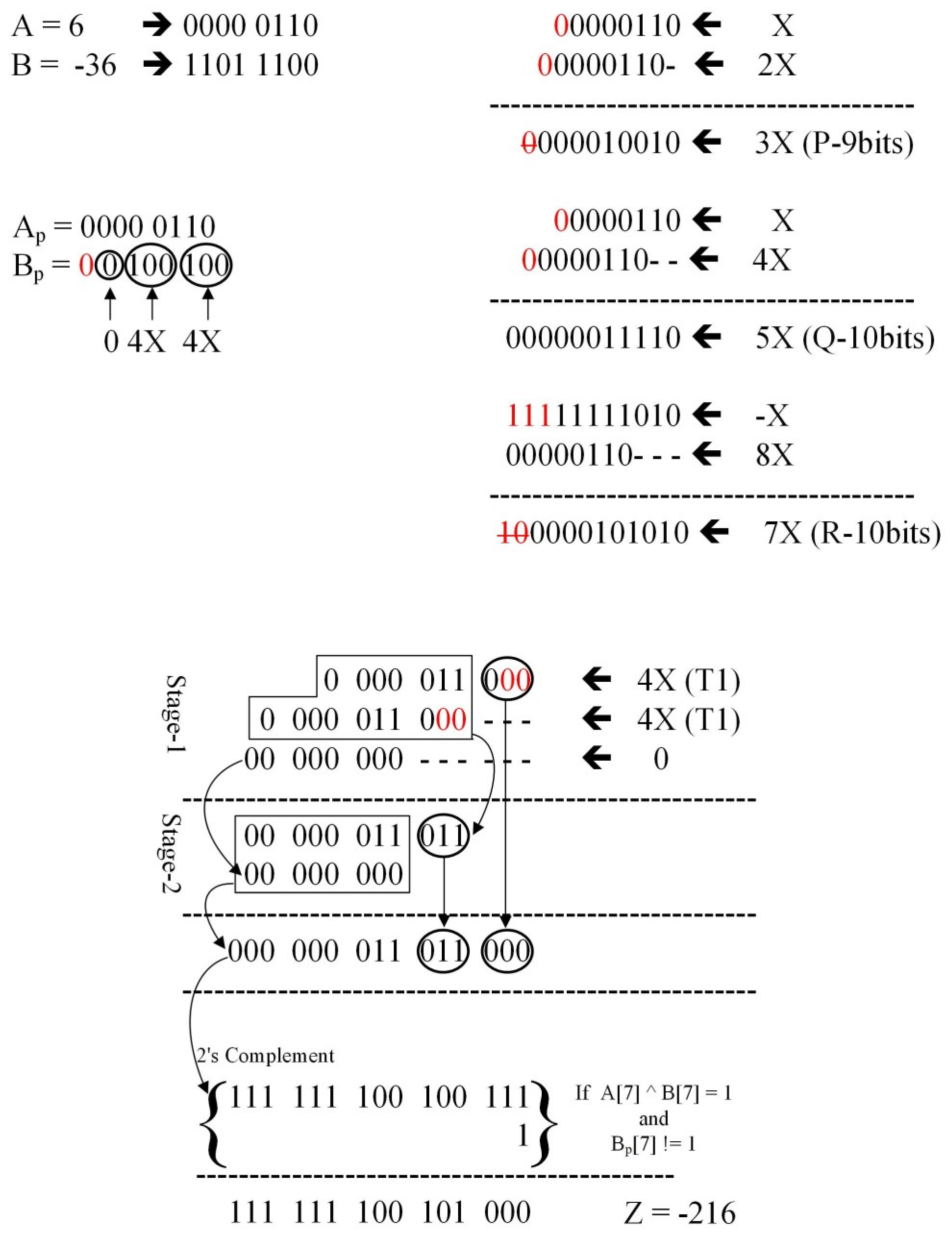
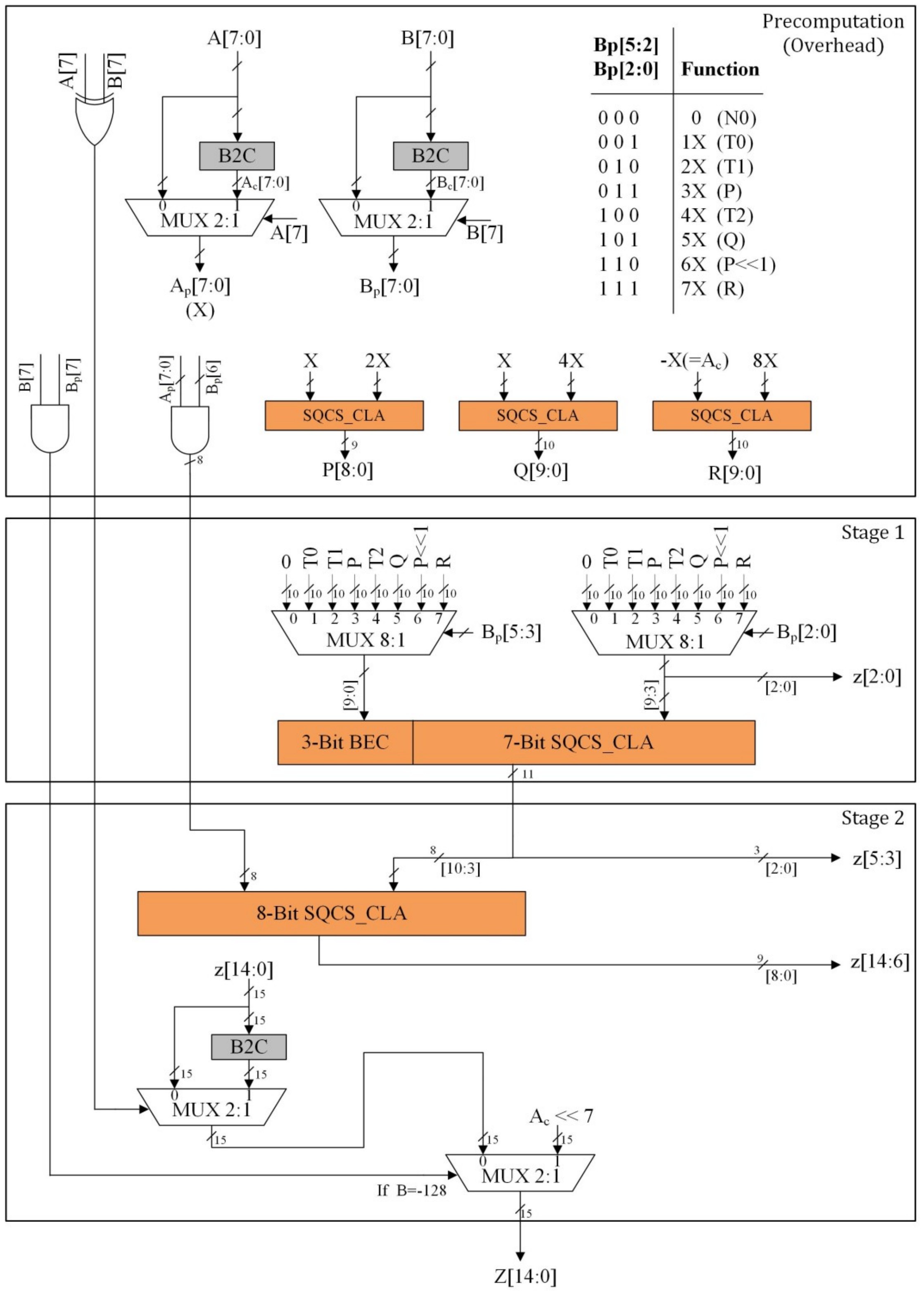


Figure 24 Example-2 calculation of the 8 x 8 signed multiplication using the proposed radix-8 architecture

And, the architecture of the proposed 8 x 8 design implemented using Synopsys RVT standard cells is shown in the following Figure 25. The precomputation block constitutes the two major parts which work sequentially. First one for finding out the magnitude using the two 8-bit B2Cs and two 8-bit wide 2 to 1 multiplexer. The second major part is the group of three non-trivial computation blocks built on SQCS-CLA. As discussed before, the non-trivial computation blocks results in vectors P, Q, and R to which hold the values of 3X, 5X, and 7X respectively. Since the magnitude of the multiplicand computed for an 8-bit signed number does not exceed 7-bit value except for the only one case where the magnitude is of 8-bit is when the actual multiplicand, A, is at its lowest value of -128 (1000 0000) which has highest magnitude of +128 (1000 0000) after 2's complement conversion. Hence, the A_p and B_p values are of 8-bit values and the P, Q, and R vectors does not exceed 9-bit value of 384 (1 1000 0000), 10-bit value of 640 (10 1000 0000), and 10-bit value of 896 (11 1000 0000) respectively. The precomputation block is kind of an overhead block looks like it is taking more design area and power since it has two B2C's working in parallel and three SQCS-CLA addition blocks running in parallel. Since the adders in this overhead stage are fed with the output from the B2C+MUX block, the delay of the overall block constitutes delay from B2C, Multiplexer, and SQCS-CLA.

Next, in stage 1, the two partial products are generated by selecting one of the 8-possible vectors, 0, T0, T1, P, T2, Q, P<<1, and R with the selection inputs $B_p[2:0]$ and $B_p[5:3]$. Here, 0 is all 0's, possible trivial values T0, T1, and T2 which are X, 2X, and 4X values respectively can be fed to the multiplexer without the need for any shifter but just



Bp[5:2] Bp[2:0]	Function
0 0 0	0 (N0)
0 0 1	1X (T0)
0 1 0	2X (T1)
0 1 1	3X (P)
1 0 0	4X (T2)
1 0 1	5X (Q)
1 1 0	6X (P<<1)
1 1 1	7X (R)

Figure 25 Proposed 8 x 8 signed multiplier with radix-8 architecture

by hardwiring the connections, and the non-trivial values, P, Q, and R, are from the non-trivial computational blocks from the precomputation block. As shown in example 1 and example 2 from Figure 23 and Figure 24 respectively, the three LSB bits of the first partial product generated from the multiplexer on the right side are assigned the first three LSB output bits, z[2:0] of the magnitude of the product. Here, the intermediate vector, z, represent the magnitude of the final product. The next six bits of the first partial product and the second partial product of ten bits are added together in stage 1. The careful grouping in the current stage and later stage results in reduction in addition hardware by employing the binary excess-1 code, BEC, style generator. Hardware of the BEC style block is less complex, simple, faster, requires less design area, and consumes less power for performing its operation. The BEC block does generate the excess-1 code if the carry output from the 7-bit SQCS-CLA block which is fed to the BEC is '1'. The 7-bit adder with 3-bit BEC structure results in a 11-bit output as an output of stage 1 and the first three bits of the output are the product magnitude bits z[5:3].

In stage 2, the result obtained from the stage 1, z[10:3], and the 8-bit input, either all 0's or X, formed based on the 7th bit of the magnitude of the multiplier at index six are added using an 8-bit SQCS-CLA. The 9-bit result of the adder is assigned to z[14:6]. As a final part of the process, the sign of the final product is determined by the sign bits, A[7] and B[7] of the actual inputs, A and B. If both the sign bits are opposite, then the 2's complemented value of the z is selected as the output. This selection is done using a 15-bit wide 2 to 1 multiplexer. And, the final minor step of stage 2 is to deal with the one extreme case where the multiplier is -128 (1000 0000) with B[7] and B_p[7] equals to 1. As mentioned above, this is the only case where the MSB bit, the sign bit, of the actual multiplier, B, and the MSB bit of the magnitude of the multiplier, $B_p[7]$ are equal to '1'. If this extreme case occurs, then the final output is simply equals to the 2's complement of the multiplicand, Ac, left shifted by 7-bit positions. Again, there is no need for any shifter since the shifted input is just hardwired to the one of the inputs of the multiplexer. If not the extreme case, the output computed using stage 1 and stage 2 using adders with sign conversion, if needed, is assigned to the final output, Z. Worst delay path consists of three sub-blocks from the precomputation stage: B2C, 2 to 1 multiplexer, and SQCS-CLA, two sub-blocks from stage 1: 8 to 1 multiplexer and the LSB side hardware for addition of the 7-bit SQCS-CLA, and four sub-blocks from stage 2: 8-bit SQCS-CLA, B2C, and two 2 to 1 multiplexers in series.

Layout of the proposed 8 x 8 signed multiplier with a simple and new radix-8 structure for partial product generation and new grouping strategy for the partial product reduction, which is synthesized at 250 MHz clock frequency and performed PnR, is shown in Figure 26. Total chip area of the layout shown in the following figure is 2531.274 $(\mu m)^2$ with the standard cell utilization factor of 60.92%. Hence, the total cell area is nearly 1542 $(\mu m)^2$ with the total number of standard cells used as 497.

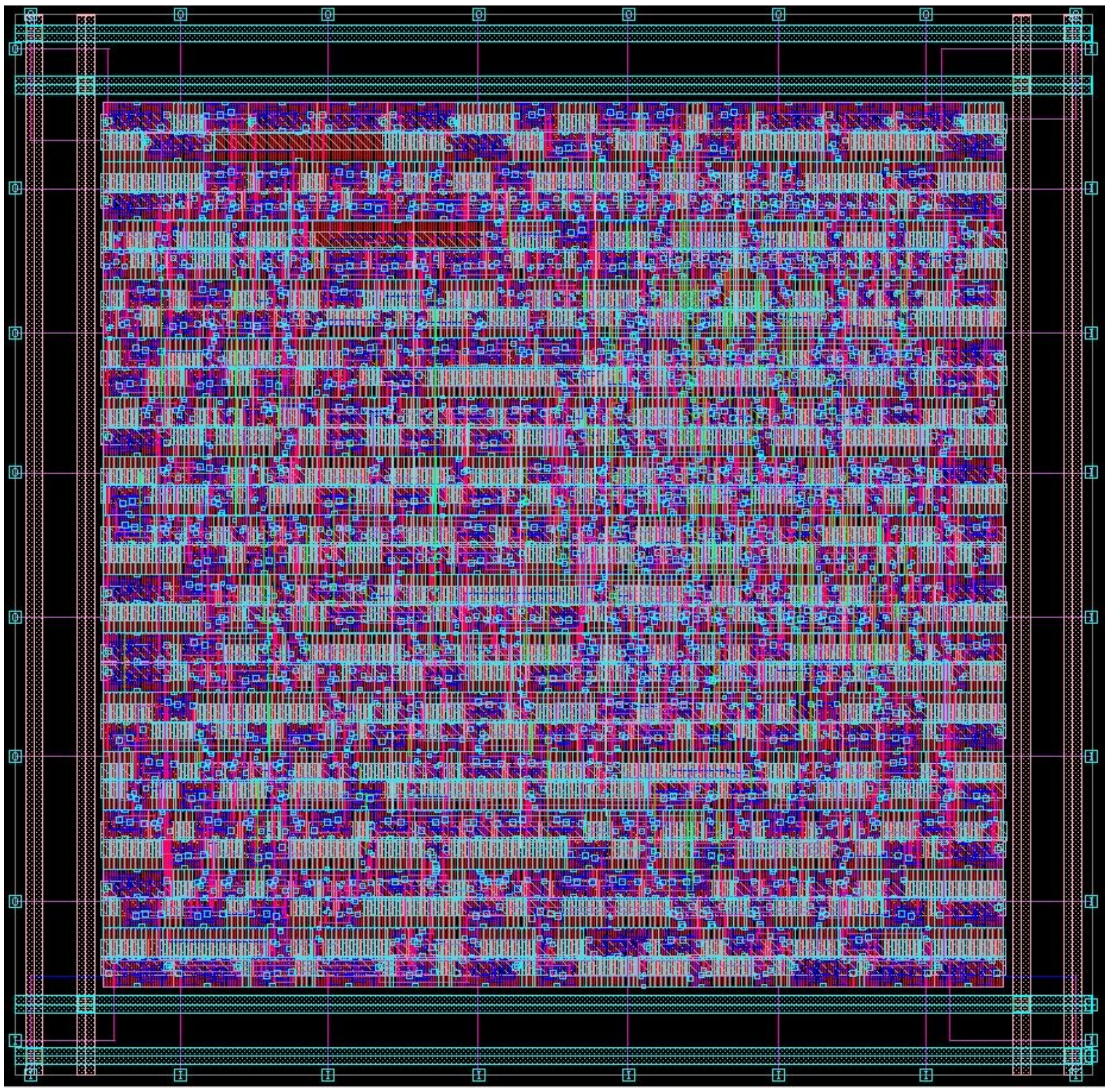
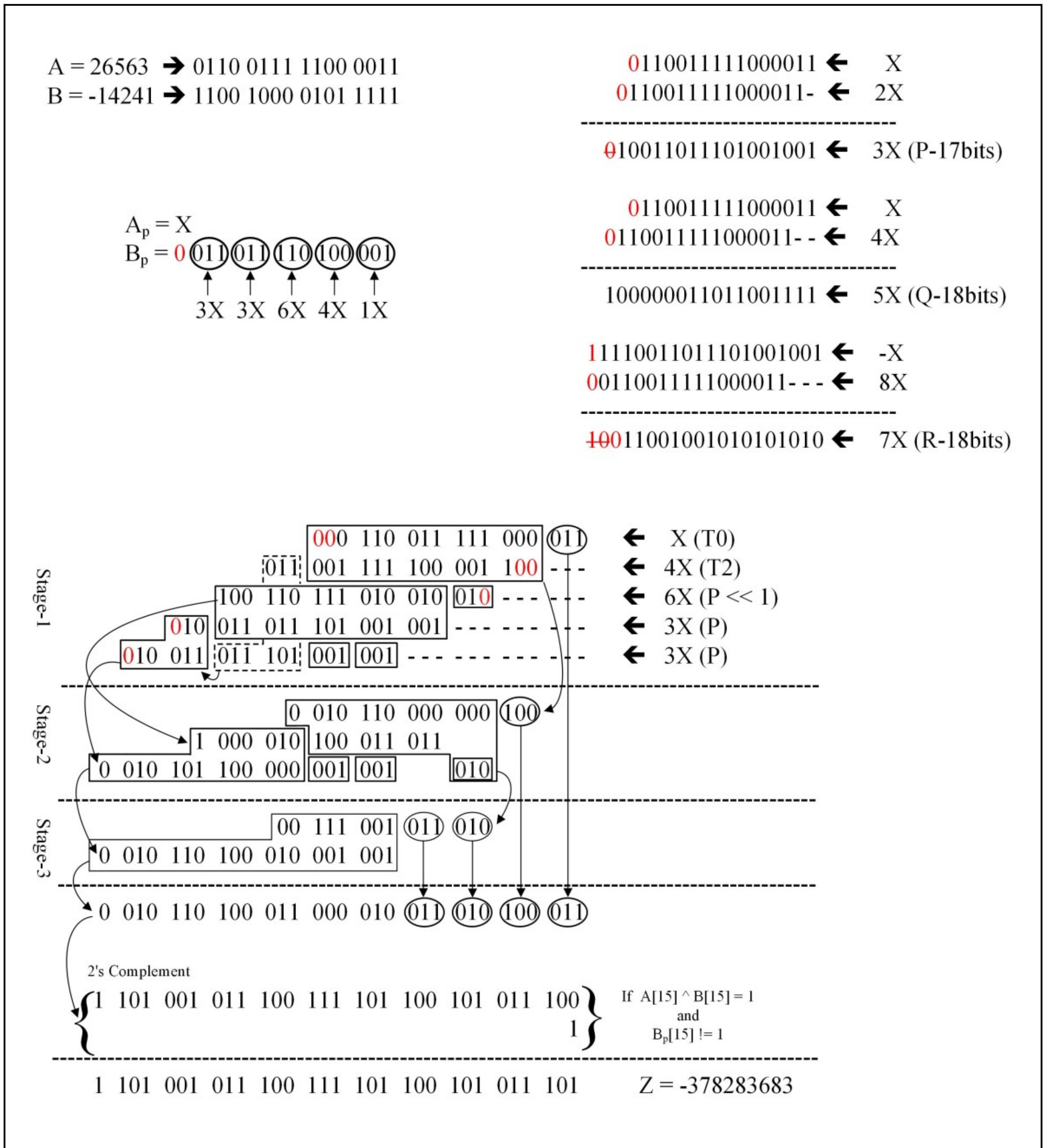


Figure 26 Layout of the proposed 8 x 8 multiplier

4.2 16 x 16 signed multiplication using the proposed design :

The complete working of the proposed radix-8 with the grouping of 3-bit multiplier bits for the 16 x 16 signed multiplier is clearly illustrated using the example-1 and example-2 shown in Figure 27 and Figure 28 respectively. And, the working is like the proposed 8 x 8 signed multiplier structure except few modifications such as increase in number of partial products hence the number of stages, change in the length of adders, and new addition plus BEC plus addition plus BEC block. As shown in the example-1 16 x 16 proposed signed multiplier operation, in the first few steps, magnitudes of both multiplicand and multiplier, A and B respectively, are computed, as the 16-bit vectors A_p (X) and B_p, followed by the computation of the three non-trivial values, 3X (NT1), 5X (NT2), and 7X (NT3), in parallel. The maximum number of bits resulting from the non-trivial computations, additions, are 17-, 18-, and 18-bits for NT1, NT2, and NT3 respectively. Later steps are divided into three stages. In stage 1, five partial products are generated based on the five 3-bit groupings, radix-8 structures, shown by encircling the magnitude bits, B_p, of the multiplier. The sixteenth bit at the MSB position can be ignored since it represents the sign, except in the case of maximum magnitude of 32768 due to the actual multiplier input of -32768 where the actual multiplier value in binary representation equals to the binary value of the magnitude of itself.

In example 1, the multiplicand and multiplier are chosen as 26563 (0110 0111 1100 0011) and -14241 (1100 1000 0101 1111) respectively. Since the actual value of the multiplicand is positive, the magnitude is equal to the actual value. Whereas in case of multiplier, the 2's complement operation is performed to find the magnitude, 14241 (0011 0111 1010 0001), since the actual multiplier value is negative. Except the MSB bit at the 16th bit position, all other 15-bits are grouped into five 3-bit groupings, 011-011-110-100-001. Based on the 3-bit grouping and the radix-8 schema given in Table 4.2, the five partial product values generated in terms of the magnitude of the actual multiplier, X, are X, 4X, 6X, 3X, and 3X respectively. Here X and 4X are trivial values, 3X is first non-trivial value (NT1), and 6X is formed by just left shifting the NT1 by 1-bit position. These partial products are shifted by 0, 3, 6, 9, and 12 to form the PP array.



A = 26563 → 0110 0111 1100 0011
B = -14241 → 1100 1000 0101 1111
0110011111000011 ← X
0110011111000011- ← 2X
010011011101001001 ← 3X (P-17bits)
0110011111000011 ← X
0110011111000011- - ← 4X
100000011011001111 ← 5X (Q-18bits)
11110011011101001001 ← -X
00110011111000011- - - ← 8X
10011001001010101010 ← 7X (R-18bits)
A_p = X
B_p = 0 011 011 110 100 001
3X 3X 6X 4X 1X
000 110 011 111 000 011 ← X (T0)
011 001 111 100 001 100 ← 4X (T2)
100 110 111 010 010 010 ← 6X (P << 1)
010 011 011 101 001 001 ← 3X (P)
010 011 011 101 001 001 ← 3X (P)
Stage-1
Stage-2
Stage-3
0 010 110 000 000 100
1 000 010 100 011 011
0 010 101 100 000 001 001 010
00 111 001 011 010
0 010 110 100 010 001 001
0 010 110 100 011 000 010 011 010 100 011
2's Complement
1 101 001 011 100 111 101 100 101 011 100
1
If A[15] ^ B[15] = 1 and B_p[15] != 1
1 101 001 011 100 111 101 100 101 011 101
Z = -378283683

Figure 27 Example-1 calculation of the 16 x 16 signed multiplication using the proposed radix-8 architecture

The five partial products from the precomputation stage are subjected to stage 1 reduction, summation. Each partial product is represented as six 3-bit groups since the partial products are shifted by 3-bit position due to the radix-8 scheme. The two rectangular solid boxes enclosing two 15-bit parts of the partial products are added to reduce. The two mirrored L-shape blocks perform addition plus BEC like operation plus addition plus BEC like operation. In any stage, if the grouping results in more BEC like operation, then it is an advantage for improving the speed while keeping the design area and power consumption low. The encircled group of 3-bits does not need any further processing and constitutes towards the final magnitude of the product. The three lone rectangular blocks containing 3-bits each: LSB block of the 3rd partial product and the two LSB blocks of the 5th partial product carried forward to the next stage unprocessed. The two 15-bit adder blocks yield 16-bit output, and the two L-shaped blocks together results in a 13-bit output. All the outputs and the blocks carried forward to the next stage, stage 2, are further processed to reduce.

In stage 2, the bits enclosed under the two solid boxes are added separately and the 3-bit groups enclosed by the small rectangular blocks are carried forward to the next stage unprocessed. The encircled 3-bit group constitute towards the magnitude of the final product.

In stage 3, the first two 3-bit groups of output resulted from the first adder of stage 2 constitute towards the magnitude of the final product as 3rd and 4th groups from the LSB side. The remaining output bits of first adder, all the output bits of the second adder, and the unprocessed 3-bit groups from stage 2 are summed together in the final stage of addition. The MSB side has more BEC like structure which is faster with less design area and power requirements. In post computation stage, the magnitude of the final product is 2's complemented based on the logic that if the two actual inputs are opposite in sign and the B_p is not equal to ‘1’. B_p is equal to ‘1’ in only one case where the multiplier is at its maximum magnitude of 32768 (1000 0000 0000 0000). At maximum magnitude case, the actual value of the signed integer and the 2’s complemented value of itself are same. In the maximum magnitude case, the final product is simply a 15-bit shifted, and 2’s complemented value of the multiplicand. In this example, example 1, the magnitude of the product value needs to be 2’s complemented to get the actual final product.

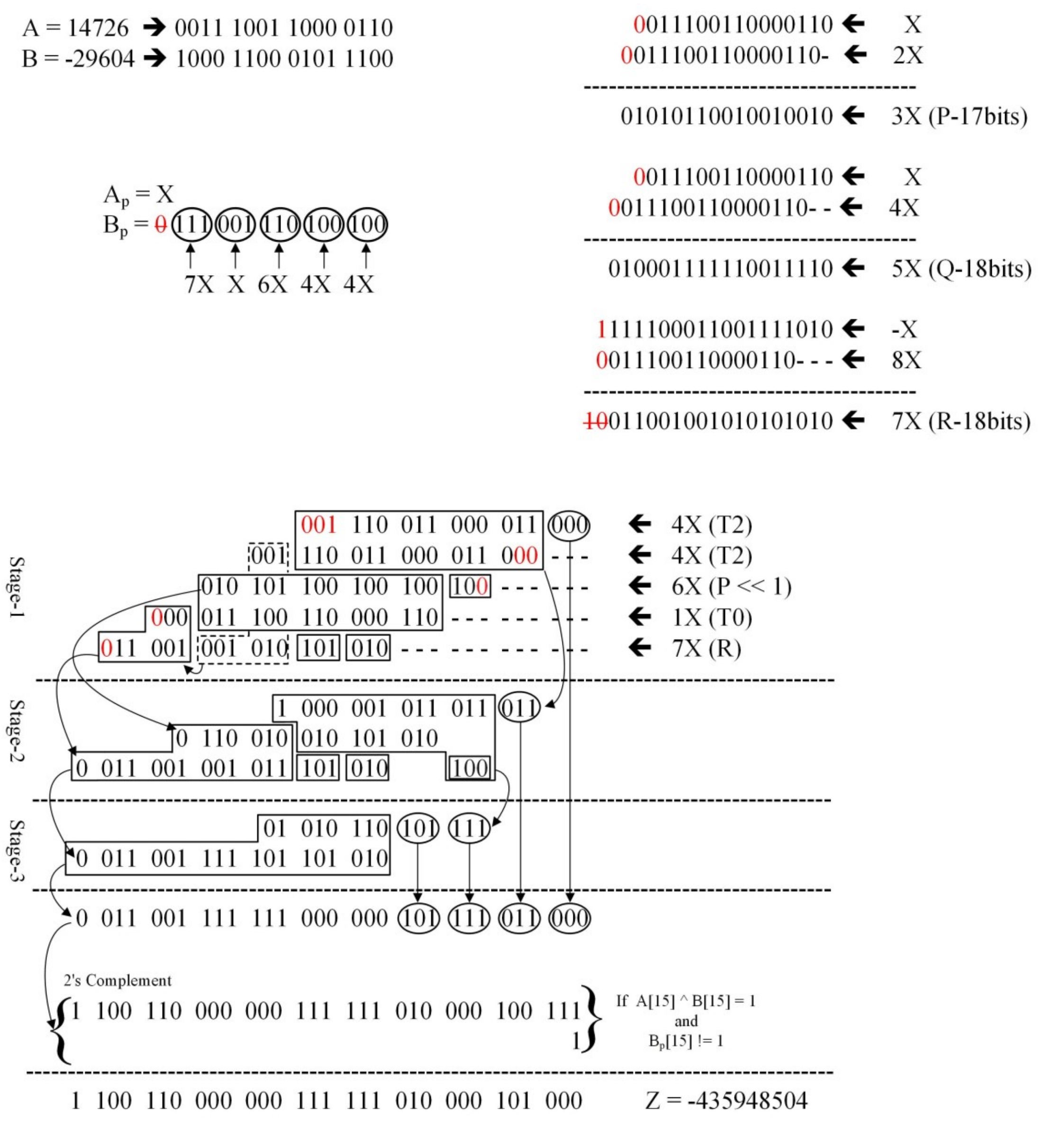

A = 14726 → 0011 1001 1000 0110
B = -29604 → 1000 1100 0101 1100
0011100110000110 ← X
0011100110000110- ← 2X
01010110010010010 ← 3X (P-17bits)
0011100110000110 ← X
0011100110000110- - ← 4X
010001111110011110 ← 5X (Q-18bits)
1111100011001111010 ← -X
0011100110000110- - - ← 8X
10011001001010101010 ← 7X (R-18bits)
A_p = X
B_p = 0 111 001 110 100 100
7X X 6X 4X 4X
Stage-1
001 110 011 000 011 000 ← 4X (T2)
001 110 011 000 011 000 ← 4X (T2)
010 101 100 100 100 100 ← 6X (P << 1)
000 011 100 110 000 110 ← 1X (T0)
011 001 001 010 101 010 ← 7X (R)
Stage-2
1 000 001 011 011 011
0 110 010 010 101 010
0 011 001 001 011 101 010 100
Stage-3
01 010 110 101 111
0 011 001 111 101 101 010
0 011 001 111 111 000 000 101 111 011 000
2's Complement
1 100 110 000 000 111 111 010 000 100 111
1
If A[15] ^ B[15] = 1 and B_p[15] != 1
1 100 110 000 000 111 111 010 000 101 000
Z = -435948504

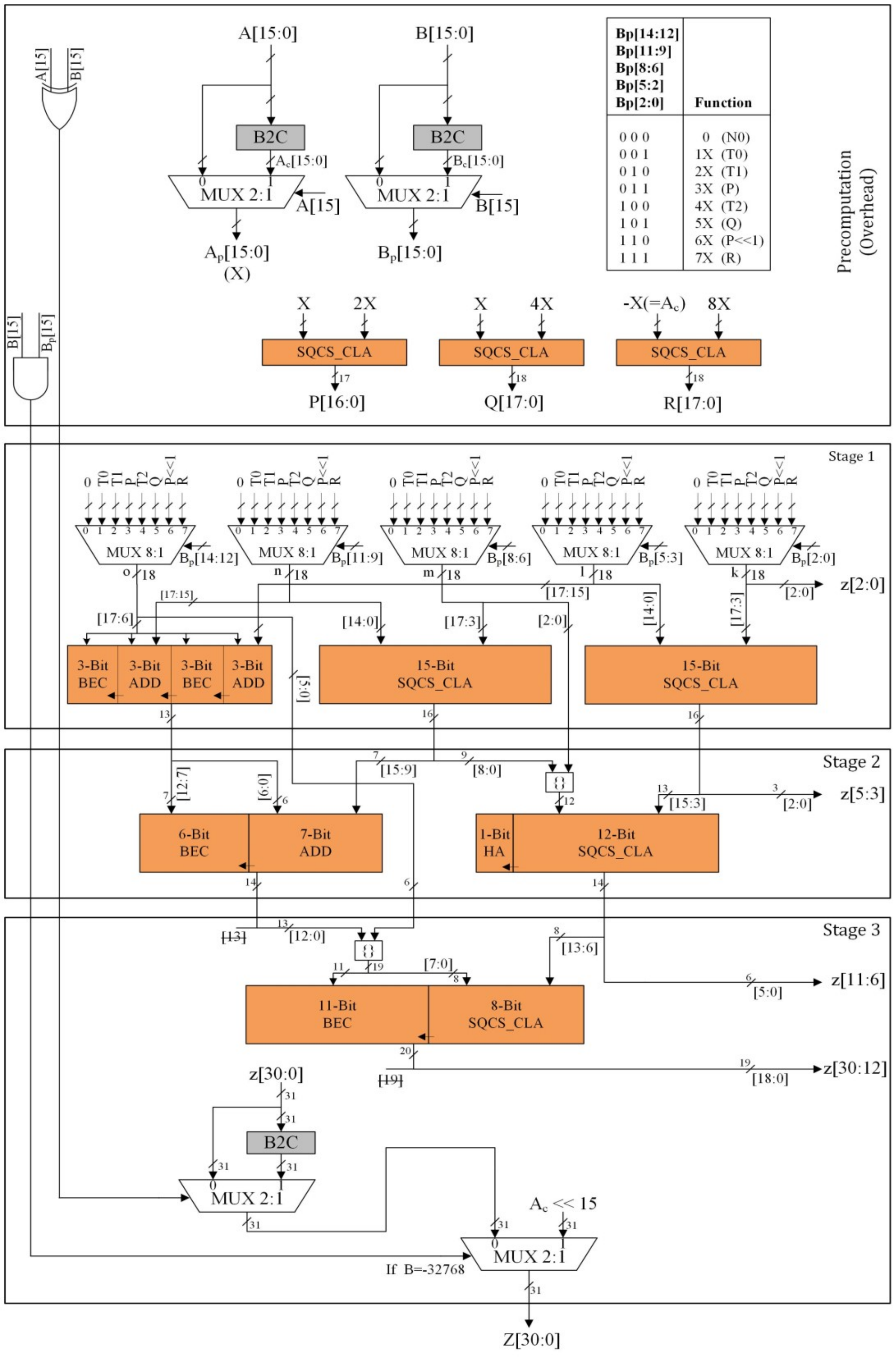
Figure 28 Example-2 calculation of the 16 x 16 signed multiplication using the proposed radix-8 architecture

The computation procedure in example 2, shown in Figure 28, is exactly same as example 1 described and illustrated above. In preprocessing stage, multiplier value is 2's complemented from (1000 1100 0101 1100) to (0111 0011 1010 0100) to compute the magnitude, B_p, and the multiplier magnitude value, A_p (X), stays same since it is a positive number. The three non-trivial values, 3X, 5X, and 7X, are computed and used based on the 3-bit grouping of the 15-bits correspond to the magnitude of the multiplier to generate the five partial products. The five partial products are subjected to reduction, summation, in stage 1 results in 3 partial products to be processed in stage 2. Stage 2 results in two partial products for stage 3 to be processed. Stage 3 addition results in the final magnitude of the product. The actual sign of the final product is determined based on the signs of the actual inputs. Magnitude value of the final product is 2's complemented if the signs are opposite otherwise actual final product value stays the same as the magnitude. As mentioned above, in an exclusive case of multiplier being the lowest, highest in magnitude, for the given number of input bits, the final product is calculated simply by left shifting the multiplicand by 15-bit positions followed by the 2's complement.

The architecture of the proposed 16 x 16 design is implemented using Synopsys RVT standard cells and is shown in the following Figure 29. In the precomputation stage, the magnitudes of the two 16-bit signed inputs, multiplicand (A) and multiplier (B), to be multiplied are computed as A_p (X) and B_p. After the magnitude conversion, the three non-trivial computation blocks, each one is made of SQCS-CLA adder, compute 3X (NT1), 5X (NT2), and 7X (NT3). The results of the non-trivial blocks are represented as vectors P, Q, and R of width 17-, 18-, and 18-bits respectively.

In stage 1, five 18-bit 8 to 1 multiplexers are used to select one of the eight possible values, {0, T0, T1, P (NT1), T2, Q (NT2), P<<1, R (NT3)}, of the partial product to form the five partial products. The 3-bit groups encircled, shown in the examples for the proposed 16 x 16 multiplier, are used as selector inputs, $B_p[2:0]$, $B_p[5:3]$,

B_p[8:6], Bp[11:9], and Bp[14:12]. Partial product generation starts from using the 3-bit grouping of the bits of B_p from the LSB side. All the partial products are generated in parallel and are of 18-bits wide. The partial products from the multiplexers are represented as vectors, k, l, m, n, and o, from right to left. The five generated partial products are subjected to reduction using two 15-bit SQCS-CLAs, yields 16-bit outputs, and a 12-bit addition plus BEC like operation block, yields 13-bit output, shown in the stage 1 of the Figure 29. The first 3-bits, k[2:0] on the LSB side of the first partial product on the right side, k, are assigned to the first 3-bits on the LSB side of the product magnitude, z[2:0]. Rest of the 15-bits of k, k[17:3], are fed to the 15-bit adder along with the LSB side 15-bits of the second partial product, l[14:0]. The 15-bits on the MSB side of the third partial product, m[17:3], and the first 15-bits on the LSB side of the fourth partial product, n[14:0], are fed to the 15-bit SQCS-CLA in the middle. Finally, the MSB side 12-bits of the fifth partial product, o[17:6], and the two MSB side 3-bit groups unprocessed of the second and fourth partial product, l[17:15] and n[17:15], are fed to the ADD plus BEC plus ADD plus BEC structure. Carry outs from each 3-bit sub-block is fed to the next 3-bit block towards the left side. The two 15-bit adders result in 16-bit outputs and the ADD plus BEC structure results in a 13-bit output.



Bp[14:12] Bp[11:9] Bp[8:6] Bp[5:2] Bp[2:0]	Function
0 0 0	0 (N0)
0 0 1	1X (T0)
0 1 0	2X (T1)
0 1 1	3X (P)
1 0 0	4X (T2)
1 0 1	5X (Q)
1 1 0	6X (P<<1)
1 1 1	7X (R)

Figure 29 Proposed 16 x 16 signed multiplier with radix-8 architecture

Stage 2 takes the three output vectors from stage 1 and further reduces to two output vectors using a 12-bit adder plus a single bit half adder (HA) structure and a 7-bit ADD plus 6-bit BEC structure. The first three bits, LSB bits, of the output vector from the first output vector from the stage 1 on the right side are assigned to z[5:3]. The MSB side 12-bits of the first output vector of stage 1 along with the concatenated 12-bits formed using the LSB side 9-bits of the second output vector from stage 1 and the three unprocessed three LSB bits of the third partial product, m[2:0], from stage 1 are fed to the SQCS-CLA plus HA block. The 13-bit output vector from the ADD plus BEC block of the second stage and the 7-bits on the MSB side of the second output vector, in the middle, of stage 1 are fed to the 7-bit ADD plus the 6-bit BEC block on the left side of the stage 2 enclosed in a solid rectangle. The fourteenth bit at index 13 can be ignored since it does not contribute to the 31-bits of the product output. Since the 7-bit addition is not in the time critical path, a simple ripple carry adder (RCA) is used instead of a fast SQCS-CLA to cut the additional cost, in terms area and power, of computation. Both addition blocks outputs 14-bit vectors.

In stage 3, the six LSB bits of the first output vector on the right side of stage 2 are assigned to z[11:6] and the rest of the MSB bits, 8-bits, are fed to the 8-bit SQCS-CLA along with the eight LSB bits of the vector formed by the concatenation of the six unprocessed bits of the fifth partial product generated in stage 1, o[5:0], and the thirteen LSB bits of the output from the 7-bit ADD plus the 6-bit BEC structure on the left side of stage 2, shown in the Figure 40. Rest of the eleven MSB bits of the concatenated vector are fed to the 11-bit BEC structure with one more bit of input, carry input, from the carry output of the 8-bit SQCS-CLA. The 19-bit addition carried out using the SQCS-CLA and the light weight BEC results in a 20-bit output vector. The 20th bit at index 19 can be ignored since it contributes to the 32nd bit of the product output, which is out of bounds of the expected 31-bit result. The LSB 19-bits are assigned to the 19 MSB bits of the magnitude of the product, z[30:12]. The post processing steps include deciding the sign of the final product, performing the sign conversion, and performing the one special case of multiplication where multiplier magnitude is at its maximum 32768. The sign of the partial product determined by performing the XOR operation on the MSB bits of the actual inputs, A[15] and B[15]. If the two signs of the inputs are opposite then the XOR results in a logic '1' which is fed to the 31-bit wide 2 to 1 MUX to choose the 2's complemented magnitude, z, value. The next and the final 31-bit wide 2 to 1 MUX chose between the product magnitude value decided by the preceding MUX and product obtained from the extreme case of multiplying multiplicand with the maximum magnitude multiplier, B, value. The final MUX is necessary since the extreme case of multiplying the multiplicand with the lowest 16-bit signed integer value is omitted in the previous stages. If the multiplier has the maximum magnitude value of 32768 due to its actual value of -32768, the final product will be simply the 15-bit left shifted multiplicand magnitude, A_c, value.

Layout of the proposed 16 x 16 signed multiplier with a simple and new radix-8 structure for partial product generation and new grouping strategy for the partial product reduction, which is synthesized at 250 MHz clock frequency and performed PnR, is shown in Figure 30. Total chip area of the layout shown in the following figure is 7585.69 $(\mu\text{m})^2$ with the standard cell utilization factor of 61.18%. Hence, the total cell area is nearly 4641 $(\mu\text{m})^2$ with the total number of standard cells used as 1516.

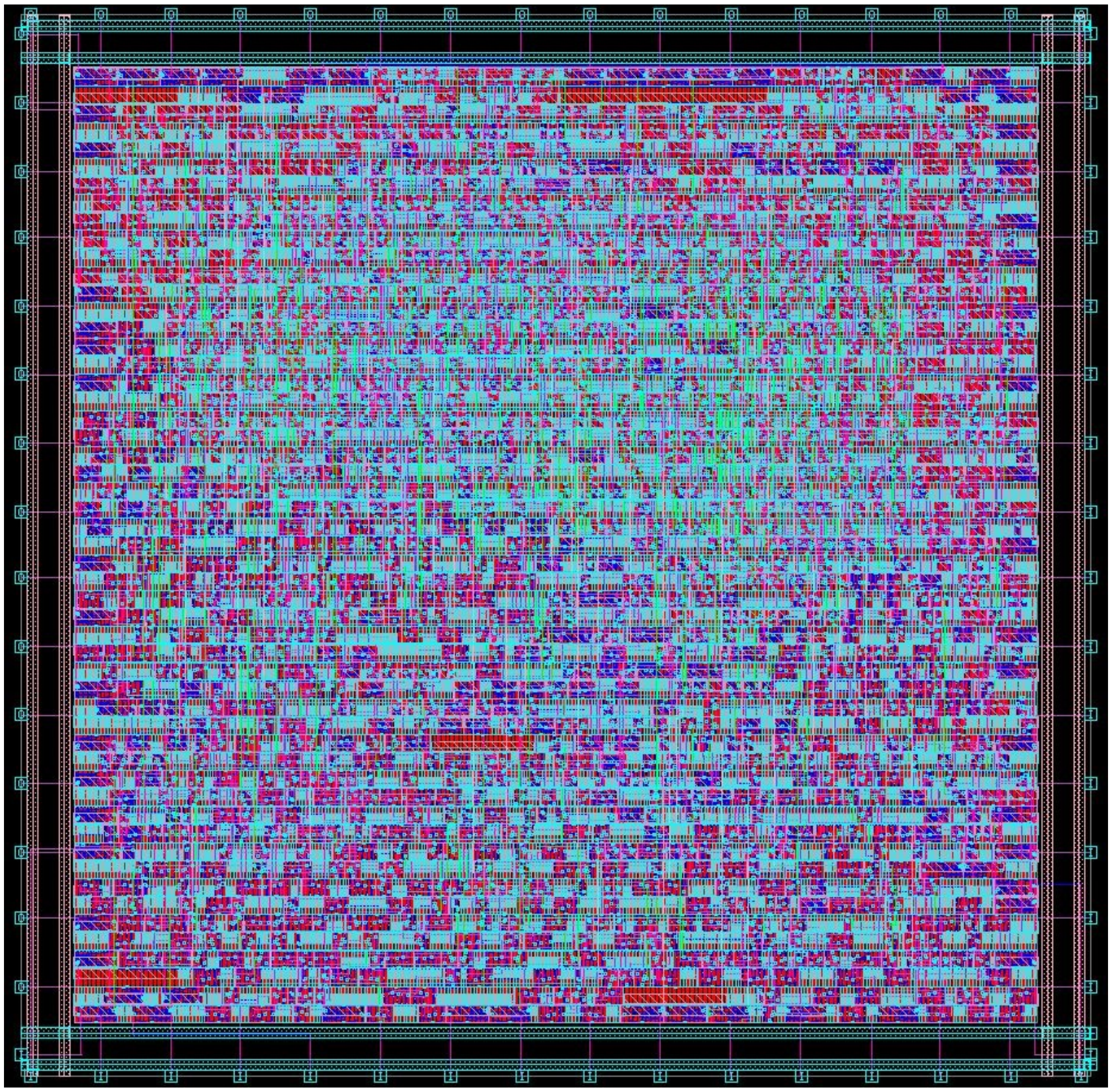


Figure 30 Layout of the proposed 16 x 16 multiplier

4.3 32 x 32 signed multiplication using the proposed design :

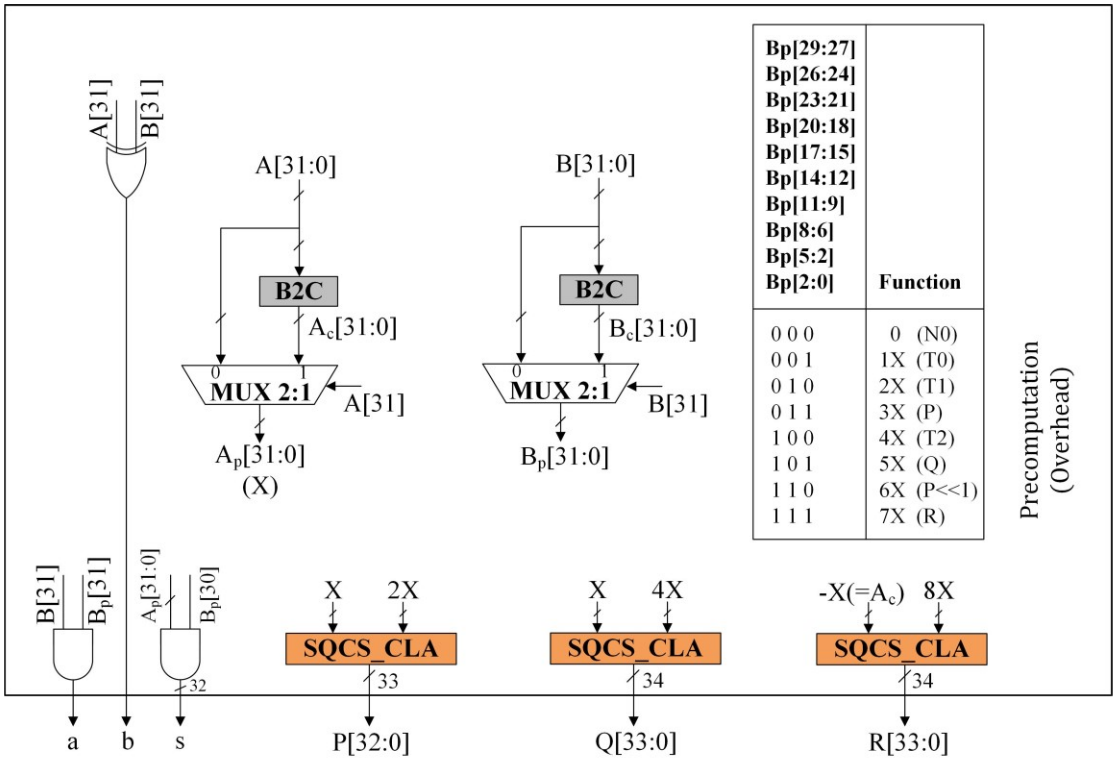


Figure 31 Stage-0 of the proposed 32 x 32 signed multiplier with radix-8 architecture

Like the proposed 8-bit and 16-bit signed multiplier architectures, 32-bit signed multiplier architecture in the precomputation stage, shown in Figure 31, majorly involves in magnitude computation for multiplicand (A) and multiplier (B) and the three non-trivial computations, P, Q, and R, required to be performed once to be used in stage-1 multiple times equal to the number of 3-bit groupings. Non-trivial computations are performed using the magnitude of the multiplicand, A_p or X. The 3-bit grouping required to be performed from the LSB side to MSB side of the magnitude bits of the multiplier, B_p, at the initial stages for the proposed radix-8 structure is left with 10 3-bit groupings and one lone MSB bit, $B_p[30]$. The 10 3-bit groupings generate 10 partial products, k, l, m, n, o, d, e, f, g, and h, of 34-bit wide and the lone bit, $B_p[30]$, generates one more partial product of value equals to 0 or X based on its binary value.

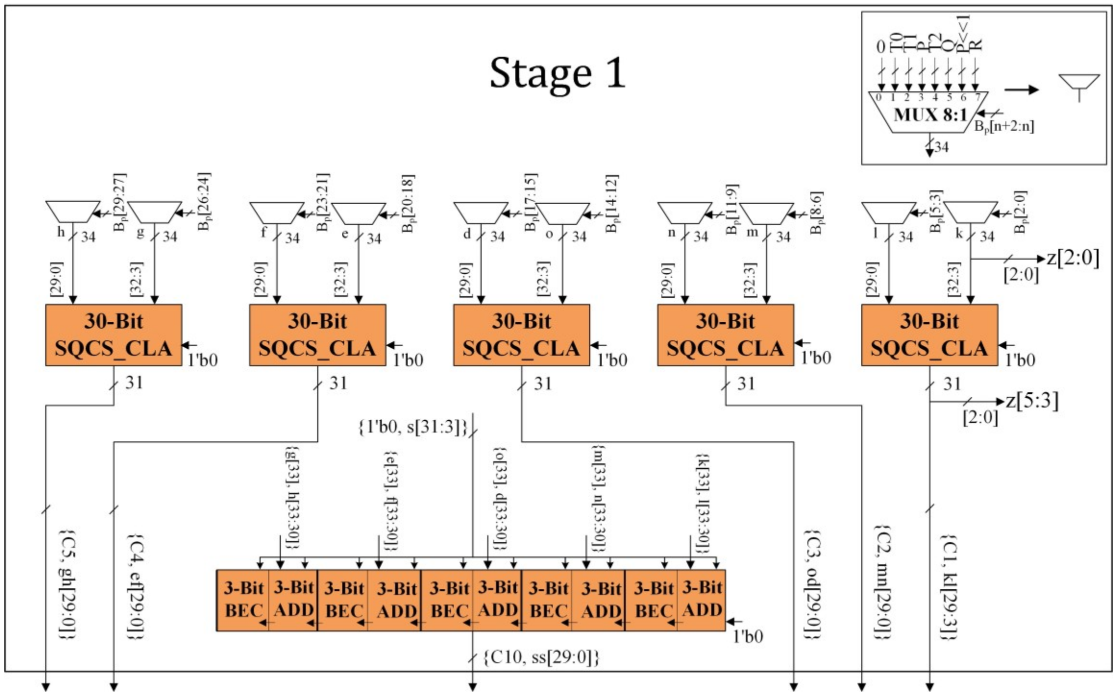


Figure 32 Stage-1 of the proposed 32 x 32 signed multiplier with radix-8 architecture

The partial products generated are reduced to find the final product magnitude in 4 stages, from stage-1 to stage-4. In stage-1, each 3-bit grouping is used as selection input bits at each multiplexer, shown in Figure 32, to choose between one of eight possible values which includes both trivial and non-trivial values. The first 10 partial products chosen by the 10 multiplexers, with each multiplexer of 8-inputs of 34-bit wide, are reduced to 6 partial products using 5 30-bit SQCS with carry look-ahead adder blocks and a chain consisting of 5 pairs of 3-bit adder and 3-bit BEC like structure. As shown in figure, all the six reduced partial products generated in stage-1 are of 31-bit wide, including carry output. The reduced partial products in the current stage are named kl, mn, od, ef, gh, and ss with the carry outputs, C1, C2, C3, C4, C5, and C10 respectively, associated with. The first six LSB bits of the final product magnitude, z[5:3] and z[2:0], are resulted in this stage.

The six partial products generated in stage-1 are further reduced in stages 2, 3, and 4, shown in Figure 33, to generate the final product magnitude, z, and the final product, Z. Stage-2 further reduced the partial products reduced in stage-1 with the help of three 27-bit SQCS with CLA blocks and a 6-bit BEC like block. Stage-2 results in further reduced partial products and the six bits of final product magnitude, z[11:6]. Stage-3 further reduces the partially reduced partial products from stage-w with the help of one 22-bit SQCS with CLA and two block of 7-bit with SQCS with CLA plus the 6-bit BEC like structure. Stage-3 results in further partially reduced partial products and 12 bits of result final product magnitude, z[23:12]. Finally, stage-4 reduces the partially reduced partial products from the previous stage using one 9-bit SQCS with CLA, one 12-bit SQCS with CLA, two blocks consisting of half adder, full adder, and 11-bit BEC like structure, one block with two full adders and 10-bit BEC like structure, and one block with two full adders and 4-bit BEC like structure. After four stages of partial product reduction and the final product magnitude generation, the B2C block and the two multiplexers are used to decide the sign of the final product and to deal with the extreme case of multiplying with the maximum multiplier magnitude value at B = -2147483648, which is omitted in the initial stages of processing. The extreme case bypasses the previous computation stages and simply generates the output by left shifting the two's complemented multiplicand value by 31-bit positions.

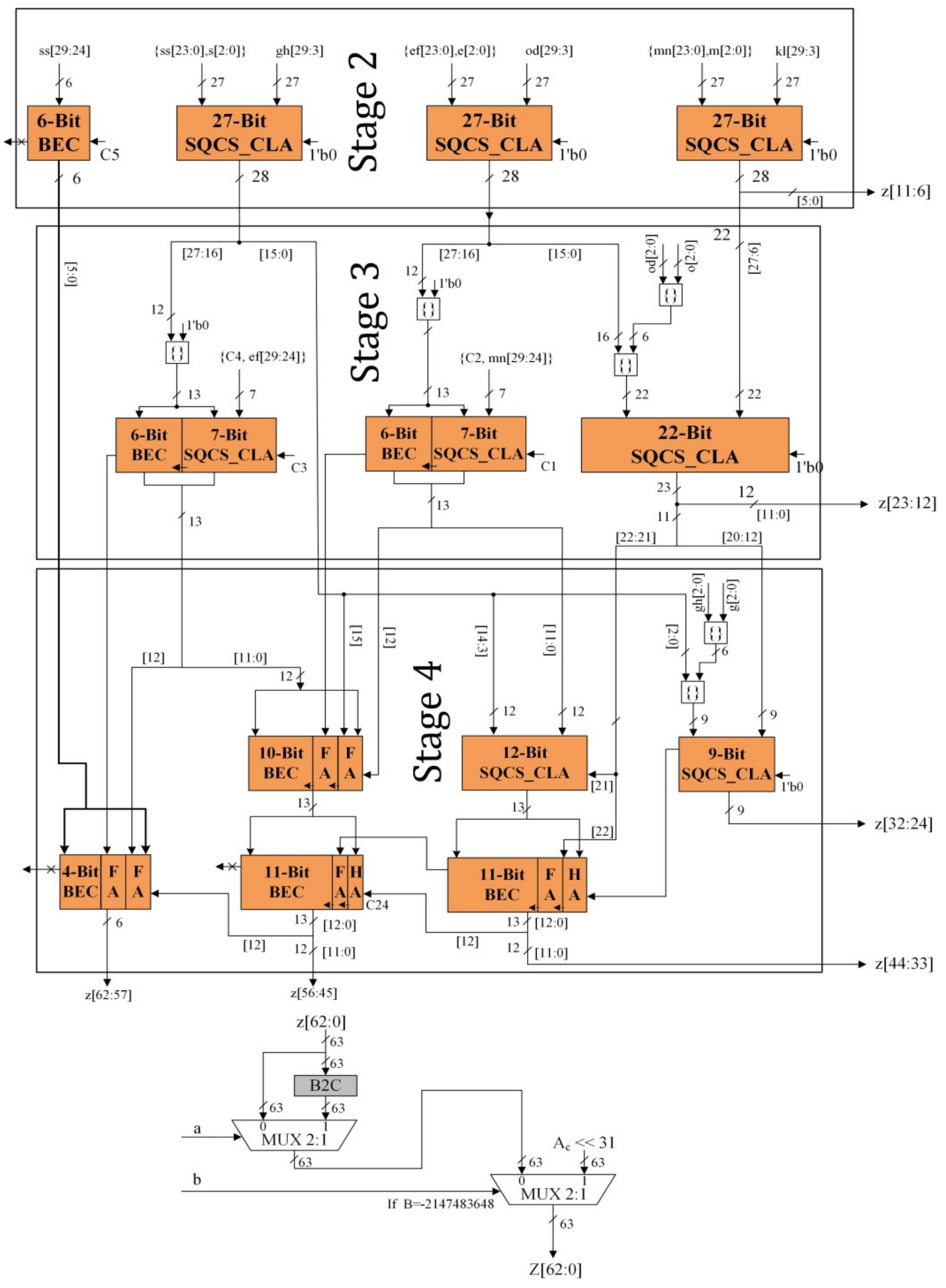


Figure 33 Stages 2, 3, and 4 of the proposed 32 x 32 signed multiplier with radix-8 architecture

Layout of the proposed 32 x 32 signed multiplier with a simple and new radix-8 structure for partial product generation and new grouping strategy for the partial product reduction, which is synthesized at 200 MHz clock frequency and performed PnR, is shown in Figure 34. Total chip area of the layout shown in the following figure is 29292.637 $(\mu m)^2$ with the standard cell utilization factor of 60.56%. Hence, the total cell area is nearly 17741 $(\mu m)^2$ with the total number of standard cells used as 5856.

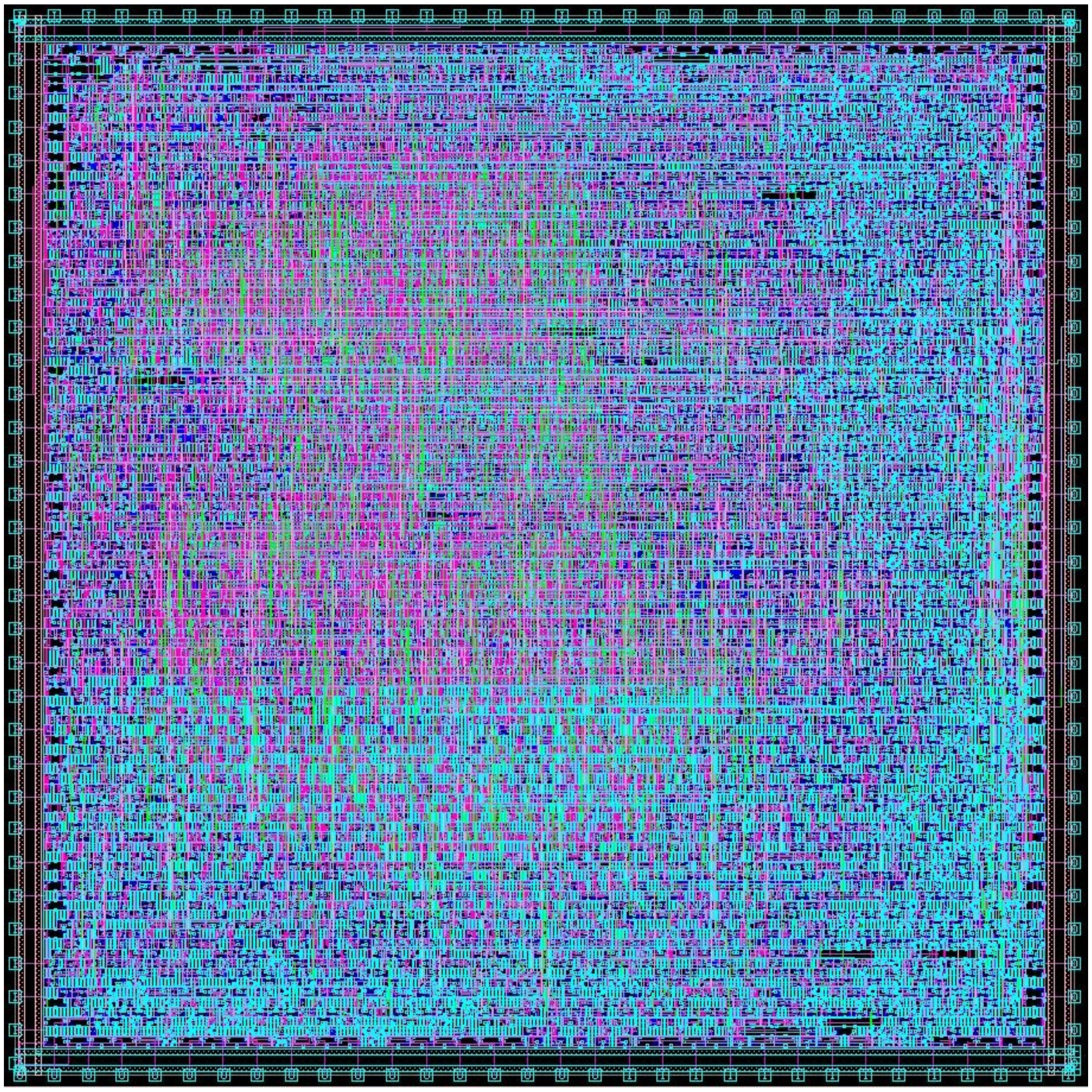
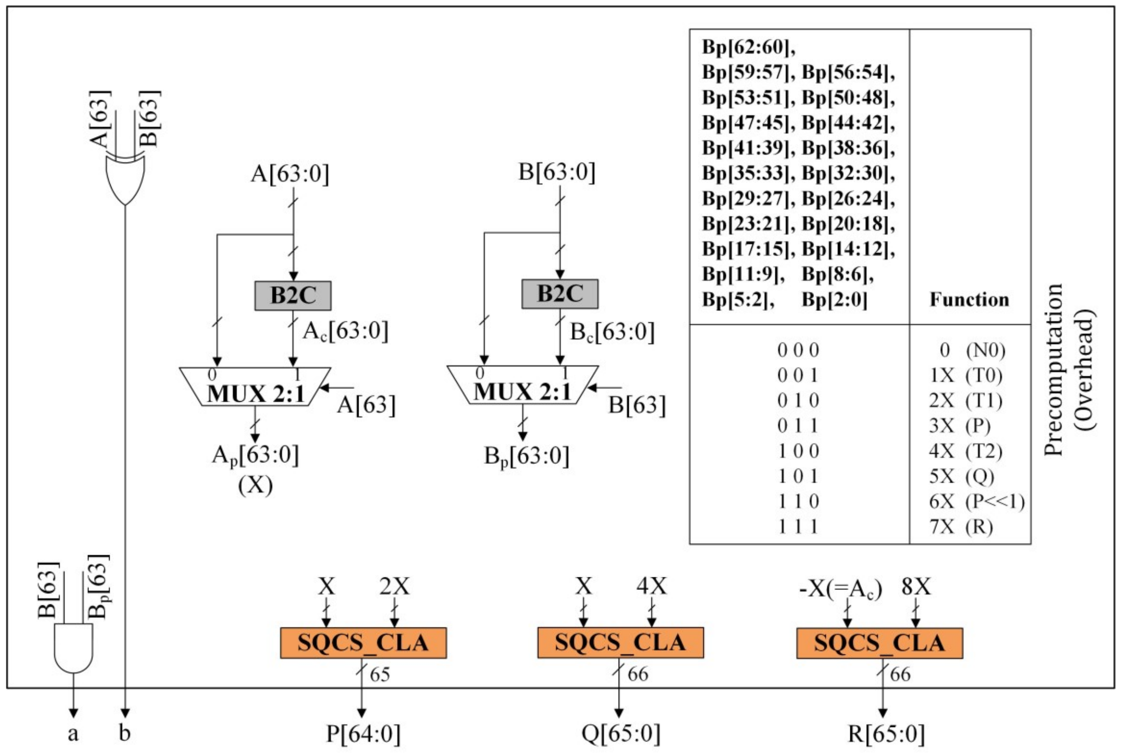


Figure 34 Layout of the proposed 32 x 32 multiplier

4.4 64 x 64 signed multiplication using the proposed design :



Bp[62:60], Bp[59:57], Bp[56:54], Bp[53:51], Bp[50:48], Bp[47:45], Bp[44:42], Bp[41:39], Bp[38:36], Bp[35:33], Bp[32:30], Bp[29:27], Bp[26:24], Bp[23:21], Bp[20:18], Bp[17:15], Bp[14:12], Bp[11:9], Bp[8:6], Bp[5:2], Bp[2:0]	Function
0 0 0	0 (N0)
0 0 1	1X (T0)
0 1 0	2X (T1)
0 1 1	3X (P)
1 0 0	4X (T2)
1 0 1	5X (Q)
1 1 0	6X (P<<1)
1 1 1	7X (R)

Figure 35 Stage-0 of the proposed 64 x 64 signed multiplier with radix-8 architecture

The precomputation stage, stage-0, of the proposed 64 x 64 signed multiplier with radix-8 structure is like 8-bit, 16-bit, and 32-bit proposed architectures discussed earlier, especially like the stage-0 of the 16-bit multiplier since all the bits of the magnitude of the multiplier are 3-bit grouped without leaving with any lone MSB bit. Computation of magnitudes of the multiplicand and the multiplier, A_p and B_p respectively, and the non-trivial computations, P, Q, and R, are computed in the current stage resulting in 65-bit, 66-bit, and 66-bit wide outputs respectively. The simple control logic blocks required to deal with the maximum magnitude of the multiplier case and to decide the sign of the final product, a and b respectively, are also implemented in the current stage of precomputation.

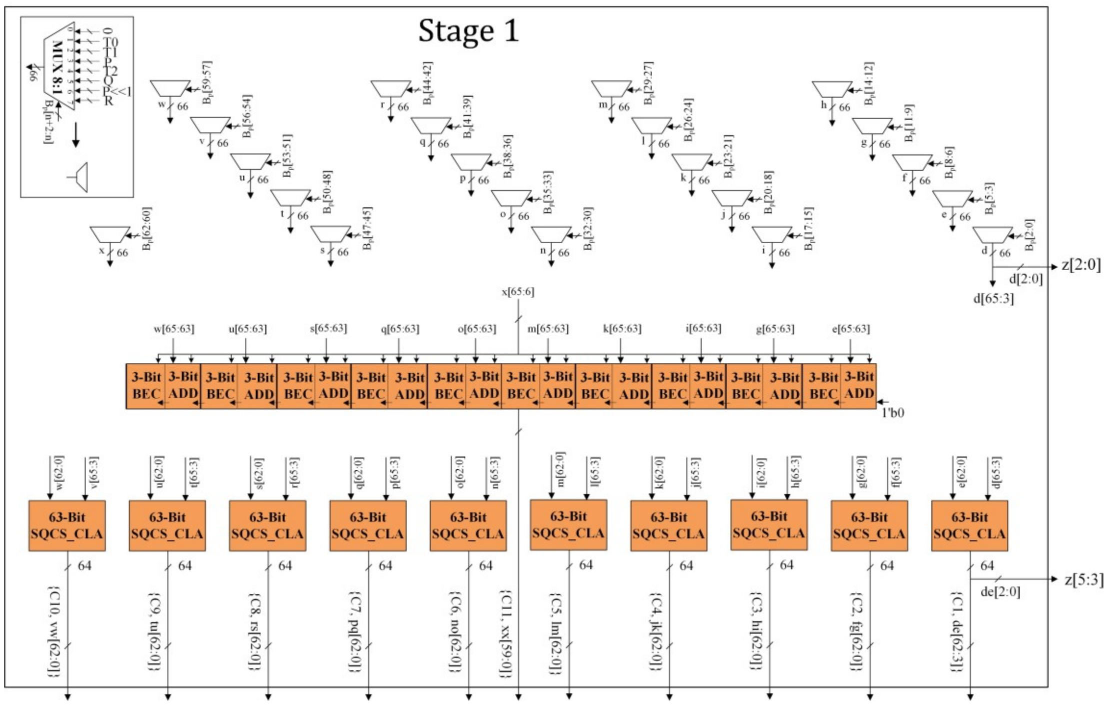


Figure 36 Stage-1 of the proposed 64 x 64 signed multiplier with radix-8 architecture

Excluding the maximum magnitude multiplier case, which is dealt at the final stage by bypassing major computation stages 1 to 5, the 63 bits of magnitude are grouped into 21 groups to generate 21 partial products by using each group of 3-bits as selection bits to choose between one of the 8 possible inputs discussed earlier in the case of the proposed smaller width multiplier architectures. As shown in Figure 36, each multiplexer in stage-1 results in a 66-bit output. The 21 partial products generated are labeled d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, and x from LSB to MSB side. The 21 partial products are reduced to 11 partially reduced partial products using the help of 10 63-bit SQCS with CLA blocks and one block of chain of 10 pairs of simple computing blocks each pair includes a 3-bit adder and a 3-bit BEC like structure. Partially reduced partial products are labeled as de, fg, hi, jk, lm, no, pq, rs, tu, vw, and xx. The first 6-bits of the magnitude of the final product, z[2:0] and z[5:3], are resulted in the current stage.

The 11 partially reduced partial products resulted in the previous stage are further reduced by using 5 51-bit SQCS with CLA blocks and a block with a chain of 5 pairs of computation blocks with each pairing consisting of a 6-bit adder and a 6-bit BEC like structure, shown in Figure 37. Stage-2 results in 6-bits of magnitude of final product, z[11:6]. The further reduced partial products are labeled as defg, hijk, lmno, pqrs, tuvw, and xxxx with the associated MSB bit and carry outputs pairs generated at each block as {C12, C13}, {C14, C15}, {C16, C17}, {C18, C19}, {C20, C21}, and {C22, C23} respectively.

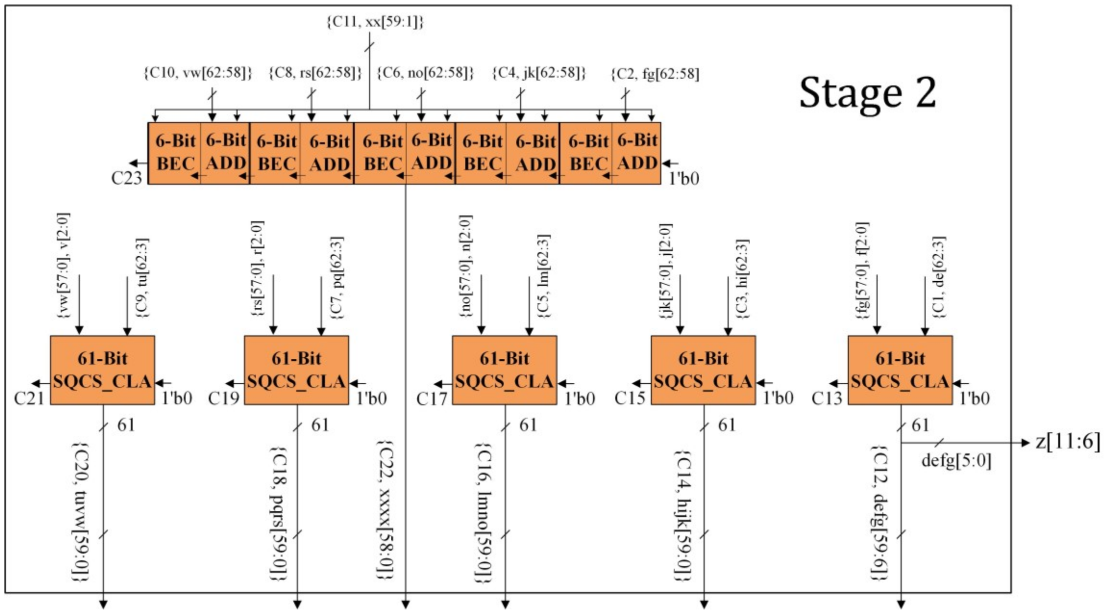


Figure 37 Stage-2 of the proposed 64 x 64 signed multiplier with radix-8 architecture

Stages 3, 4, and 5, shown in Figure 38, involve further reduction of partially reduced partial products from the previous stages. Further reduction of partial products is in stage-3 is performed using one 56-bit wide, one 48-bit wide, and two 32-bit wide SQCS with CLA blocks, one 11-bit BEC like structure and two 12-bit BEC like structures. Stage-3 results in 12-bits of magnitude of the final product, z[23:12]. Stage-4 further reduces the previously reduced partial products using a 45-bit SQCS with CLA block, a 37-bit SQCS with CLA, a 12-bit BEC like structure, and a 22-bit BEC like structure. Stage-4 results in 24-bits of the magnitude of the final product, z[47:24]. Partial product reduction in stage-5 uses one 34-bit SQCS with CLA and one 45-bit BEC like structure. Stage-5 results in the MSB 79-bits of the magnitude of the final product, z[126:82] and z[81:48]. Finally, B2C block converts the sign of the magnitude of the final product, and the two multiplexers, with a and b as selection signals, decide the sign of the final product and selects the final product in the case of maximum magnitude of the multiplier case, which is omitted in the starting stages, respectively. Each multiplexer takes 127-bit inputs and results in 127-bit output. In case of dealing with the extreme case of computing the final product with the least multiplier value, highest possible magnitude for the current multiplier size, the corresponding input is simply generated by left shifting the two's complemented multiplicand value, A_c, by 63-bit positions. The final product output of 127-bit wide is labeled as Z[126:0].

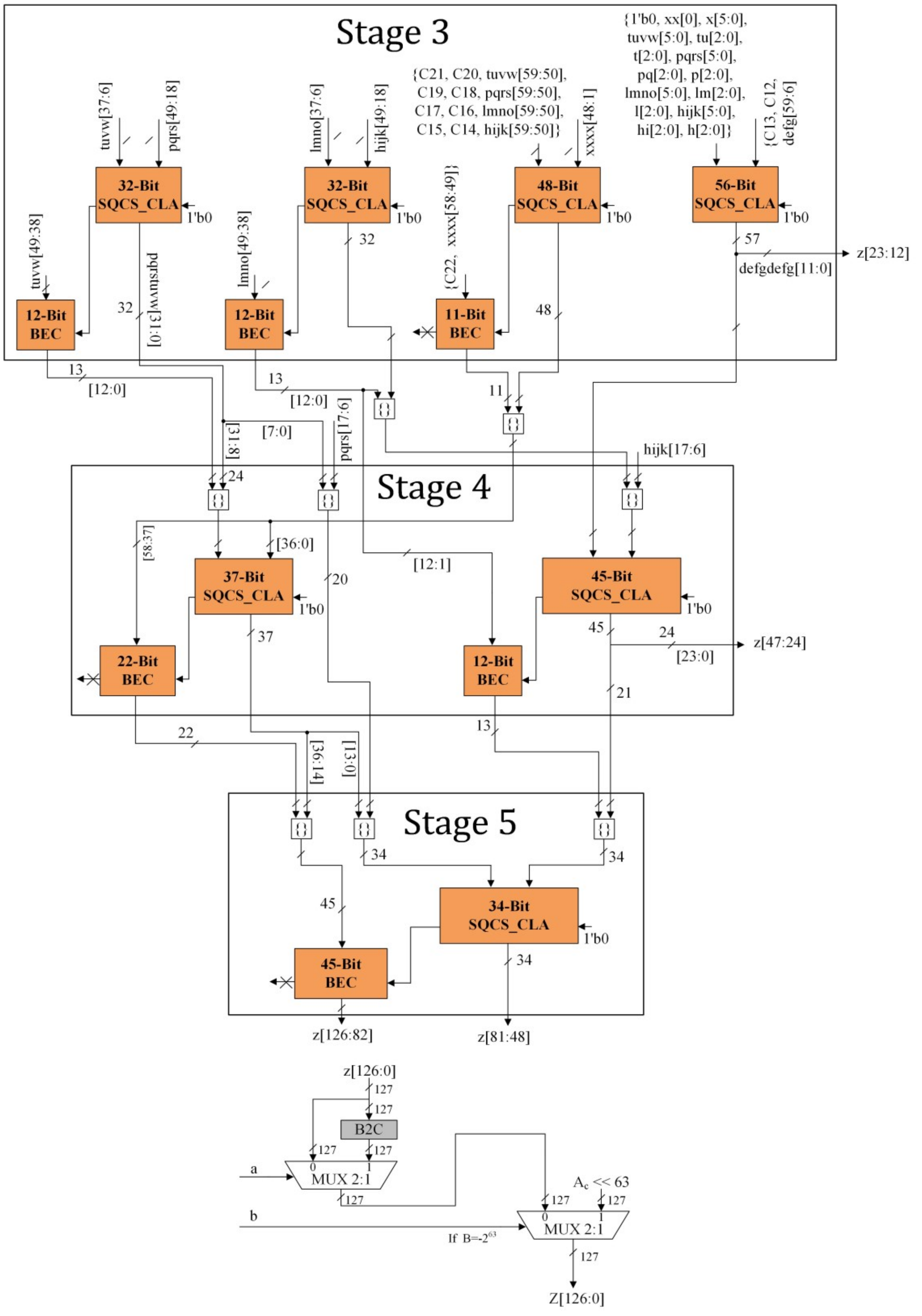


Figure 38 Stages 3, 4, and 5 of the proposed 64 x 64 signed multiplier with radix-8 architecture

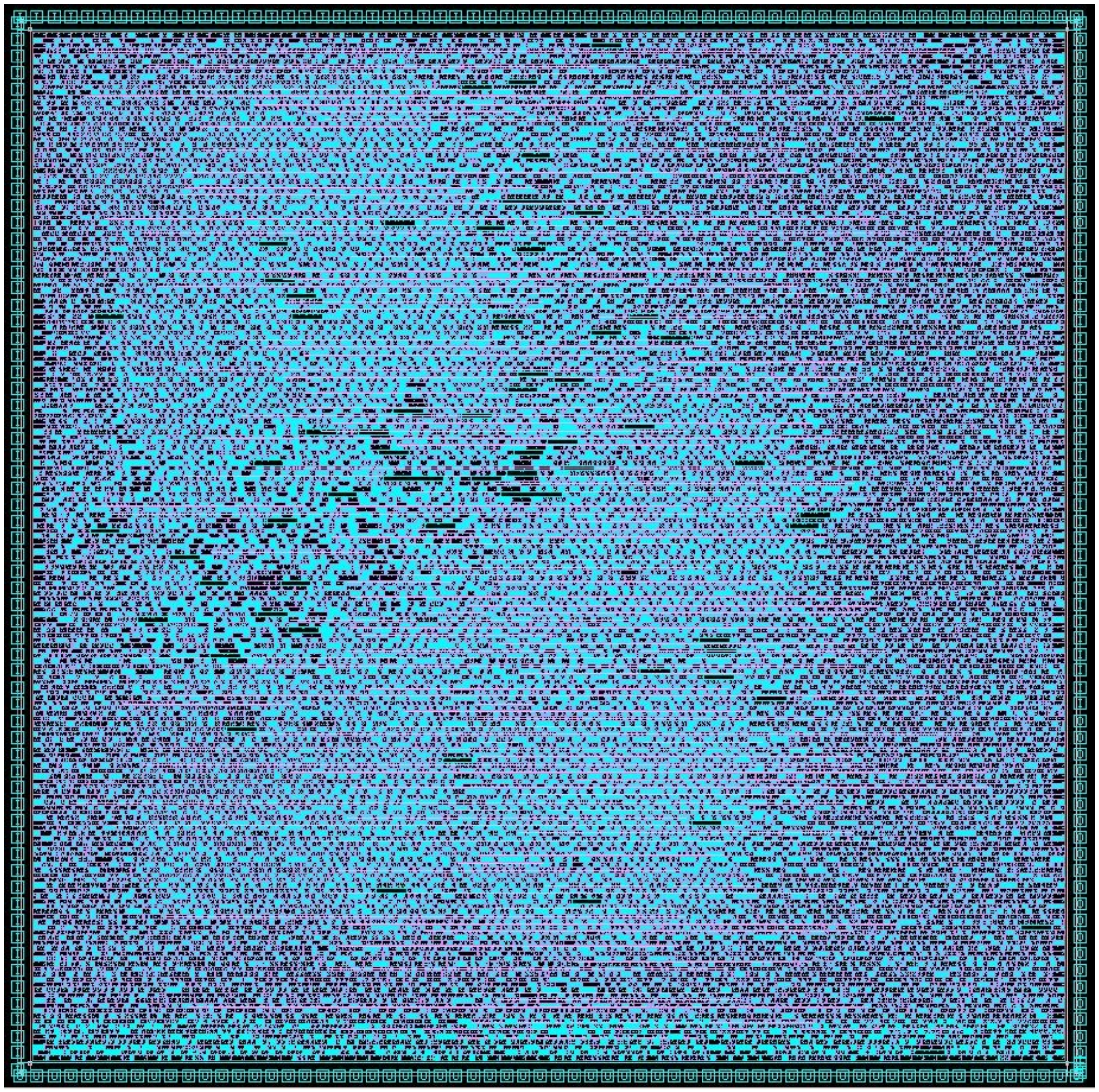


Figure 39 Layout of the proposed 64 x 64 multiplier

Layout of the proposed 32 x 32 signed multiplier with a simple and new radix-8 structure for partial product generation and new grouping strategy for the partial product reduction, which is synthesized at 200 MHz clock frequency and performed PnR, is shown in Figure 39. Total chip area of the layout shown in the above figure is

108593.952 (μm)^2 with the standard cell utilization factor of 61.46%. Hence, the total cell area is nearly 66743 (μm)^2 with the total number of standard cells used as 22194.

4.5 Results & Performance Comparison

The proposed multiplier design and the optimized Booth algorithms presented in [41] and [43] [42] are synthesized using Synopsys design compiler (DC). Regular voltage threshold (RVT) cells, for which the voltage threshold is in between high voltage threshold (HVT) cells and low voltage threshold (LVT) cells, at typical-typical corner are used to design, synthesize, and perform the placement and routing (PnR). The total power consumption and delay are measured with the supply voltage of 1.05 V. All the delay values reported are either from Synopsys Design Compiler (DC) generated timing reports or from IC Compiler (ICC) generated timing reports. But, the delay values tabled and discussed in this work are not from the Prime-Time tool generated timing reports. Prime-Time (PT) tool is a powerful and the industry gold-standard static timing analysis (STA) tool that provides a single, golden, trusted signoff solution for timing, signal integrity, power, and variation-aware analysis [47]. Synopsys DC tool generated results correlate within 10% of physical implementation [48]. Path delay reported from PT is based on both Cell delay and net delay. PT can be used pre-PnR stage to confirm the design achieves the timing goal and post-PnR stage for the post-layout timing signoff. All the synthesized binary two’s complement (B2C) designs include input and output registers, but the area and power values, shown in the following Table 4.3, are only for the combinational part of the design. Area includes the combinational design area, buffer/inverter area, and the net interconnect area. But, the current B2C design used does not include any inverters or buffers and hence the buffer/inverter area for B2C designs of various sizes are 0. Total delay includes the delay from the input registers and the combinational delay. The delay is the data arrival time (DAT) from the output of the input registers to the input of the output registers. Addition of the input and the output registers for the synthesis purpose results in an increase in the combinational circuit delay. Since the 8-bit proposed multiplier structure requires deployment of 8-bit and 15-bit B2C and the 16-bit multiplier uses 16-bit and 31-bit B2C, synthesis results for all four designs, which are synthesized at 500MHz clock frequency, are included in Table 4.3.

Table 4.3 Synthesis (pre-layout) results for 8-bit and 16-bit B2C in [41]

B2C Design	Frequency (MHz)	Area $(\mu m)^2$	Power (μW)	Delay (ns)
8-bit (Boppana et al. [41])	500	50	5.4	0.47
15-bit (Boppana et al. [41])	500	117	9.6	0.68
16-bit (Boppana et al. [41])	500	127	10.3	0.73
31-bit (Boppana et al. [41])	500	273	20.5	1.32
32-bit (Boppana et al. [41])	500/250	275	20.6/18.0	1.32
63-bit (Boppana et al. [41])	250	576	37.5	2.53
64-bit (Boppana et al. [41])	250	586	38.1	2.59
127-bit (Boppana et al. [41])	250	1300	96.8	3.97

Proposed multiplier of 32 x 32 size uses two B2C blocks of size 32-bit, here N = 32, at the precomputation stage and one 63-bit B2C, of width 2*N – 1, is used at the final stage to perform the magnitude conversion of the magnitude of the product obtained by performing the partial product reduction. Similarly, proposed multiplier of size 64 x 64, here N = 64, uses two B2C blocks of size 64-bits at the precomputation stage and one

B2C block of size 123-bits at the final stage. B2C blocks used in designing of 32-bit and 64-bit multipliers are synthesized at 250MHz since the designs are wider with longer data arrival time. A major observation pertaining to the delay of the proposed multiplier designs can be made that the B2C blocks cause longer processing delays. This leads to the room to perform more research towards finding faster B2C designs. Also, area, power, and delay are lesser compared to the published values for the 8-bit B2C in [41], shown in Figure 40, since the 9th output bit is not required for the proposed design in this article due to the reason mentioned above.

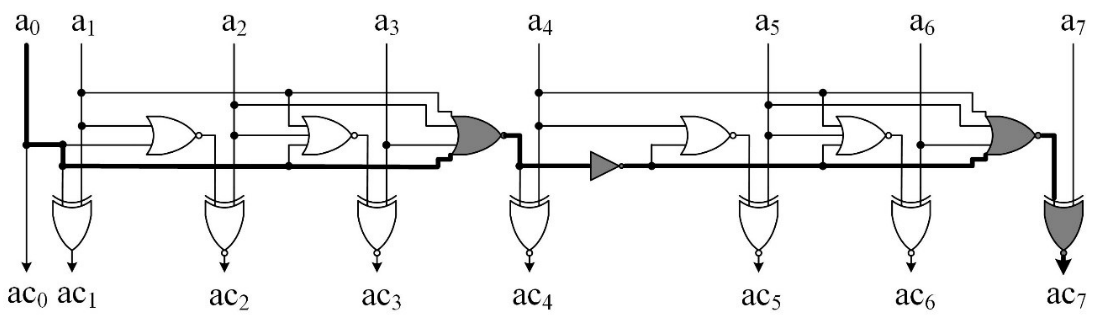


Figure 40 Optimized 8-bit B2C in [41] used in the 8x8 multiplier proposed in this work

The critical path delay for the B2C used and shown in Figure 40 is formulated as follows:

$$CPD = floor\left(\frac{N-2}{3}\right) \times NOR4's + floor\left(\frac{N-3}{3}\right) \times INV's + NOR\#\left[\left((N-2) \bmod 3\right) + int\left(bool\left(\sim\left((N-2) \bmod 3\right) == 0\right)\right)\right] + XOR2$$

(4.1)

The synthesis results for the rest of the sub-components used in the proposed multipliers of various sizes, 8-bit, 16-bit, 32-bit, and 64-bit, with radix-8 structure are shown in the Table 4.4. All the inputs coming to each of the combinational logic blocks comes through the registers and similarly, all the outputs of the combinational blocks are connected to registers. Area and power are measured only for the combinational part of the design like the measurements done for the B2C design. And, the delay measured and included in the table is the data arrival time.

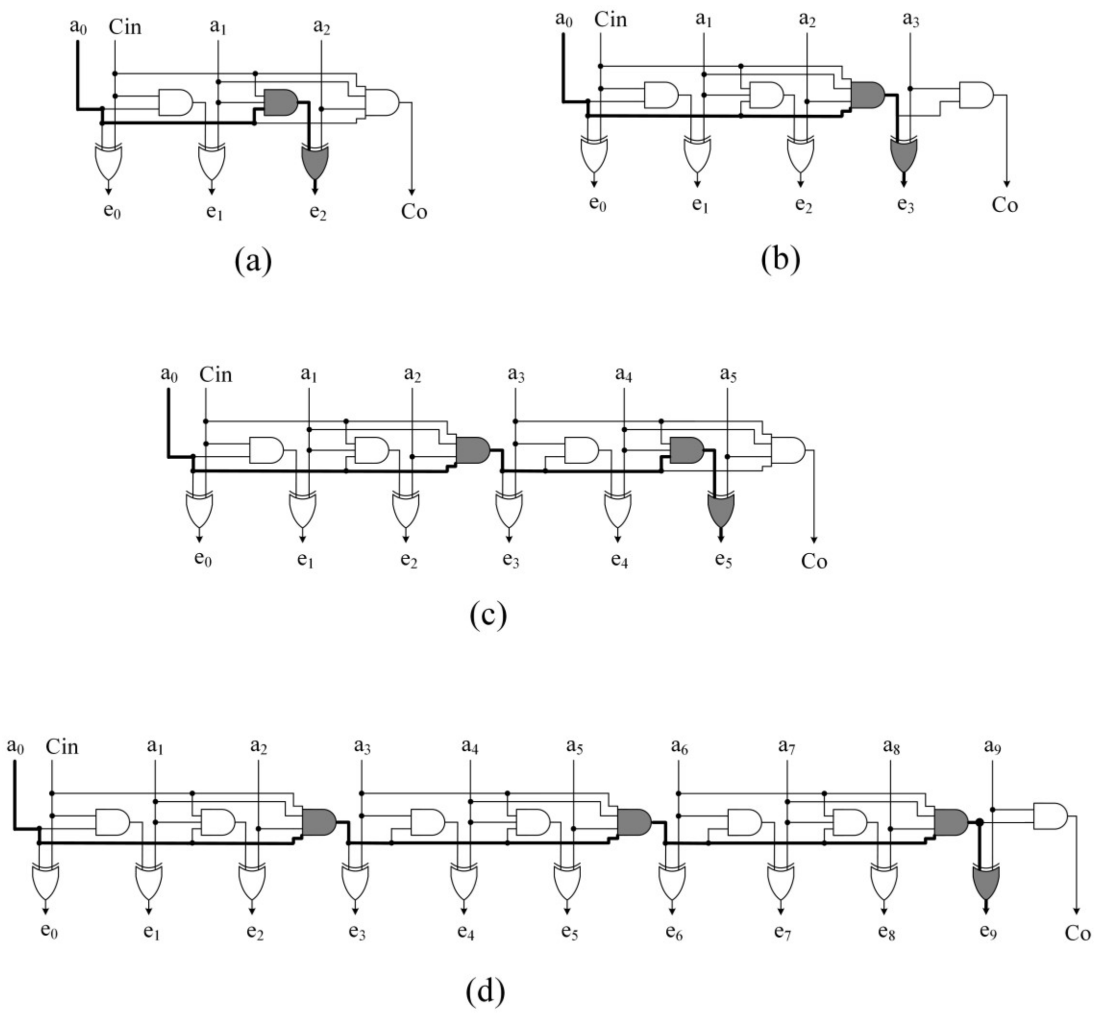


Figure 41 BEC like structure with widths (a) 3-bit, (b) 4-bit, (c) 6-bit, and (d) 10-bit

BEC like structure used has the similar structure like B2C block. B2C involves bit wise inversion and excesss-1 calculation. The actual BEC (Binary Excess-1 Converter) simply adds 1 to the input data word of respective size used. Whereas, BEC like structures used in all four different size multipliers compute the excess 1, i.e., addition of 1, if Cin = 1, otherwise results in exact input as output without any addition of 1. N-bit wide structure yields a N+1 result, i.e., Co = 1, if and only if all N-bits are binary 1. For instance, if the 3-bit BEC like structure is considered with inputs as '111' and Cin=1, then only the output is 4-bit, '1000', i.e., Co = 1 and $\{e_2e_1e_0\} = \{000\}$. BEC like structure is very simple to design, occupies less area, consumes less power, and performs computation quickly. Each bit of BEC like block requires an average design area of an AND gate plus a 2-input XOR gate. Some of the BEC like structures of 3-bit, 4-bit, 6-bit, and 10-bit wide are shown in Figure 41 (a), (b), (c), and (d) respectively. The usage of BEC like structures has been increased with the increase in the width of the multiplier. 3-bit wide BEC like structure is used in the proposed 8 x 8 multiplier, 3-bit, 6-bit, and 11-bit wide BEC like structures is used in the proposed 16 x 16 multiplier, 4-bit, 6-bit, 10-bit, and 11-bit wide structures used in the proposed 32 x 32 multiplier, and 6-bit, 11-bit, 12-bit, 22-bit, and 45-bit wide BEC like structures used in the proposed 64 x 64 multiplier. To make the fair comparison between BEC structures of various size, all the designs are synthesized at 500MHz. Simple and light weight BEC like designs and SQCS adder with CLA blocks of similar width have the similar delays. The delays of the BEC like structures and SQCS adders of sizes 12-bit, 22-bit, and 45-bit are {0.62, 0.72}ns, {1.04, 1.06}ns, and {1.96, 1.92}ns respectively. The delay of the BEC like block is nearly proportional to the number of 3-bit chains times the added delay by the 4-input AND gate plus the 2-input XOR gate. The formulation required to calculate or estimate the exact critical path delay (CPD) of the BEC like structure is shown in the following equation consisting of $floor\left(\frac{N-1}{3}\right)$ times the 4-input AND gate delays plus one $\left[\left((N-1)\ mod\ 3\right)+int\left(bool\left(\sim\left((N-1)\ mod\ 3\right)==0\right)\right)\right]$ -input AND gate delay, and one two input XOR gate delay.

$$CPD = floor\left(\frac{N-1}{3}\right) \times AND4's + AND\#\left[\left((N-1)\ mod\ 3\right)+int\left(bool\left(\sim\left((N-1)\ mod\ 3\right)==0\right)\right)\right] + XOR2 \quad (4.2)$$

Table 4.4 Synthesis (pre-layout) results for the sub-components used in the 8 x 8, 16 x 16, 32 x 32, and 64 x 64 signed multiplier designs proposed

Design Name	Frequency (MHz)	Area $(\mu m)^2$	Power (μW)	Delay (ns)
3-bit BEC	500	22	2.3	0.30
4-bit BEC	500	27	2.8	0.32
6-bit BEC	500	44	4.1	0.42
10-bit BEC	500	83	5.9	0.52
11-bit BEC	500	95	6.5	0.57
12-bit BEC	500	104	7.0	0.62
22-bit BEC	500	193	12.6	1.04
45-bit BEC	500	406	24.7	1.96

7-bit SQCS-CLA (34)	500	230	28	0.57
8-bit SQCS-CLA (44)	500	262	32	0.63
9-bit SQCS-CLA (234)	500	299	36	0.61
12-bit SQCS-CLA (2334)	500	404	48	0.72
15-bit SQCS-CLA (22344)	500	504	60	0.83
22-bit SQCS-CLA (22344)	500	747	88.8	1.06
27-bit SQCS-CLA (222333444)	500	923	109	1.27
30-bit SQCS-CLA (2223333444)	500	1030	121	1.37
32-bit SQCS-CLA (22-bit & 10-bit SQCS-CLA)	500/250	1095	129.1/95.3	1.47
34-bit SQCS-CLA (27-bit & 7-bit SQCS-CLA)	500/250	1163	137.7/101.4	1.52
37-bit SQCS-CLA (30-bit & 7-bit SQCS-CLA)	500/250	1270	150/110.4	1.62
45-bit SQCS-CLA (30-bit & 15-bit SQCS-CLA)	500/250	1547	182/134.2	1.92
48-bit SQCS-CLA (30-bit, 10-bit, & 8-bit SQCS-CLA)	250	1650	143.1	2.02
56-bit SQCS-CLA (27-bit, 22-bit, & 7-bit SQCS-CLA)	250	1919	167.1	2.28
61-bit SQCS-CLA (30-bit, 22-bit, & 9-bit SQCS-CLA)	250	2098	181.8	2.46
63-bit SQCS-CLA (30-bit, 27-bit,	250	2172	187.9	2.59

& 6-bit SQCS-CLA)				
8-bit NT Block	500	309	36	0.71
16-bit NT Block	500	655	80	1.20
32-bit NT Block	250	1348	120.9	2.17
64-bit NT Block	250	2946	273.5	3.97

Coming to the comparison of the design area required and power consumption of 12-bit, 22-bit, and 45-bit wide SQCS adder and BEC like structure, SQCS adder requires approximately 3.7 times the area and utilizes approximately 7 times the power compared with BEC like structure. Square root carry select (SQCS) adders with carry-look-ahead (CLA) blocks of various sizes are used along with the BEC like blocks to reduce the partial products. SQCS with CLA blocks are also used and the only computation used to generate the P, Q, and R outputs of the non-trivial blocks. The SQCS with CLA blocks used in NT blocks are not exactly of the square-root carry select structure. These blocks follow the square root carry select structure for the initial few blocks at the LSB side and the later follows the linear carry select structure. Hence, the adders used in the NT blocks can be labeled as L/SQCS_CLA. This leads to more room for finding and deploying a faster adder to reduce the delay.

An important observation can be made based on the processing delays of the B2C blocks, delays of the non-trivial blocks, and the delays of the complete 8-bit, 16-bit, 32-bit, and 64-bit multiplier designs, included in Table 4.5, that the B2C and NT blocks constitute towards the major portion of the total delay of the proposed multiplier designs. The following table, Table 4.5, include the synthesis and PnR results of the 8-bit and 16-

bit multipliers at 250 MHz clock frequency. The delays of the 8-bit and 16-bit multipliers are {2.17, 1.77} ns and {3.4, 2.74} ns respectively for the {synthesized, PnR} designs. Similarly, the delays measured for the 32-bit and 64-bit multipliers synthesized, at clock frequencies of 200 MHz and 100 MHz respectively, are {4.97, 4.44} and {9.98, 7.81} respectively for the {synthesized, PnR} designs. In case of synthesized designs, the delays are increasing at a rate of around 1.5x moving from 8-bit to 16-bit and 16-bit to 32-bit, whereas, rate is changed to nearly 2x moving from 32-bit to 64-bit. Similarly, the rate at which the delays increase by for the placement and routed designs moving from 8-bit to 16-bit to 32-bit is nearly 1.5x but this rate is changed to 1.75x moving from 32-bit to 64-bit design. This increase in rate of change of delay is clearly due to the use of mostly linear style carry select adders in NT blocks and in stages involving partial products. The area column of the table includes the non-combinational area (NCA) by the registers, combinational area (CA), and the total area (TA) which is the sum of NCA, CA, buffer/inverter area, macro/black box area if any, and the net interconnect area for the proposed design. Total cell area (TCA) includes CA and NCA. The power column in the table consists of two groups of power: one includes the power consumption by the registers and the second of the power consumption by the combinational circuit. The total power consumption by the each of the power group objects consists of internal power (IP), switching power (SP), leakage power (LP), and the total power (TP) which is the sum of IP, SP, and LP. Total power section includes the power consumption by the registers and the combinational part of the design. In general, if the multiplier size doubles then the size of the array formed by the partial products doubles vertically and horizontally. Hence, the area increases by four times and the delay increases by approximately two times. But, whereas in the case of the proposed design, the delay and the total area are increased by approximately 1.5 times and 3 times respectively when moving from the 8-bit proposed multiplier to the 16-bit proposed multiplier. With the proposed architecture, the critical path delay (CPD), total power consumption, and total area are increased by 1.5 to 2 times, 3 to 4 times, and 3 to 4 times respectively while comparing with the designed multiplier of twice the size.

Table 4.5 Synthesis and post-layout results for the proposed design

Radix-8 (3-bit grouping)						
# Multiplier Size	Synthesis			PnR		
	Area $(\mu m)^2$	Delay (DAT) (ns)	Power (μW)	Area $(\mu m)^2$	Delay (DAT) (ns)	Power (μW)
8 (1133) (2 stages)	Reg: 205(NCA)	**2.17**	**@4n clk** Reg: 50(IP) 0.5(SP) 35(LP) ------ 85(TP)	Reg: 205(NCA)	**1.77**	**@4n clk** Reg: 54(IP) 1.3(SP) 35(LP) ------ 90(TP)
	Total: 1327(CA) **1815**(TA)		Total: 110(IP) 28(SP) 133(LP) ------ **271**	Total: 1337(CA) **1840**(TA)		Total: 115(IP) 43(SP) 140(LP) ------ **298**
16 (133333) (3 stages)	Reg: 417(NCA)	**3.4**	**@4n clk** Reg: 101(IP) 1.5(SP) 71(LP) ------ 174(TP)	Reg: 417(NCA)	**2.74**	**@4n clk** Reg: 112(IP) 3.2(SP) 71(LP) ------ 186(TP)
	Total: 4150(CA) **5725**(TA)		Total: 297(IP) 82(SP) 367(LP) ------	Total: 4224(CA) **5777**(TA)		Total: 315(IP) 148(SP) 418(LP) ------

			746(TP)			**881**(TP)

32
(11333333333
3)
(4 stages)
	Reg:
	840(NCA)
		4.97	**@5n clk**
			Reg:
			161(IP)
			1.9(SP)
			142(LP)
			------ 	
			305(TP)
				Reg:
				841(NCA)
					4.44	**@5n clk**
						Reg:
						181(IP)
						4.7(SP)
						143(LP)
						------ 	
						328(TP)

	Total:
	16881(CA)
	22776(TA)
			Total:
			892(IP)
			309(SP)
			1446(LP)
			------ 	
			2647(TP)
				Total:
				16899(CA)
				25106(TA)
						Total:
						939(IP)
						552(SP)
						1704(LP)
						------ 	
						3194(TP)

64
(13333333333
33333333333)
(5 stages)
	Reg:
	1685(NCA)
		9.98	**@10n clk**
			Reg:
			161(IP)
			1.8(SP)
			285(LP)
			------ 	
			447(TP)
				Reg:
				1686(NCA)
					7.81	**@10n clk**
						Reg:
						187(IP)
						7.5(SP)
						285(LP)
						------ 	
						479(TP)

	Total:
	63548(CA)
	85708(TA)
			Total:
			1628(IP)
			625(SP)
			5224(LP)
			------ 	
			7478(TP)
				Total:
				66743(CA)
				100764(TA)
						Total:
						1789(IP)
						1364(SP)
						7016(LP)
						------ 	
						10169(TP)

NCA - Non-Combinational Area; CA – Combinational Area; TA – Total Area; TCA – Total Cell Area; IP – Internal Power; SP – Switching Power; LP – Leakage Power; TP – Total Power; TDP – Total Dynamic Power; CLP – Cell Leakage Power;

Similarly, synthesis and PnR are performed on the proposed designs of sizes, 8-, 16-, 32-, and 64-bit, synthesized at highest clock frequency and the respective pre- and post-layout reports including the area, delay, and power are included in the following table, Table 4.6. The highest clock frequencies at which the designs, of widths 8-, 16-, 32-, and 64-bit, synthesized are 476 MHz, 333 MHz, 207 MHz, and 104 MHz respectively, resulting in post-layout delays of 1.69 ns, 2.49 ns, 4.22 ns, and 6.95 ns respectively, with the total area and power consumption of {1875 $(\mu\text{m})^2$, 0.46 mW}, {5973 $(\mu m)^2$, 1.09 mW}, {25107 $(\mu m)^2$, 3.26 mW}, and {100993 $(\mu m)^2$, 10.45 mW} respectively for the post-layouts.

Table 4.6 Synthesis and post-layout results for the proposed designs synthesized at highest frequency

Radix-8 (3-bit grouping)						
# Multiplier Size	Synthesis			PnR		
	Area $(\mu m)^2$	Delay (DAT) (ns)	Power (μW)	Area $(\mu m)^2$	Delay (DAT) (ns)	Power (μW)
8 (1133) (2 stages)	Reg: 205(NCA)	**2.07**	**@2.1n clk** Reg: 96(IP) 1(SP) 35(LP) ------ 131(TP)	Reg: (NCA) 205	**1.69**	**@2.1n clk** Reg: 104(IP) 2.5(SP) 35(LP) ------ 141(TP)
	Total: 1368(CA) 1573(TCA) **1860(TA)**		Total: 215(IP) 55(SP) 142(LP) ------ **413(TP)**	Total: 1371(CA) 1576(TCA) **1875(TA)**		Total: 224(IP) 84(SP) 147(LP) ------ **455(TP)**
16 (133333) (3 stages)	Reg: 417(NCA)	**2.98**	**@3n clk** Reg: 135(IP) 2(SP) 71(LP) ------ 208(TP)	Reg: 417(NCA)	**2.49**	**@3n clk** Reg: 151(IP) 5(SP) 71(LP) ------ 226(TP)
	Total: 4376(CA) 4793(TCA) **5976(TA)**		Total: 410(IP) 116(SP) 401(LP) ------ **928(TP)**	Total: 4400(CA) 4818(TCA) **5973(TA)**		Total: 435(IP) 214(SP) 440(LP) ------ **1089(TP)**

32
(113333333333)
(4 stages)
	Reg:
	840(NCA)
		4.79	**@4.82n clk**
			Reg:
			167(IP)
			2(SP)
			142(LP)
			------ 	
			311(TP)
				Reg:
				841(NCA)
					4.22	**@4.82n clk**
						Reg:
						189(IP)
						6(SP)
						143(LP)
						------ 	
						338(TP)

	Total:
	16988(CA)
	17827(TCA)
	22905(TA)
			Total:
			926(IP)
			322(SP)
			1457(LP)
			------ 	
			2704(TP)
				Total:
				16954(CA)
				17794(TCA)
				25107(TA)
						Total:
						976(IP)
						576(SP)
						1708(LP)
						------ 	
						3260(TP)

64
(13333333333
33333333333)
(5 stages)
	Reg:
	1686(NCA)
		9.57	**@9.6n clk**
			Reg:
			168(IP)
			2(SP)
			285(LP)
			------ 	
			454(TP)
				Reg:
				1688(NCA)
					6.95	**@9.6n clk**
						Reg:
						195(IP)
						8(SP)
						286(LP)
						------ 	
						488(TP)

	Total:
	63849(CA)
	65535(TCA)
	85771(TA)
			Total:
			1705(IP)
			656(SP)
			5264(LP)
			------ 	
			7625(TP)
				Total:
				65356(CA)
				67044(TCA)
				100993(TA)
						Total:
						1877(IP)
						1469(SP)
						7100(LP)
						------ 	
						10447(TP)

A fact that the cell leakage power nearly equals to the total dynamic power for the advanced nodes can be observed based on the TDP and CLP values included in the following Table 4.7 for various sizes of the proposed design. In case of 32-bit design, the CLP is greater than the TDP while the CLP is more than twice the TDP value for the 64-bit proposed design. This is because of the inevitable scenario of smaller size nodes prone to more leakages.

Table 4.7 Comparison of Total Dynamic Power (TDP) and Cell Leakage Power (CLP) for the proposed synthesized and post-layout designs of various sizes

Design	Synthesis		PnR	
	TDP (mW)	CLP (mW)	TDP (mW)	CLP (mW)
8 @ 4ns	0.1380559	0.1330748	0.1581373	0.1399205
16 @ 4ns	0.3793445	0.3670299	0.4632997	0.4181780
32 @ 5ns	1.2016	1.4456	1.4909	1.7040
64 @ 10ns	2.2534	5.2243	3.1530	7.0163

Delay, area, and power consumption measured for the conventional Booth multiplier synthesized at the respective highest frequency, included in the following Table 4.8, for the multipliers are increasing in factors of nearly 2x, 4x, and 3x respectively while doubling the operand sizes. There is an improvement, reduction in this scenario, of delay, area, and power consumption by a factor of nearly 1.5x to 2x when comparing the results obtained from the synthesized conventional Booth multiplier of various sizes, included in the Table 4.8, with the results obtained from the synthesized modified Booth multiplier presented or synthesized simple radix-8 design proposed of various sizes presented in this work.

Comparison of the digital design performance characteristics for the 8-bit design in [46] and the modified Booth multiplier design in [41], both synthesized at 500MHz, results in an observation of reduction in delay, area, power, ADP, PDP, and EDP by 6.33%, -0.32%, 22.02%, 6.03%, 26.95%, and 31.58% respectively. The performance characteristics compared in the same order for the synthesized 8-bit design in [46] with the post-layout 8-bit design in [41], the performance improvements are more significant with 24.68%, 5.46%, 21.56%, 28.8%, 40.92%, and 55.5% respectively. Also, the data obtained from [22] for the signed radix-2^m parallel multipliers of sizes, 8-, 16-, 32-, and 64-bit with four proposed architectures based on optimized Wallace and Dadda tree partial product compression strategies are presented in the following table for comparison.

Optimized Booth multiplier implementation in [41] employs optimized partial product generation and parallel addition of the partial products. Addition is performed in the binary tree style. The number of partial products to be summed following the architecture in [41] are same as in the case of conventional Booth multiplier and the numbers are 4, 8, 16, and 32 respectively for the 8-, 16-, 32-, and 64-bit multipliers with two times the change in the width of the partial products from increase in the multiplier size by two times. Performance comparison between the optimized Booth multiplier presented in this work [41], as a first contribution, and the low PDP Booth multiplier presented in [42] [43], both designs synthesized at 500MHz, leads to an observation of reduction in delay, area, power, ADP, PDP, and EDP by 25.63%, 15.03%, 26.57%, 36.79%, 45.39%, and 59.36% respectively. Comparing the design area required for the optimized Booth multiplier structure in [41] and the signed multiplication included in this work, the design space requirement quadruple using the structure in [41] since the number of partial products and the width of each partial product doubles which doubles the number of adders and the size of each adder when the multiplier input size doubles. Whereas in case of the proposed work with radix-8 structure, the effective increase in the design area is only three times because of the reduction in the number of partial products and the separating out of some group of bits and performing simple additions along with the simple BEC like structures. The approximate increase in the delay of the multiplier structure in [41] is two times with the increase in multiplier size by two times because of the increase in number of partial products by two times and the increase in the size of the additions at each stage. Partial products generated using the architecture in [41] are {4, 8, 16, 32} and using the proposed architecture are {3, 5, 11, 21} for the 8-, 16-, 32-, and 64-bit multipliers respectively. Designing wider multipliers using higher radix structures further reduces the number of partial products and hence the delay and the cost of computation. All the results including the delay, area, and power values for the 8-bit, 16-bit, 32-bit, and 64-bit multipliers, synthesized and post-layout design completed, with the proposed architecture and the modified Booth architecture, are included in the Table 4.8. Projected performance and cost numbers for the proposed 32-bit and 64-bit multiplier architectures shows significant improvement compared to the conventional Booth multiplier design of 16-bit and 32-bit wide. 16 x 16 multiplier using the proposed radix-8 architecture outperformed the parallel 16 x 16 multiplier with radix-10 BCD multiplier presented in [49] when the total power consumptions are compared, even with the non-combinational power consumption due to the registers at the inputs and outputs included. If the frequency is also considered for scaling down the power consumption for the design in [49] to make more appropriate comparison to be performed at the respective highest frequencies, the power consumption, PDP, and EDP values for the 16-bit multiplier with the proposed radix-8 architecture are very small in comparison.

Table 4.8 Performance comparison of the multipliers

Multiplier	Width (N x N) (N-bits)	Tech.	Freq. (MHz) / Period (ns)	Supply Voltage (V)	Data Arrival Time (ns)	Area $(\mu m)^2$	Total Power (mW)	ADP (m^2-s) x 10^{-21}	PDP (E/op) (W-s) $\left(\frac{pJ}{op}\right)$	EDP (J-s) x 10^{-21}
	64	32nm (Post-)	f=100		7.81	100764[a]	10.169[b]	786967	79.42	620.269
			$f_{highest}$ = 104.2		6.95	100993[a]	10.447[b]	701901	72.61	504.616
		32nm	f=100		9.98	85708[a]	7.478[b]	855366	74.63	744.818

(This work) 2022		(Pre-)	$f_{highest}$ = 104.2		9.57	85771[a]	7.625[b]	820828	72.97	698.335
	32	32nm (Post-)	f=200		4.44	25106[a]	3.194[b]	111471	14.181	62.965
			$f_{highest}$ = 207.5		4.22	25107[a]	3.26[b]	105952	13.757	58.055
		32nm (Pre-)	f=200		4.97	22776[a]	2.647[b]	113197	13.156	66.050
			$f_{highest}$ = 207.5		4.79	22905[a]	2.704[b]	109715	12.952	62.041
	16	32nm (Post-)	f=250		2.74	5777[a]	0.881[b]	15829	2.388	6.614
			$f_{highest}$ = 333.3		2.49	5973[a]	1.089[b]	14880	2.712	6.752
		32nm (Pre-)	f=250		3.4	5725[a]	0.746[b]	19465	2.536	8.624
			$f_{highest}$ = 333.3		2.98	5976[a]	0.928[b]	17808	2.765	8.241
	8	32nm (Post-)	f=250		1.77	1840[a]	0.298[b]	3257	0.527	0.934
			$f_{highest}$ = 476.2		1.69	1875[a]	0.455[b]	3169	0.769	1.300
		32nm (Pre-)	f=250		2.17	1815[a]	0.271[b]	3939	0.588	1.276
			$f_{highest}$ = 476.2		2.07	1860[a]	0.413[b]	3850	0.854	1.767
Estimated (Boppana et al. [41])	64	32nm (Post-)	$f_{highest}$ = ~ 117.5	-	~ 8.48 (2x 32b)	~ 95040 (4x 32b)	-	~ 805939	-	-
		32nm (Pre-)	$f_{highest}$ = ~ 101.3	-	~ 9.84 (2x 32b)	~ 101504 (4x 32b)	-	~ 998799	-	-
	32	32nm (Post-)	$f_{highest}$ = ~ 234.2	-	~ 4.24 (2x 16b)	~ 23760 (4x 16b)	-	~ 100742	-	-
		32nm (Pre-)	$f_{highest}$ = ~ 202.0	-	~ 4.92 (2x 16b)	~ 25376 (4x 16b)	-	~ 124850	-	-
	16	32nm (Post-)	$f_{highest}$ = ~ 465.1	-	~ 2.12 (2x 8b)	~ 5940 (4x 8b)	-	~ 12593	-	-
		32nm (Pre-)	$f_{highest}$ = ~ 401.6	-	~ 2.46 (2x 8b)	~ 6344 (4x 8b)	-	~ 15606	-	-
(Boppana et al. [41]) (This work) 2019	8	32nm (Post-)	f=500	1.05	1.19	1177[a]	0.342[b]	1401	0.407	0.484
			$f_{highest}$ = 793.6		1.06	1485[a]	0.526[b]	1574	0.558	0.591
		32nm (Pre-)	f=500		1.48	1249[a]	0.340[b]	1849	0.503	0.745
			$f_{highest}$ = 793.6		1.23	1586[a]	0.509[b]	1951	0.625	0.769
[22] NR Wallace [c] 2020	64	65nm (Pre-)	57.1	1.0	17.5	78194.5	3.3539	1368404	65.6	1148.00
	32		109.9		9.1	23562.8	1.9386	214422	17.6	160.16
	16		196.2		5.1	6716.3	1.0661	34253	5.4	27.54
	8		344.9		2.9	2075.8	0.6458	6020	1.9	5.51
[22] NR Dadda [c] 2020	64	65nm (Pre-)	57.8	1.0	17.3	75280.9	3.3762	1302360	58.4	1010.32
	32		111.1		9	20803.6	1.8313	187232	16.5	148.50
	16		208.3		4.8	6627.9	1.1083	31814	5.3	25.44
	8		357.1		2.8	2033.7	0.6746	5694	1.9	5.32
[22] NR-SO Wallace [c] 2020	64	65nm (Pre-)	57.8	1.0	17.3	69218.2	2.9294	1197475	50.7	877.11
	32		109.9		9.1	21002.8	1.7806	191126	16.2	147.42
	16		196.0		5.1	6113.1	0.9968	31177	5.1	26.01
	8		344.8		2.9	1947.9	0.6384	5649	1.8	5.22
[22] NR-SO	64		57.8		17.3	65187.2	2.8878	1127739	50.0	865.00
	32	65nm	111.1	1.0	9	18092.4	1.6836	162832	15.2	136.80

Dadda [c] 2020	16	(Pre-)	208.3		4.8	5975.8	1.0821	28684	5.2	24.96
	8		357.1		2.8	1878.2	0.6796	5259	1.9	5.32
[22] Baseline [c] (Radix-4)	64	65nm (Pre-)	38.3	1.0	26.1	128500.8	8.8409	3353871	230.8	6023.88
	32		76.9		13	32765.2	2.5130	425948	32.7	425.10
	16		149.3		6.7	8006.4	0.9146	53643	6.1	40.87
	8		302.8		3.3	2333.8	0.9277	7702	3.1	10.23
Gorgin et al. [49] 2017	16	TSMC 130nm (Pre-)	min. power	1.5	2.462 (10)[c]	-	10.9 (90)[c]	-	26.836	66.07
			min. delay		1.083 (4.4)[c]	-	28.9 (240)[c]	-	31.299	33.896
Qian et al. [46] 2016	8	45nm (Pre-)	f=500	1.25	1.58	1245	0.436	1967	0.689	1.088
Modified Booth [d] [43] [42] 2017-18	8	32nm (Pre-)	f=500	1.05	1.99	1470[a]	0.463[b]	2925	0.921	1.833
Conv. Booth Multiplier	32	32nm (Pre-)	f=125	1.05	7.97	38895[a]	3.611[b]	309993	28.78	229.37
			$f_{highest}$ = 133.3		7.47	39768[a]	3.884[b]	297067	29.013	216.73
	16		f=250		3.97	9757[a]	1.223[b]	38735	4.855	19.275
			$f_{highest}$ = 263.2		3.78	9851 [a]	1.289[b]	37237	4.872	18.418
	8		400		2.28	2315[a]	0.380[b]	5278	0.866	1.975
			$f_{highest}$ = 476.2		2.07	2398[a]	0.453[b]	4964	0.938	1.941

[a] Area includes the area of the input and output registers
[b] Power includes the power consumption by the input and output registers
[c] Actual values published
[d] Performance data obtained from redesigning & synthesizing the design from the original work
NOTE: Area, delay, and power are normalized to 32nm @ 1.05V unless specified
NOTE: Pre- and Post- refers to Synthesis (Pre-layout) and PnR (Post-layout) results respectively

The results published in [46], [49] are normalized to 32nm CMOS technology with 1.05V supply voltage, included in Table 4.8, using the following normalization equations [50] to make fair comparisons on various performance aspects.

$$T_{d_{norm}} = T_d \,.\left(\frac{32\ nm}{tech}\right), P_{norm} = P\,.\left(\frac{32\ nm}{tech}\right)\left(\frac{1.05}{V_{tech}}\right)^2, Area_{norm} = Area\,.\left(\frac{32\ nm}{tech}\right)^2 \quad \textbf{(4.3)}$$

Going further down towards using smaller nodes, the common scaling factor, S, for scaling the parameters such as width, channel length, oxide thickness, and supply voltage does not work since different parameters scale down in different ratios. Hence, more accurate scaling should be performed using the following proportions or scaling factors [51], especially to scale the performance characteristics of digital designs towards smaller nodes, which are being mostly used in the current digital designs, to perform the fair comparison with the performance characteristics of the digital designs constructed using larger CMOS technology nodes. The following scaling factors also needed to be verified while scaling up/down the performance characteristics of the digital designs designed using smaller and advanced node technologies.

$$Area \propto W \times L$$

$$Capacitance\ (C_L) \propto WL \times C_{ox} \propto WL \times \frac{\varepsilon_{ox}}{t_{ox}}$$

$$Propagation\ delay\ (t_p) \propto \frac{C_L V_{DD}}{k(V_{DD}-V_T)^2} \propto \frac{C_{ox}WL \times V_{DD}}{\mu C_{ox}\frac{W}{L}(V_{DD}-V_T)^2} \propto \frac{L^2 \times V_{DD}}{\mu(V_{DD}-V_T)^2}$$

$$Power\ (P) = C_L {V_{DD}}^2 f \propto \frac{WL\frac{\varepsilon_{ox}}{t_{ox}}{V_{DD}}^2}{t_p} \propto \frac{WL\frac{\varepsilon_{ox}}{t_{ox}}{V_{DD}}^2}{\frac{L^2 \times V_{DD}}{\mu(V_{DD}-V_T)^2}} \propto \frac{W \times V_{DD} \times (V_{DD}-V_T)^2}{L \times t_{ox}}$$

(4.4)

Where, W represents width of a MOS transistor, L represents the length of the MOS transistor, t_{ox} represents the oxide thickness, V_{DD} represents the supply voltage, V_T represents the respective MOS threshold voltage, μ represents the mobility of the respective carriers, and ε_{ox} represents the permittivity of the gate oxide.

5 Conclusions and Future Work

5.1 Conclusion

Optimized 8 x 8 Booth multiplier architecture with optimized B2C, optimized Booth encoding, and parallel addition is presented, and the performance is compared with the state-of-the-art modified Booth multipliers. Significant improvement in speed performance is observed with less design cost and power consumption. A simple and novel way of sign number multiplication with radix-8 structure is presented by designing an overhead block containing non-trivial computation block with reduced number of partial products. The careful grouping of the partial products for the addition reduces the design area while increasing the speed. The performance measures along with the cost associated for the 32-bit and 64-bit multipliers for the state-of-the-art optimized Booth multipliers are compared with the designs using the proposed radix-8 structure. It can be concluded that the new multiplier architecture with radix-8 structure has a remarkable advantage of speed while using the less hardware for the large width sign number multiplications and designing the wide multipliers with higher radix such as radix-16, -32, -64, etc., will be very promising to perform faster and efficient operations.

5.2 Major Contributions

- Designed an optimized B2C with low-power and low-area to perform the two's complement operation. The optimized design presented in this work has nearly 10.5% and 10.1% of improvement in area and power when compared to the state-of-the-art B2C design designed and synthesized using the similar technology and supply voltage.

- Designed an optimized Booth encoder with inherent multiplexer operation for faster and low-cost partial product generation in the case of modified 8 x 8 Booth multiplier implementation.

- Developed a new grouping strategy for the reduction of the parallel products by deploying the parallel adders to reduce the number of stages of additions from three to two in the case of modified 8 x 8 Booth multiplier implementation when compared to the state-of-the-art low PDP Booth multiplier design.

- Implemented SQCS-CLA for faster partial product reduction to get quicker final product in the case of both the proposed works: 1. Modified 8 x 8 Booth multiplier design, and 2. Proposed novel radix-8 architecture for the signed number multiplication.

- Developed a simple and novel radix-8 structure, with 3-bit grouping, for the signed number multiplication. Designed 8 x 8, 16 x 16, 32 x 32, and 64 x 64 multipliers using the proposed radix-8 architecture with the non-trivial blocks and an optimized strategy of partial product reduction. Number of {partial products, additions} are reduced from {16, 15} and {32, 31} to {11, 13} and {21, 23} for the 32-bit and 64-bit multipliers using the proposed radix-8 structure when compared with the state-of-the-art low PDP Booth multiplier design.

- Performed synthesis and placement & routing (PnR) on the proposed modified 8 x 8 Booth multiplier design and the proposed radix-8 structure based 8 x 8, 16 x 16, 32 x 32, and 64 x 64 signed multiplier designs. Comparing the performance results of the 32-bit and 64-bit wide multipliers designed using the simple novel radix-8 structure with the estimated performance measurements for the state-of-the-art optimized Booth multiplier design presented in this work, (synthesize and PnR)ed, reduction in delay by (2.64%, 0.47%) and (2.74%, 18.04%) respectively, and reduction in area-delay-product by (12.12%, -5.17%) and (17.82%, 12.91%) respectively is achieved.

5.3 Future Work

- Designing more efficient grouping strategies for efficient partial product reduction.

- Designing and deploying faster and efficient large width adders inside non-trivial computation block and in the partial product reduction stages.

- Designing and deploying faster yet efficient binary 2’s complement blocks to reduce the computational latency.

- Extending the use of the proposed idea by designing the sign number multiplier with higher radix structures to further improve the speed and efficiency of the computation.

- All the designs are designed using the RVT cells at low drive strength of ‘1’ unit. Hence, there is still more room for improving the speed reducing the power by deploying the combination of high drive strength standard cells, low-VT (LVT), and high-VT (HVT) cells.

- Using Synopsys Prime Time (PT) tool, static timing analysis (STA) should be performed to verify if the timing goal set is achievable in pre-PnR stage and in post-PnR stage for accurate and industry gold-standard timing signoff.

- Need to come up with more accurate way of scaling down/up the performance characteristics, such as delay, area, and power, of the designs to make fair comparisons between various designs designed using various technologies at their respective supply voltages.

5.4 Publications

Peer-Reviewed International Archival Journal Articles

- **Boppana, N.V.V.K.**, Kommareddy, J. & Ren, S. Low-Cost and High-Performance 8×8 Booth Multiplier. *Circuits, Systems, and Signal*

Processing **38,** 4357–4368 (2019). https://doi.org/10.1007/s00034-019-01044-x

- H. Xue, R. Patel, **N. V. V. K Boppana**, S. Ren, "A low power-delay-product radix-4 8*8 Booth multiplier in CMOS", IET Electronics Letters, Vol. 54, Issue 6, p. 344-346, March 2018. https://doi.org/10.1049/el.2017.3996

- **N. V. Vijaya Krishna Boppana**, Saiyu Ren, "A Low-Power and Area-Efficient 64-Bit Digital Comparator", Journal of Circuits, Systems, & Computers, Vol. 25, Issue 12, July 2016. https://doi.org/10.1142/S0218126616501486

- **N. V. Vijaya Krishna Boppana**, Saiyu Ren, "A simple yet efficient parallel signed multiplier design using a novel radix-8 structure" (Under Review)

Conference Papers

- **N V Vijaya Krishna Boppana**, Saiyu Ren, Henry Chen, "Low-power and high-speed CPL-CSA adder", NAECON 2014 - IEEE National Aerospace and Electronics Conference, pp. 346-350, June 24, 2014

6 List of Abbreviations

CCC	Control-Compute-Communicate
ADC	Analog-to-Digital Converter
CAD	Computer Aided Design
DSP	Digital Signal Processing
FIR	Finite Impulse Response
IIR	Infinite Impulse Response
LMS	Least Mean Squares
RLS	Recursive Least Mean Squares
FFT	Fast Fourier Transform
CPU	Central Processing Unit
GPU	Graphics Processing Unit
CAGR	Compound Annual Growth Rate
ANT	All-N-Transistors
MODL	Multiple Output Domino Logic
ASIC	Application Specific Integrated Circuits
PPG	Partial Product Generation
PPR	Partial Product Reduction
PPA	Partial Product Array

PDP	Power-Delay Product
PDA	Power-Delay-Area Product
EDP	Energy-Delay Product
WTM	Wallace Tree Multiplier
CBMW	Counter-based Modular Wallace (tree multiplier)
RCW	Reduced Complexity Wallace (tree multiplier)
R4B-RCW	Radix-4 Booth-Reduced Complexity Wallace (tree multiplier)
CBW	Counter-Based Wallace (tree multiplier)
MBE	Modified Booth Encoding
TDM	Three-Dimensional-reduction-Method
MLCSMA	Multiple-Level Conditional-Sum Adder
CSMA	Conditional-Sum Adder
CCA	Conditional-Carry Adder
PPRT	Partial Product Reduction Tree
B2C	Binary 2’s Complement
BEC	Binary Excess-1 Code
EAP	Energy-Area Product
CPA	Carry Propagation Adder
RCA	Ripple Carry Adder
SQCS	Square root Carry Select
CLA	Carry Look-ahead Adder
CPA	Carry Propagating Adder
RVT	Regular Voltage Threshold

HVT	High Voltage Threshold
LVT	Low Voltage Threshold
DC	Design Compiler (Synopsys RTL tool)
PnR	Placement and Routing
ICC	IC Compiler (Synopsys PnR tool)
PT	Prime Time (Synopsys STA tool)
STA	Static Timing Analysis
DAT	Data Arrival Time
NR	RCA-less optimization (Non-RCA)
NR-SO	Optimized Sign extension without intermediary RCAs

7 References

[1] I. G. Research, "Global Portable Medical Devices Market - 2021," [Online]. Available: https://www.researchandmarkets.com/reports/5451206/global-portable-medical-devices-market-2021#rela1-5317149. [Accessed 06 12 2021].

[2] G. Research, "Global Portable Medical Devices Market 2020-2030," [Online]. Available: https://www.researchandmarkets.com/reports/5397805/global-portable-medical-devices-market-2020-2030#rela2-5317149. [Accessed 06 12 2021].

[3] "wearable-medical-devices-market," [Online]. Available: https://www.grandviewresearch.com/industry-analysis/wearable-medical-devices-market. [Accessed 06 12 2021].

[4] A. D. Booth, "A Signed Binary Multiplication Technique," *The Quarterly Journal of Mechanics and Applied Mathematics,* vol. 4, no. 2, pp. 236-240, 1951.

[5] B. Parhami, Computer Arithmetic: Algorithms and Hardware Designs, New York, NY, USA: Oxford University Press, 2000.

[6] R. Panchangam, "Minimization of Power Dissipation in Digital Circuits Using Pipelinng and A Study of Clock Gating Technique," 2004.

[7] C. S. Wallace, "A Suggestion for a Fast Multiplier," *IEEE Transactions on Electronic Computers,* vol. 13, no. 1, pp. 14-17, 1964.

[8] W. J. Townsend, E. E. Swartzlander and J. A. Abraham, "A comparison of Dadda and Wallace multiplier delays," in *Proc. SPIE 5205, Advanced Signal Processing Algorithms, Architectures, and Implementations XIII*, San Diego, CA, United States, 2003.

[9] K. C. Bickerstaff, E. E. Swartzlander and M. Schulte, "Analysis of Column Compression Multipliers," in *Proceedings 15th IEEE Symposium on Computer Arithmetic. ARITH-15 2001*, Vail, CO, USA, 2001.

[10] K. A. C. Bickerstaff, M. Schulte and E. E. Swartzlander Jr., "Reduced are multipliers," in *Intl. Conf. on Application-Specific Array Processors*, Venice, Italy, 1993.

[11] V. Solanki, A. D. Darji and H. Singapuri, "Design of Low-Power Wallace Tree Multiplier Architecture Using Modular Approach," *Circuits, Systems, and Signal*

Processing, vol. 40, pp. 4407-4427, 2021.

[12] K. B. Jaiswal, N. K. V, P. Seshadri and L. G, "Low power wallace tree multiplier using modified full adder," in *3rd International Conference on Signal Processing, Communication and Networking (ICSCN)*, Chennai, India, 2015.

[13] M. Mehta, V. Parmar and E. Swartzlander, "High-speed multiplier design using multi-input counter and compressor circuits," in *Proceedings 10th IEEE Symposium on Computer Arithmetic*, Grenoble, France, 1991.

[14] S. Veeramachaneni, A. Lingamneni , K. K. Madhava and B. S. Mandalika, "Novel architectures for efficient (m, n) parallel counters," in *Proceedings of the 17th ACM Great Lakes symposium on VLSI*, Stresa-Lago Maggiore Italy, 2007.

[15] C. Fritz and A. T. Fam, "Fast Binary Counters Based on Symmetric Stacking," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 25, no. 10, pp. 2971-2975, 2017.

[16] A. Saha, R. Pal, A. G. Naik and D. Pal, "Novel cmos multi-bit counter for speed-power optimization in multiplier design," *AEU - International Journal of Electronics and Communications,* vol. 95, pp. 189-198, 2018.

[17] R. S. Waters and E. E. Swartzlander, "A Reduced Complexity Wallace Multiplier Reduction," *IEEE Transactions on Computers,* vol. 59, no. 8, pp. 1134-1137, 2010.

[18] S. Asif and Y. Kong, "Analysis of different architectures of counter based Wallace multipliers," in *Tenth International Conference on Computer Engineering & Systems (ICCES)*, Cairo, Egypt, 2015.

[19] S. Asif and Y. Kong, "Performance analysis of Wallace and radix-4 Booth-Wallace multipliers," in *Electronic System Level Synthesis Conference (ESLsyn)*, San Francisco, CA, USA, 2015.

[20] L. Dadda, "Some Schemes for Parallel Multipliers," *Alta Frequenza,* vol. 34, no. 5, pp. 349-356, 1965.

[21] E. E. Swartzlander, "Merged Arithmetic," *IEEE Transactions on Computers,* Vols. C-29, no. 10, pp. 946-950, 1980.

[22] L. M. G. Rocha, M. Macedo, G. Paim, E. Costa and S. Bampi, "Improving the Partial Product Tree Compression on Signed Radix-2m Parallel Multipliers," in *18th IEEE International New Circuits and Systems Conference (NEWCAS)*, Montreal, QC,

Canada, 2020.

[23] P. Patali and S. T. Kassim, "An Efficient Architecture for Signed Carry Save Multiplication," *IEEE Letters of the Computer Society,* vol. 3, no. 1, pp. 9-12, 2020.

[24] W.-C. Yeh and C.-W. Jen, "High-speed Booth encoded parallel multiplier design," *IEEE Transactions on Computers,* vol. 49, no. 7, pp. 692-701, 2000.

[25] O. L. Macsorley, "High-Speed Arithmetic in Binary Computers," *Proceedings of the IRE,* vol. 49, no. 1, pp. 67-91, 1961.

[26] V. Oklobdzija, D. Villeger and S. Liu, "A method for speed optimized partial product reduction and generation of fast parallel multipliers using an algorithmic approach," *IEEE Transactions on Computers,* vol. 45, no. 3, pp. 294-306, 1996.

[27] P. Stelling, C. Martel, V. Oklobdzija and R. Ravi, "Optimal circuits for parallel multipliers," *IEEE Transactions on Computers,* vol. 47, no. 3, pp. 273 - 285, 1998.

[28] K. Hwang, Computer Arithmetic: Principles, Architecture, and Design, John Wiley & Sons, 1976.

[29] K.-H. Cheng, S.-M. Chiang and S.-W. Cheng, "The improvement of conditional sum adder for low power applications," in *Proceedings Eleventh Annual IEEE International ASIC Conference*, Rochester, NY, USA, 1998.

[30] W. Gallagher and E. Swartzlander, "High radix booth multipliers using reduced area adder trees," in *Proceedings of 1994 28th Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, USA, 1994.

[31] A. Goldovsky, B. Patel, M. Schulte, R. Kolagotla, H. Srinivas and G. Burns, "Design and implementation of a 16 by 16 low-power two's complement multiplier," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, Geneva, Switzerland, 2000.

[32] C. R. Baugh and B. A. Wooley, "A Two's Complement Parallel Array Multiplication Algorithm," *IEEE Transactions on Computers,* Vols. C-22, no. 12, pp. 1045-1047, 1973.

[33] D. Kroft, "Comments on "A Two's Complement Parallel Array Multiplication Algorithm"," *IEEE Transactions on Computers,* Vols. C-23, no. 12, pp. 1327-1328, Dec. 1974.

[34] A. Khatibzadeh, K. Raahemifar and M. Ahmadi, "A 1.8 V 1.1 GHz novel digital multiplier," in *Canadian Conference on Electrical and Computer Engineering*, Saskatoon, SK, Canada, 2005.

[35] G. K. G and S. K. Sahoo, "Implementation of a high speed multiplier for high-performance and low power applications," in *19th International Symposium on VLSI Design and Test*, Ahmedabad, India, 2015.

[36] D. Jaina, K. Sethi and R. Panda, "Vedic Mathematics Based Multiply Accumulate Unit," in *International Conference on Computational Intelligence and Communication Networks*, Gwalior, India, 2011.

[37] T. G. Noll, "Carry-save architectures for high-speed digital signal processing," *Journal of VLSI signal processing systems for signal, image and video technology,* vol. 3, pp. 121-140, 1991.

[38] S. R. Huddar, S. R. Rupanagudi, M. Kalpana and S. Mohan, "Novel high speed vedic mathematics multiplier using compressors," in *International Mutli-Conference on Automation, Computing, Communication, Control and Compressed Sensing (iMac4s)*, Kottayam, India, 2013.

[39] J. F.-A. "M*N Booth encoded multiplier generator using optimized Wallace trees," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 1, no. 2, pp. 120-125, 1993.

[40] R. Katreepalli and T. Haniotakis, "Power-delay-area efficient design of vedic multiplier using adaptable manchester carry chain adder," in *International Conference on Communication and Signal Processing (ICCSP)*, Chennai, India, 2017.

[41] N. Boppana, J. Kommareddy and S. Ren, "Low-Cost and High-Performance 8 × 8 Booth Multiplier," *Circuits, Systems, and Signal Processing,* vol. 38, pp. 4357-4368, 2019.

[42] H. Xue, R. Patel, N. Boppana and S. Ren, "Low-power-delay-product radix-4 8*8 Booth multiplier in CMOS," *Electronics Letters,* vol. 54, no. 6, pp. 344-346, 2018.

[43] R. N. Patel, "Implementation of High Speed and Low Power Radix-4 8*8 Booth Multiplier in CMOS 32nm Technology," Wright State University, Master's thesis. OhioLINK Electronic Theses and Dissertations Center, 2017.

[44] Z. Zhang and Y. He, "A Low-Error Energy-Efficient Fixed-Width Booth Multiplier With Sign-Digit-Based Conditional Probability Estimation," *IEEE Transactions on Circuits and Systems II: Express Briefs,* vol. 65, no. 2, pp. 236-240, Feb. 2018.

[45] B. K. Mohanty and V. Tiwari, "Modified PEB Formulation for Hardware-Efficient Fixed-Width Booth Multiplier," *Circuits, Systems, and Signal Processing,* vol. 33, no. 12, pp. 3981-3994, 2014.

[46] L. Qian, C. Wang, W. Liu, F. Lombardi and J. Han, "Design and evaluation of an approximate Wallace-Booth multiplier," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, Montreal, QC, Canada, 2016.

[47] "Prime Time Static Timing Analysis," Synopsys, [Online]. Available: https://www.synopsys.com/implementation-and-signoff/signoff/primetime.html.

[48] "Design Compiler," Synopsys, [Online]. Available: https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/dc-ultra.html.

[49] S. Gorgin and G. Jaberipur, "Sign-Magnitude Encoding for Efficient VLSI Realization of Decimal Multiplication," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 25, no. 1, pp. 75-86, 2017.

[50] P. I.-J. Chuang, M. Sachdev and V. C. Gaudet, "A 167-ps 2.34-mW Single-Cycle 64-Bit Binary Tree Comparator With Constant-Delay Logic in 65-nm CMOS," *IEEE Transactions on Circuits and Systems I: Regular Papers,* vol. 61, no. 1, pp. 160-171, 2013.

[51] J. M. Rabaey, A. Chandrakasan and B. Nikolic, Digital Integrated Circuits, 2nd edition, Pearson, 2002.