DESIGN & ANALYSIS OF A NOVEL HYBRID 8 × 8 BIT MULTIPLY AND ACCUMULATE (MAC) ARCHITECTURE USING CLOCK GATING SCHEME FOR PIPELINED PROCESSING

A

Thesis

Submitted to

LOVELY PROFESSIONAL UNIVERSITY

Transforming Education Transforming India

For the award of

DOCTOR OF PHILOSOPHY (Ph.D.)

in

Electronics & Electrical Engineering

By

Rajkumar Sarma

41400103

Supervised by

Dr. Cherry Bhargava

Co-supervised by

Dr. Shruti Jain

LOVELY FACULTY OF TECHNOLOGY & SCIENCES
LOVELY PROFESSIONAL UNIVERSITY
PUNJAB

2020

CANDIDATE DECLARATION

I hereby certify that the work which is being presented in the thesis, entitled **“Design & analysis of a novel hybrid 8 × 8 bit Multiply and Accumulate (MAC) architecture using clock gating scheme for pipelined processing”** in fulfilment of requirements for the award of degree of Doctor of Philosophy in Electronics & Electrical Engineering is an authentic record of my own research work carried out under the supervision of Dr. Cherry Bhargava & co-supervision of Dr. Shruti Jain. The matter presented in this thesis has not been submitted elsewhere in part or fully to any other University or Institute for the award of any degree.

Rajkumar Sarma
(Registration No: 41400103)
School of Electronics & Electrical Engineering
Lovely Professional University
Phagwara. Punjab, India
Date:

CERTIFICATE

I hereby certify that **Rajkumar Sarma (Registration No: 41400103)** has prepared thesis entitled **“Design & analysis of a novel hybrid 8 × 8 bit Multiply and Accumulate (MAC) architecture using clock gating scheme for pipelined processing”**, for the award of degree of Doctor of Philosophy in Electronics & Electrical Engineering, under my guidance. The matter presented in this thesis has not been submitted elsewhere in part or fully to any other University or Institute for the award of any degree.

Dr. Cherry Bhargava
Associate Professor & Head
VLSI Design
School of Electronics & Electrical Engineering
Lovely Professional University
Phagwara, Punjab, India
Date:

Dr. Shruti Jain
Associate Professor
Department of Electronics & Communication Engineering
Jaypee University of Information Technology
Waknaghat, Solan, HP, India
Date:

ABSTRACT

In the era of digital signal processing, like graphics and computation systems, multiplication-accumulation (MAC) is one of the prime operations. A MAC unit is a vital component of a digital system, like different FFT algorithms, convolution, image processing algorithms, etcetera. In this research work, various MAC architectures, along with its sub-blocks such as adder and multiplier, are reviewed thoroughly. The study shows that the efficiency of a MAC unit, along with its sub-blocks is mainly dependent upon the speed of operation, power dissipation, and chip area of the circuit along with the complexity level of the circuit. Many of the researchers have also emphasized on optimization of these design constraints to make the MAC efficient. Earlier studies have stressed on increasing the efficiency of the overall MAC unit, whereas some have presented the techniques to produce remarkable efficiency of the sub-blocks. In this research work, the effectiveness of the MAC is further improved by adopting both the approaches, i.e., the overall architecture of the MAC unit is optimized by applying a novel algorithm and the performance of the sub-blocks of the MAC is maximized by choosing hybrid design techniques. Techniques such as block enabling and pipelining are adopted in the proposed MAC architecture to make the overall unit efficient. A novel Universal Compressor based Multiplier (UCM) architecture is also proposed to make the sub-blocks of the MAC more efficient.

The proposed UCM yields a high-speed operation, and hence, the enhanced performance is reported. The novel design of UCM is analyzed using the Cadence Spectre tool in 90 nm CMOS technology, which is further prototyped on the Nexys-4 Artix-7 FPGA board. Also, a Process-Voltage-Temperature (PVT) variation analysis is performed on the UCM architecture using Cadence ADE-XL for proper validation, which results in faster operation in ultra-low supply voltages (less than 0.9 V) for higher-order bit multiplication. In comparison to Wallace tree-based architecture (in 0.6 V to 0.9 V supply voltages), the proposed design has reduced the delay by 0.73% and 5.05% for 5 × 5-bit and 9 × 9-bit operations respectively.

The novel architectures for Unsigned MAC (UMAC), Unsigned Synchronized MAC (USMAC), Signed MAC (SMAC), and Signed Floating-point MAC (SFMAC) are designed using proposed UCM architecture. The designed architectures are simulated on CMOS 90nm technology using Cadence Virtuoso. The UMAC, USMAC, and SMAC can accommodate two 8-bit inputs and produces 16-bit output. Additionally, an extra bit is used in the case of SMAC architecture for representing a signed number.

On the other hand, each input of the SFMAC representation is of 13 bits, in which two bits are reserved for the sign bits of the number and its exponent. Remaining eleven bits are used for 8-bit binary representation and 3-bit exponent representation. Therefore, the input numbers in the proposed SFMAC have a range from $-(0.11111111)_2 \times 2^{+3}$ to $+(0.11111111)_2 \times 2^{+3}$ and hence, the range of the inputs in a decimal number system is from $-(7.96872)_{10}$ to $+(7.96872)_{10}$. The performance of UMAC, USMAC, SMAC, and SFMAC architectures are compared on the basis of power at 2V supply voltage, 20 ns simulation period, and 333.33 MHz clock frequency. It is inferred that the SFMAC results in maximum static and average dynamic power in comparison to other proposed MAC architectures because the transistor count in SFMAC is 2.5 and 5 times more than SMAC and USMAC architecture respectively. Furthermore, a power comparison of SFMAC architecture at different CMOS technologies (TSMC 130 nm and GPDK 90 nm) in a specific input vector is studied at a frequency of 83.33 MHz. Finally, a performance comparison of the proposed MAC architectures and the existing architectures are discussed in detail, which shows significant improvement in terms of static as well as average power.

Keywords: *Compressor-based Multiplier; Low power; High speed; Nexys-4 Artix-7 FPGA; Cadence Virtuoso; Signed-Floating-point MAC; Block Enabling; Clock Gating.*

ACKNOWLEDGEMENT

I am deeply indebted and grateful to many people who supported me during the research work and preparation of the thesis.

First and foremost, I give special thanks and glory to the God Almighty for giving me the wisdom and health to complete this endeavor.

I would like to express sincere gratitude and appreciation to my supervisor **Dr. Cherry Bhargava**, Associate Professor, School of Electronics & Electrical Engineering, Lovely Professional University (Phagwara, Punjab) for her constant encouragement, support and guidance. Her sincerity, positive and supportive attitude, calmness and scholarly advice have been a steady source of inspiration to me. Her guidance helped me all the time during my research work. I could not have imagined having a better advisor and mentor for my Ph.D. study.

It is an honor and great privilege to pay ineffable gratitude to **Dr. Shruti Jain**, Associate Professor, Department of Electronics & Communication Engineering, Jaypee University of Information Technology (Waknaghat, Solan, HP), for her guidance and assistance as co-supervisor. Her creative comments and suggestions from the initial conception till the completion of this work are highly appreciated. l am greatly indebted to her for her valuable advice, constructive criticism and their extensive discussions around my work.

I will forever be thankful to **Dr. Sandeep Dhariwal**, Associate Professor, Alliance College of Engineering and Design, Alliance University, (Bengaluru), for his encouragement and support. His invaluable and scholastic insights always motivated me.

I want to thank **Dr. Loviraj Gupta**, Executive Dean, Lovely Faculty of Technology & Sciences, Lovely Professional University (Phagwara, Punjab) for providing me valuable suggestion during my patent filing. This has helped me to expand my work in a different level.

I am greatly thankful to **Mr. Ashok Mittal**, Honorable Chancellor, Lovely Professional University (Phagwara, Punjab) for providing me required facilities to carry out my research work.

I express my veneration to **Prof. Bhupinder Verma**, Head of the School and all my colleagues of School of Electronics & Electrical Engineering, Lovely Professional University (Phagwara, Punjab) for their encouragement and support. I acknowledge the contribution of the technical and non-technical staff in the School of Electronics & Electrical Engineering, Lovely Professional University (Phagwara, Punjab) for providing necessary facilities during the pursuit of my research.

Nobody has been more important to me in the pursuit of this research work than the members of my family. I owe heartfelt thanks to my parents, my siblings and my in-laws for their love, motivation, encouragement and guidance when it was mostly required. Most importantly, I owe a lot to my ever supporting and caring wife **Dr. Subha Lakshmi Saikia,** who have provided me unending inspiration emotionally and mentally. I always fall short of words and felt impossible to describe her support in words.

Rajkumar Sarma

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

MAC	Multiply and Accumulate
FFT	Fast Fourier Transform
UCM	Universal Compressor-based Multiplier
FPGA	Field Programmable Gate Array
PVT	Process, Voltage and Temperature
ADE	Analog Design Environment
V	Voltage
UMAC	Unsigned Multiply and Accumulate
USMAC	Unsigned Synchronous Multiply and Accumulate
SMAC	Signed Multiply and Accumulate
SFMAC	Signed Floating-point Multiply and Accumulate
CMOS	Complementary Metal Oxide Semiconductor
MHz	Mega Hertz
GPDK	Generic Process Design Kit
DSP	Digital Signal Processing/Digital Signal Processor
PDA	Personal Digital Assistant
VLSI	Very Large-Scale Integration
NCSim	Incisive Enterprise Simulator
RTL	Register-Transfer Level
PPRT	Partial Product Reduction Tree
VHDL	VHSIC (Very High-Speed Integrated Circuit) Hardware Description Language
HDL	Hardware Description Language
PMOS	P-channel Metal Oxide Semiconductor
NMOS	N-channel Metal Oxide Semiconductor
IC	Integrated Circuits
µs	Micro Seconds
HA	Half Adder

FA	Full Adder
CPL	Complementary Pass Transistor Logic
DPL	Double Pass Transistor Logic
BCD	Binary Coded Decimal
V_{DD}	Supply Voltage
f_{clk}	Clock Frequency
α	Switching Activities
C_L	Load Capacitance
I/O	Input-Output
ULSI	Ultra-Large-Scale Integration
PC	Personal Computer
NiMH	Nickel-Metal Hydride
NiCd	Nickel-Cadmium
DEC	Digital Equipment Corporation
W	Watt
V_{SS}	Source Supply or Ground
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
τ_P	Propagation Delay
τ_{PLH}	Signal Switching from Logic Low to High
τ_{PHL}	Signal Switching from Logic High to Low
ECL	Emitter-Coupled Logic
TTL	Transistor-Transistor Logic
PDP	Power-Delay Product
BiCMOS	Bipolar Complementary Metal Oxide Semiconductor
CSA	Carry Select Adder
MAVIP	Multifunctional Architecture for Video and Image Processing
MIN	Minimum
MAX	Maximum
ASIC	Application Specific Integrated Circuits
SAD	Sum of Absolute Difference
CSD	Canonical Signed Digit

TSMC	Taiwan Semiconductor Manufacturing Company
MOSIS	Metal Oxide Semiconductor Implementation Service
CSLA	Carry Select Adder
FAM	Fused Add-Multiply
BEC	Binary to Excess 1 Converter
GDI	Gate Diffusion Input
DML	Dual Mode Logic
FADD	Floating-point Addition
LZA	Leading Zero Anticipatory
OBDD	Ordered Binary Decision Diagrams
DMT	Discrete Multitone
SCM	Single Carrier Modulation
VDSL	Very high-speed Digital Subscriber Line
RSFQ DS	Rapid Single Flux Quantum Digital Signal
GHz	Giga Hertz
FPMAC	Floating-Point Multiply and Accumulate
FMA	Fused Multiply-Accumulate
ISE	Integrated Synthesis Environment
PMADD	Parallel Multiply Add
nm	Nano Meter
APC	Adaptive Power Control
UMC	United Microelectronics Corporation
LNS	Logarithmic Number System
LS-DCCFF	Low-Swing Differential Conditional Capturing Flip-Flop
ASIP	Application-Specific Instruction-set Processor
mW	Milli Watt
TG	Transmission Gate
IEEE	Institute of Electrical and Electronics Engineers
PWM	Pulse Width Modulation
2D MAC	Two Dimensional Multiply-Accumulate
IP	Intellectual Property

LUT	Look-Up Table
SIMD	Single Instruction Multiple Data
PID	Proportional Integral Derivative
SIM	Simulator
FIR	Finite Impulse Response
IIR	Infinite Impulse Response
DA	Distributed Arithmetic
CNTFET	Carbon Nanotube Field-Effect Transistor
HSPICE	Hailey Simulation Program with Integrated Circuit Emphasis
HMC-MAC	Hybrid Memory Cube Multiply-Accumulate
CMAC	Complex Multiply Accumulate Cell
RPR-MAC	Reduced Precision Redundancy Multiply And Accumulate
FINFET	Fin Shaped Field Effect Transistor
DWT	Discrete Wavelet Transform
RAM	Random Access Memory
ROM	Read Only Memory
SOC	System on Chip
RCA	Ripple Carry Adder
M	Mantissa
B	Base
E	Exponent
PP-GENERATOR	Partial Product Generator
ND	New Data
EOS	End-Of-Symbol
EOU	End-Of-User
EOD	End-Of-Data
RFD	Ready-For-Data
CLK	Clock
PED	Pulse Edge Detector
ps	Pico Seconds
MUX	Multiplexer

FIFO	First in First out
ALU	Arithmetic and Logic Unit
MSB	Most Significant Bit
S BIT	Sign Bit
Exp S BIT	Exponents’ Sign Bit
EA	Exponential Adder
ECC	Exponent Comparator Circuit
ESC	Exponent Shifter Circuit
EXP	Exponent
LSB	Least Significant Bit
NUM	Number (Output)
GDS-II	Geometrical Data Base Standard for Information Interchange
FF	Fast-Fast
FS	Fast-Slow
NN	Normal-Normal
SF	Slow-Fast
SS	Slow-Slow
LED	Light Emitting Diode
IOB	Input Output Buffer

CHAPTER 1: INTRODUCTION

The invention of the TRANfer-reSISTOR (transistor) by William B. Shockley, Walter H. Brattain, and John Bardeen at AT & T Bell laboratories had changed the electronics industry dramatically and opened the way for the advancement of the Integrated Circuit technology. Jack Kilby designed the first IC at Texas Instruments in early 1960, and since then, there is an evolution of different generations of IC technology. The types of generation are based on the transistor count, such as SSI consisting of 10 to 100 transistors, MSI consisting of 100 to 1000 transistors, LSI consisting of 1000 to 10000 transistors, and VLSI consisting of more than 10000 transistors. The fifth-generation, which has emerged recently as ULSI for which the range of transistor count on a single IC chip is not defined yet. Further miniaturization is yet to come, and there must inevitably be more revolutionary progress in applying the ULSI technology.

Silicon CMOS technology has become the dominant manufacturing process for relatively high performance and cost-effective VLSI/ULSI circuits over the past several years. This development's ground-breaking essence is demonstrated by the rapid growth in which the number of transistors on a single chip integrated into circuits. Though transistor count (i.e. the area) is the primary reason for such development, energy efficiency and high-speed designs are also the primary concerns for the designers. Therefore, the typical design constraints of VLSI/ULSI circuits are power, delay and area. Any digital system's performance is measured concerning the power, delay and area. Design constraints can be explained as follows:

- *Timing*: Any circuit has specific timing requirements. A circuit with optimized delay is the prime concern for VLSI designers.
- *Area*: A circuit’s size can’t exceed the threshold limit. Here, circuit size refers to the backend design or final layout.

- *Power*: A circuit must have the capability to save as much as the power it can. But the VLSI designers must be careful while minimizing the power of digital circuits because a decrease in power consumption can make the circuit slower.

There is an inverse relationship between the area and time constraints. The design has to be parallelized (which usually means that larger circuits have to be designed) to optimize timing (faster circuits) constraint for a specific technology. Designers typically have to compromise on circuit speed to create smaller circuits. Figure 1.1 shows the inverse relationship.

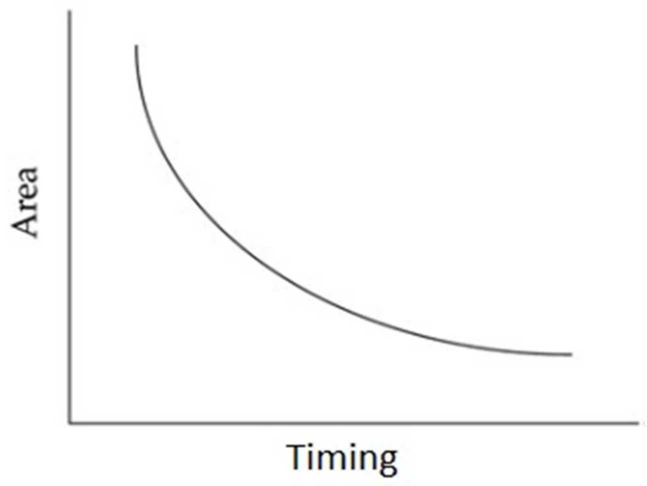


Figure 1.1: Area versus Timing trade-off

In addition to design constraints, the optimization of target technology is influenced by operating environment variables such as I/O delays, drive strengths and output loads. To ensure that the circuits are configured for the appropriate operating environment, operating environment factors must be input into the logic synthesis tool.

1.1 NEED FOR LOW-POWER DESIGN

The popularity of portable devices and the requirement to limit the power consumption (and therefore heat dissipation) in heavily-dense VLSI/ULSI chips have resulted in quick and revolutionary advances in low-power design over the past few years. Mobile applications necessitating low-power dissipation and high throughput, let’s say notebook PCs, mobile communication devices, and PDAs, are the driving forces behind these innovations. In most of these cases, low power consumption requirements need to be met along with equally challenging targets of high chip density and high speed. Therefore, the low-power IC design surfaced as a beneficial and fast-developing area of CMOS circuit design. Usually, the restricted battery life places very stringent demands on the portable system's overall power requirements. New types of rechargeable batteries say "Nickel-Metal Hydride (NiMH)" are being produced with better energy storage capacity than the traditional "Nickel-Cadmium (NiCd)" batteries. Still, there is no prospect of a significant increase in energy capacity in the foreseeable future. The energy density (which is the energy stored/unit weight) provided by new advancement in technologies (such as NiMH) is approximately 30 Watt-hour/pound, which is quite lesser considering the growing applications of portable systems. Scaling down the energy dissipation of ICs by improving functionality is, therefore, a significant task in the development of portable devices.

In high-performance digital systems, such as microprocessors-microcontrollers, DSPs, etcetera, the need for low-power circuit development is also becoming a significant concern. Targeting higher chip density and higher processing speed contributes to the development of high-clock rate in very complex circuits. If the chip's clock speed rises then the chip's energy dissipation, thereby increasing the temperature linearly. As the dissipated heat has to be efficiently removed to maintain the temperature of the chip at an optimum level, the packaging cost, cooling and heat extraction becomes an important aspect. A few elite microchips structured in the mid-1990s (such as, Intel Pentium, DEC Alpha, PowerPC) which operates in a frequency ranging from 100-300 MHz, and the total average power is ranging from 20-50 W. ULSI's reliability is one more critical factor to look after for the design engineers, as it emphases to the demand for energy-efficient design. There is a near connection between electronic circuit maximum power-dissipation and reliability concerns like electro-migration and system degradation caused by the carriers. Additionally, the thermal stress caused by chip heat dissipation is also a significant issue to look after in terms of reliability. As a consequence, increasing power-consumption is also critical for improving performance.

The procedures used in digital systems to achieve low-power consumption vary from device to device, technology to technology or algorithm to algorithm level. The standard system features (say threshold voltage), device dimension and interconnection properties are essential factors in reducing power consumption. Circuit level approaches such as a careful selection of circuit design logic family, decrement in the total number of voltage transitions and clocking approaches can be used to minimize transistor-level energy dissipation. Measures at the architecture level include intelligent power management of different system components, pipeline and concurrent usage, and bus layout design.

Lastly, a good set of data processing algorithms also reduces the power consumed by the device as it reduces the number of switching activity for a particular task.

1.1.1 Causes for power dissipation

The energy or power dissipation in CMOS based circuits is categorized into three main categories, namely,

1. *Switching or Dynamic power consumption*
2. *Short circuit power consumption*
3. *Leakage power consumption*

A fourth power element, namely static power, would also be considered if the device or chip contains circuits other than standard CMOS gates that have direct current paths between V_{DD} and V_{SS}.

Switching or Dynamic power consumption:

Dynamic power is the dissipation of energy during a switching activity which means that a CMOS logic gate's output node voltage makes a switch that consumes electricity. For digital CMOS circuits, as energy is collected from the V_{DD} to charge the capacitance at the output node, dynamic power is dissipated. The output node voltage usually transitions from 0 to V_{DD} during the charging cycle, and the power used for the conversion is relatively independent of the circuit's functionalities.

Short-Circuit Power Consumption:

The dissipation of the dynamic power described in the last sub-section is simply due to the power needed to charge the parasitic capacitance in the circuit, and the dynamic power is non-dependent on the input signal's rise/fall times. Now, in a situation where a CMOS logic gate is controlled with finite rise/fall time on the input waveforms, both the N-Channel MOSFETs and the P-Channel MOSFETs in the design may conduct momentarily and concurrently for a small duration of time during the transitions. This eventually forms a direct current path between the V_{DD} and the V_{SS}.

Leakage Power Consumption:

The N-Channel MOSFETs and the P-Channel MOSFETs used in digital designs using CMOS circuits usually have reverse leakage currents as well as sub-threshold currents with non-zero values practically. In a chip containing an enormous number of transistors, these flows of current can add to the total energy or power dissipation even when the transistors are not performing any transient activity. Primarily the processing parameters determine the scale of the leakage currents. The leakage current components found in N-Channel MOSFETs and P-Channel MOSFETs are:

A. Reverse-diode leakage current
B. Sub-threshold leakage current

1.2 FACTORS AFFECTING HIGH-SPEED DESIGN

The delay for a CMOS based circuit relies on the charge-discharge rate at the output of all capacitors. The capacitance of all capacitors connected to the circuit is due to two elements called the parasitic capacitance and the load capacitance. The propagation delay (τ_p) of a CMOS inverter is given by equation 1.1.

$$\tau_p = 2C_L/KV_{dd} \qquad (1.1)$$

The delay in propagation (in general "propagation delay") is the time taken to transfer a signal to the output from the input. Typically, it is defined between the 50% points, as shown in figure 1.2. The propagation delay of the logic gate is the mean of the output signal switching from logic low to high (τ_{PLH}) and high to low (τ_{PHL}). As shown in figure 1.2, the dotted lines, i.e. the ideal input or output which has immediately changed from low to high and high to low. But practically, any system can't change abruptly from logic high to low or vice versa. Therefore, there is a requirement for rise and fall time. Rise time is the time taken by a signal to change from 10% to 90% of the final value; whereas the fall time is the time taken by a signal to change from 90% to 10% of the final value.

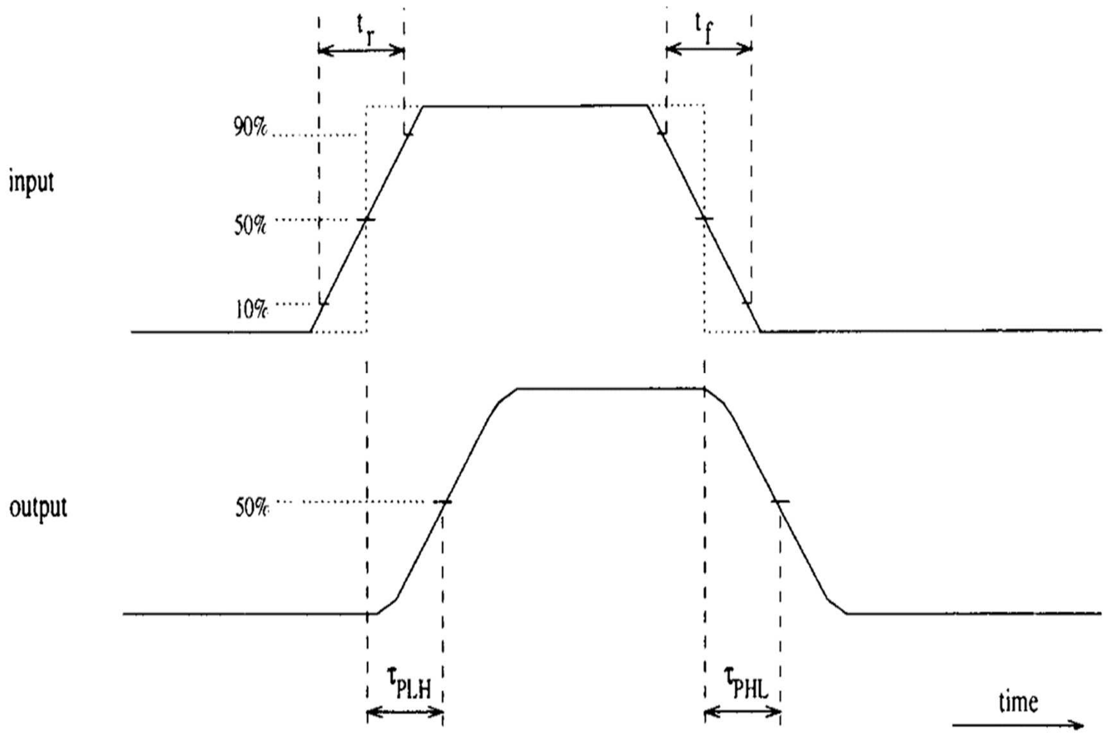


Figure 1.2: Rise-fall time during input output transitions

For CMOS inverter, as shown in equation 1.1, the propagation delay varies directly with the changes of load capacitance and varies inversely with the value of 'K'. The same relation can be obtained for the output transistor in bipolar technology. The relationship of propagation delay and load capacitance is shown in figure 1.3 in graphical representation for three logic families, i.e. ECL, CMOS and TTL. As the graph depicts, the delay is low at low capacitances in the CMOS logic family in comparison to TTL logic. The main reason for the same is the load capacitance is an external capacitance, and it doesn't include the internal capacitance of the logic gate. The internal capacitance for CMOS devices is smaller than bipolar devices because a CMOS device takes considerably smaller space in the layout than the bipolar device.

Hence the larger size of the device offers higher input capacitance. However, as the load capacitance is much bigger than internal load capacitance, its influence is not visible in the propagation delay.

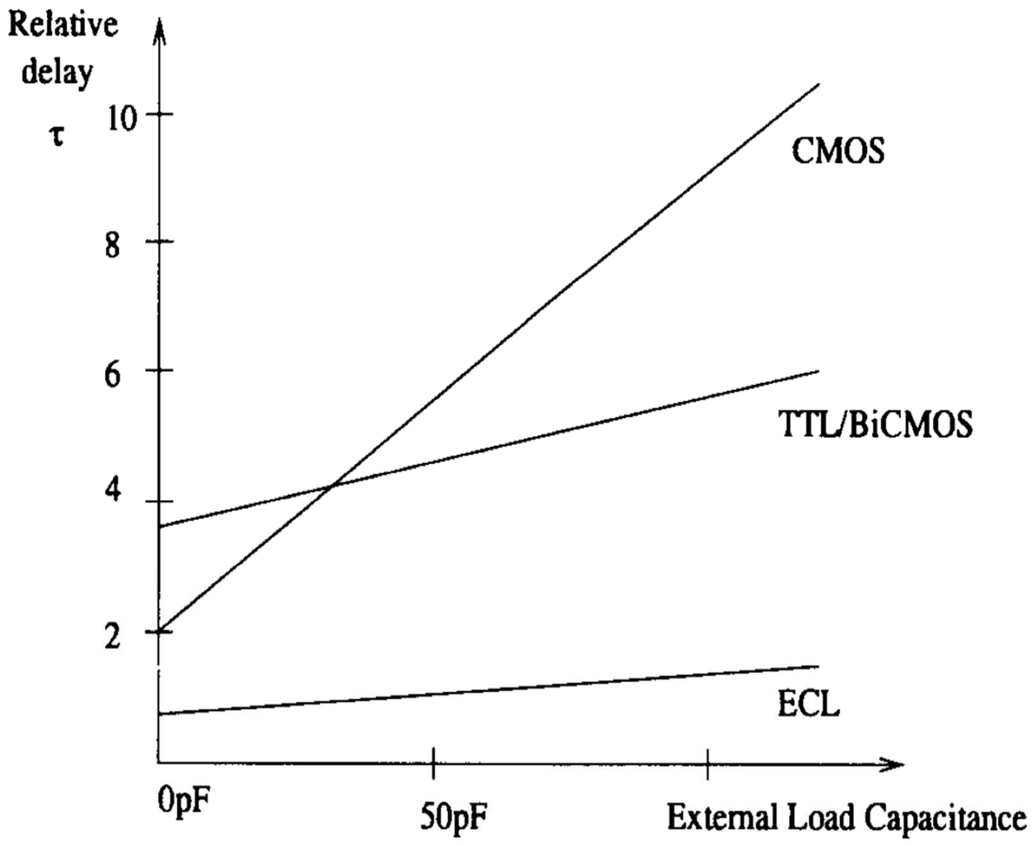


Figure 1.3: Propagation delay versus load capacitance for different logic families

On the other hand, the practical value of K is more significant for bipolar devices than CMOS devices. Therefore, with larger load capacitance the propagation delay for CMOS devices are more than that of TTL devices. Thus, if a large capacitance to be driven, i.e. the system has large fan-in, then bipolar devices are preferred. On the other way around, if the system has low output capacitance to drive, (i.e. < 30 pF), then CMOS can be preferred. For ECL logic family, the delay versus capacitance shows that these devices are fastest among all three since ECL logic systems don't enter saturation. Therefore, a circuit with higher speed and lower power consumption is always desired. Moreover, as there is a trade-off between the power consumption and delay, the performance of a circuit is mostly evaluated in terms of Power-Delay Product (PDP).

1.3 INTRODUCTION TO MULTIPLY & ACCUMULATE (MAC) ARCHITECTURE

Today’s portable devices are capable of doing image filtering to face recognition, an audio signal enhancement to voice recognition and gesture-based control to biometric authentication. All those functionalities are the applications of Digital Signal Processing (DSP). A large number of mathematical operations are performed repeatedly and quickly on a series of data samples by DSP algorithms. Most operating systems and general-purpose microprocessors can successfully execute DSP algorithms. Still, because of power efficiency constraints, they are not suitable for use in portable devices such as PDAs and mobile phones. However, the rapid growth of portable electronics has introduced the significant challenges of low power and high throughput for VLSI/ULSI design engineers.

Among the other digital blocks, Multiply and Accumulate (MAC) unit plays a vital role while evaluating the performance of a DSP block. While performing convolution, filtering or any other DSP operations, it is always desired to use an efficient MAC unit. The efficiency of a MAC unit is measured in terms of two factors:

1) Speed of operation
2) Overall power consumption [1, 2]

The essential operation of the MAC is to fetch the inputs from the input devices or memory and process it through the multiplier block and provide the result to an adder which sum-up the current multiplier output with the previously accumulated result and then again accumulate the result in an accumulator register. Generalized block diagram of 8 × 8-bit MAC is shown in figure 1.4. The MAC architecture contains the main functional blocks as multiplier, adder and register/accumulator. The multiplier performs the multiplication operation over the two input operands; the adder performs the addition of the result of the multiplier with the result of the previous cycle and the register or accumulator stores the sum for next cycle addition. Different approaches for multiplication as well as the addition for MAC operation is described in detail in the literature by [3, 4] etcetera. Mathematically, the operation of the MAC is to generate the product of two operands X_i and Y_i and add the result with the previously stored result from the last multiplication in a single clock period [5]. The operation of MAC can be expressed, as mentioned in equation 1.2.

$$F = \sum_{i=0}^{n-1} X_i Y_i \tag{1.2}$$

Where ‘i’ denote the range of the values.

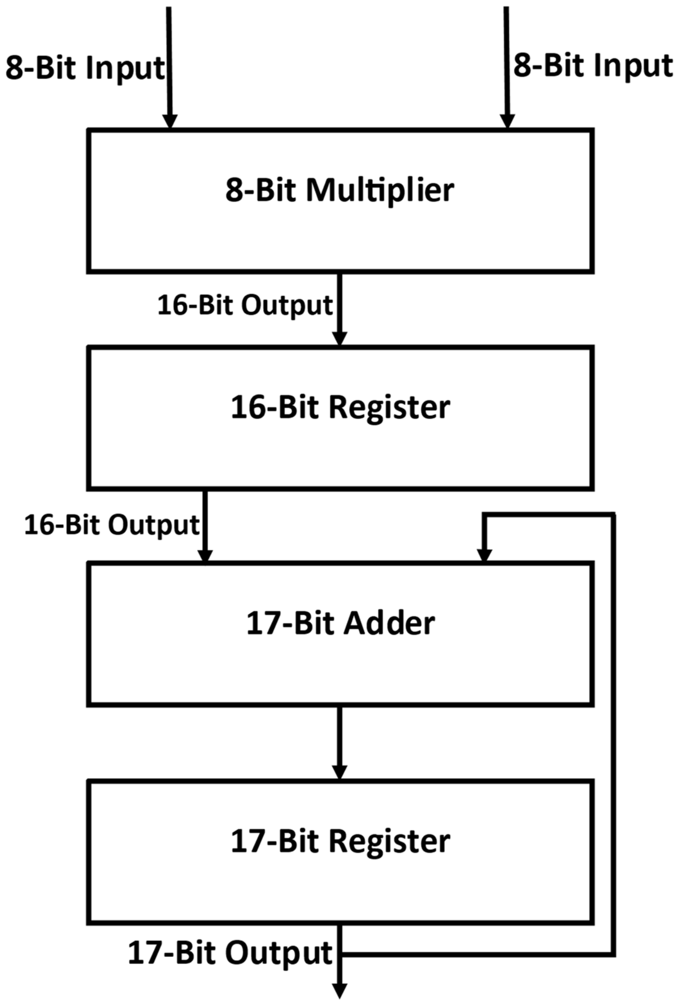


Figure 1.4: Generalized block diagram of 8 × 8 bit MAC

A high-speed MAC architecture which promises with an optimized area is proposed in [1]. It uses 4:2 compressor circuits to improve speed. In 2012, a novel architecture for the multiplier is proposed by [6]. In 2013, a novel architecture using modified Wallace tree multiplier is proposed by [7]. The implementation is done for 64 bits. Modified Braun multiplier is used to implement a basic MAC unit in [8]. The implementation is done on NCSim and RTL Compiler. A low power Baugh Wooley multiplier-based unit is proposed in the year 2014 by [9]. A pipelined based architecture has been proposed in this work. Split MAC architecture is explained by [10]. A technique to compress the partial product using "interleaved adders" and a "modified hybrid Partial-Product-Reduction-Tree (PPRT)" schemes are proposed in this work to enhance the speed of operation further. There are several architectures explained in the past by various designers. However, all these different architectures (90% of them in the literature) are designed with the help of Hardware Descriptive Languages (HDL) such as Verilog or VHDL. The main disadvantage of using HDL is that the basic blocks, those are to be used while designing any architecture, use the predefined system defined primitives (standard PMOS-NMOS implementation). Because of which, even after using smart and efficient structural designs, the architecture lags in certain aspects. The main reason for such a shortcoming is the non-optimization of basic building blocks viz. multiplier, accumulator and adder.

1.3.1 Multiplier

In DSP architectures, multiplication is the fundamental operation. Multipliers require large area (because of partial product generation), long latency and consume relatively higher power than adder/subtractor circuits. Any multiplier-based system's performance is evaluated based on the optimization of the primary design constraints (explain later in this chapter). The reason for the same is that the multiplier is the slowest unit in the arithmetic system. Hence, maximizing the speed of operation of the multiplier along with optimization of power and area is the primary concern for any system design. However, the trade-off between area and speed & power and speed are unavoidable. Therefore, minimizing one of design constraint (power, delay or speed) may have the possibility to increase the other one. Moreover, as mentioned above, the hardware requirement in multiplier circuit is enormous. Hence, low power design is a challenge as it has become the authoritative measures for designing the power-efficient multiplier designs for high speed and compact devices. As mentioned earlier, the multiplier is one of the central units for designing a power-efficient circuit, where the multiplier block decides the efficiency of the DSP. Therefore, extensive research work has been performed on low power multiplier designs with different area-speed constraints.

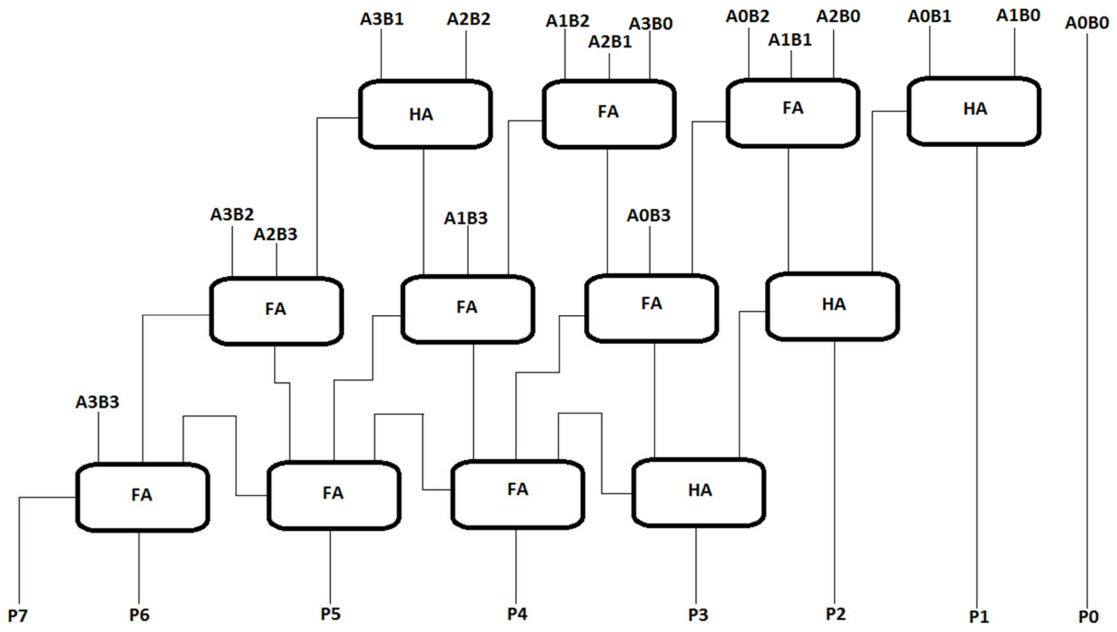


Figure 1.5: 4-bit Wallace tree multiplier

Table 1.1: Comparisons of performance parameters for different logic styles

Multiplier Type	**Logic Style**	**Delay (ns)**	**Power (µW)**	**PDP (fJ)**	**No. of Transistor**
Array	CMOS	8.300	10.73	89.06	384
	CPL	4.337	24.70	131.82	368
	DPL	4.667	19.72	92.03	448
Tree	CMOS	4.247	10.68	45.35	384
	CPL	4.105	23.61	125.25	368
	DPL	4.526	19.87	89.93	448

Australian computer scientist Chris Wallace proposed a fast multiplication technique in the year 1964 [11]. The hardware requirement in this architecture is very high, but it reduces the delay substantially. The architecture promises to get the products and quotients within a time of 1 µs and 3 µs respectively if it is used in diode-transistor logic. The architecture proposed in [11] can be used where a high-speed design is a primary concern, not the regularity of the structure. Figure 1.5 shows the conventional Wallace tree architecture. As mentioned by [12] in 2012, “The Wallace tree multiplier is faster than an array multiplier because its height is logarithmic in word size, not linear”. The only disadvantage of Wallace tree multiplier is that its irregular structure. In the recent past, many attempts are made to modify the Wallace tree structure, but a hand full of attempts are made to make the design regular. The performance comparison of array multiplier and Wallace tree in different logic style is given in table 1.1 [13].

1.3.2 Accumulator

An accumulator or register is a temporary storage where the internal as well as final arithmetical and/or logical results are stored. Without an accumulator or register it becomes very crucial to store the outcome of each and every operation (summation, multiplication, shift, etcetera) to the main memory. The main reason to use the accumulator or register is to read the stored data in the immediate previous cycle and to use it in the next operation because mathematical operations often take place in a stepwise manner, using the results from one operation as the input to the next. Moreover, the main memory access is slower than accessing an accumulator or register repetitively; which eventually decreases the speed of operation of the circuit. But it is to be noted that, though the technology used for accessing large main memory is slower but its design cost is cheaper than that of an accumulator or register as the memory.

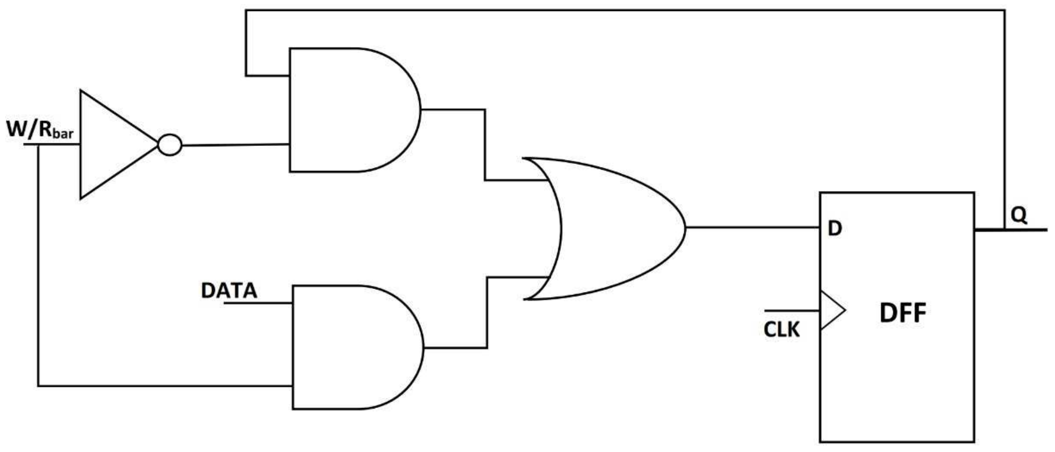


Figure 1.6: Basic accumulator or register circuit

The fundamental element constituting an accumulator or a register is a D-flip-flop which can store a 1-bit of data. Two AND gates with clock input are also used. Hence, the register cell has three inputs, namely "write or negation of read", "clock" and "D". The output of the block is Q. Figure 1.6 shows the single-bit register [5].

1.3.3 Adder

An adder is also known as summer is a logic circuit which adds two numbers. Adders or summer circuit is used not only for addition but also for multiplication, updating the addresses, increment/decrement operations, table indices etcetera. The adder operation is performed in binary number systems, but the adder can also be applied on BCD, excess -3 etcetera. Adders are of two types:

- **Half Adder:** It adds two 1-bit binary numbers and the outputs are 'sum' and 'carry' values. For 'sum' output is the XOR of the two inputs whereas, the 'carry' output is the AND of the two inputs. Half adder is used rigorously in full adder circuit, multi-bit adder circuit, multiplier circuit etcetera.
- **Full Adder:** It performs addition operation on three 1-bit variables and produces the 'sum' and 'carry' outputs. It takes into account the carry input also. Most of the n-bit adder architectures utilize full adders. The multiplier, adder-cum-subtractor circuit etcetera use the full adder circuit rigorously.

1.3.4 Block enabled technique & pipelined architecture

As the feature size is scaled down, low power is the most critical issue in today's VLSI design. Block Enabling is one of the most elegant and classic technique for reduction of dynamic power, a significant contributor in total power consumption of any VLSI circuit [13].

$$P_{dynamic} = P_{int\,ernal} + V_{DD}^{2}.f_{clk}.\alpha.C_L \qquad (1.3)$$

The mathematical expression for dynamic power is shown in equation 1.3, where 'V_{DD}' is the supply voltage, 'f_{clk}' is the clock frequency, 'α' represents the switching activities at nodes and 'C_L' represents load capacitances. Block enabling technique facilitates saving of electrical power used by digital signal processors by reducing the switching activity 'α'. The power-saving is ensured in this technique by activating the design block as and when required. For this, initially, the delay for each building block of the architecture needs to be calculated. Every building block of the architecture gets enabled only after the desired delay required by that block to produce the output correctly. The successive blocks are disabled until the inputs are available to the respective block and thus saving power [14].

The basic idea of pipelining comes from everyday life. For example, water pipe continuously sends water without waiting for the water previously sent to be out, which leads to a reduction in critical paths. In DSP, pipelining either reduces the power consumption at the same speed or increases the clock speed. In the buffered and synchronous pipelined architectures, "pipeline registers" are introduced between the functional blocks, and are synchronized (using a clock pulse). The delay between each clock signal is set in such a way that when the registers are clocked, the data stored in it is passed to the next stage. The representation of pipelined architecture with block enabling technique is shown in figure 1.7.

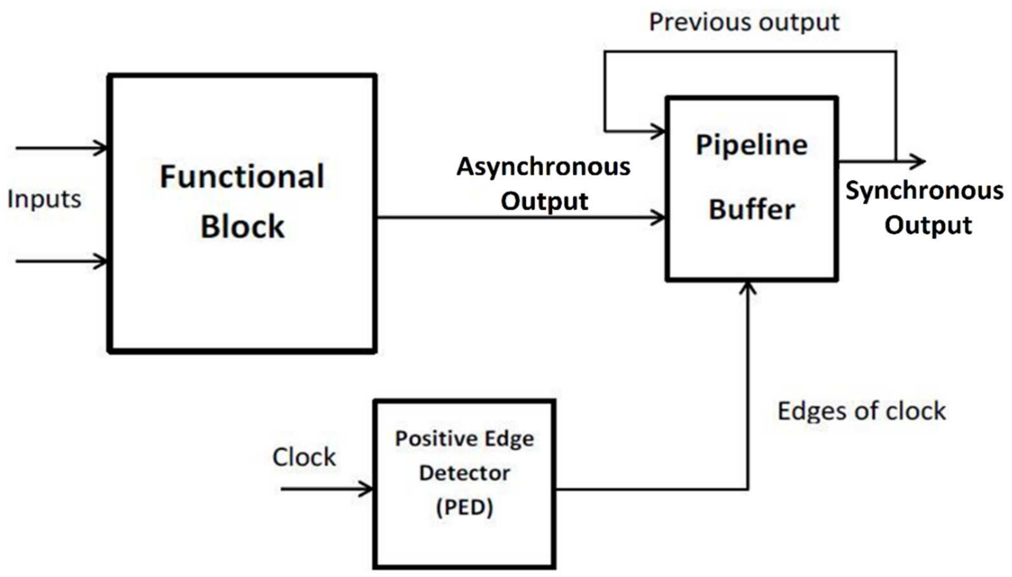


Figure 1.7: Pipelined and Block enabled Architecture

The main objective of the research work is to investigate various pipelined MAC architectures which are efficient in terms of the implementation of the high-yielding signal processing architectures and also to have lesser power consumption. This is because, the power consumption, speed and high-yielding rates are always interlinked with the DSP systems. Initially, a 1 × 1-bit fixed point unsigned MAC unit is designed in full custom IC design platform (using Cadence Virtuoso) with appropriate geometries to produce optimized power, area, and delay. Similarly, using the same concept, a 1 × 1-bit floating point signed/unsigned MAC unit is proposed and later the work is extended till 8 × 8 bit fixed-point signed/unsigned number and 8 × 8 bit floating-point signed/unsigned number. Full custom IC design platform is chosen for this research work to optimize the essential and fundamental building block.

1.4 SIGNIFICANCE OF THIS RESEARCH

As discussed earlier in this chapter, the multiplier and accumulator are the critical components of MAC architecture [1-9]. As the efficiency of the MAC is dependent upon the efficiency of the multiplier (mainly), an efficient multiplier (in terms of typical design constraints) design can further improvise the efficiency of a MAC unit. Moreover, the existing multipliers in the literature [11-13] are mostly based on the Wallace tree algorithm. It is claimed that the multipliers based on the Wallace tree reduce the steps involved to add the partial products. Still, it uses half adder or full adder for the addition of partial products which increases the complexity of the circuit. Further, any electronic circuit can be designed by two different approaches, namely the top-down approach and the bottom-up approach. In the top-down approach, the designs are implemented by focusing mainly on the output efficiency of the overall design. i.e. importance is given on the implementation of the process or algorithm, not on the optimization of primary cells. On the other hand, in the case of the bottom-up approach, the whole digital architecture is designed starting from its primary cell, i.e. importance is given on optimization of the primary cell as well as on the practical implementation of the algorithm. In the existing literature, full custom circuit design for the MAC unit has never been proposed [1-9]. Additionally, synchronization, clock gating techniques and pipelining can further enhance the speed of operation and minimize the power consumption.

Therefore, in this research work, a universal compressor (N:M of any size) based multiplier is proposed to use it as the core of the proposed MAC unit to improve the efficiency. Additionally, a full custom IC approach with synchronization, clock gating techniques and pipelining is adopted in the design of the proposed MAC to optimize the overall architecture which eventually provides much more efficiency in terms of power as well as delay.

1.5 ROADMAP OF THE THESIS

Chapter 2 offers a detailed review of literature based on adder, multiplier and MAC architectures. It has presented the recent developments in these areas in recent years. Based on the literature survey, the objectives of this research work are framed.

Chapter 3 shows the design and implementation of the novel UCM architecture, which promises higher speed at ultra-low supply voltages (less than 0.6V). A novel universal compressor (N:M of any size) is used for the addition of partial products while designing the multiplier. The multiplier is named as Universal Compressor-based Multiplier (UCM). The prototype of the proposed multiplier is implemented on FPGA. The UCM architecture is applied for developments of different architectures of MAC for fixed-point unsigned/signed, and floating-point unsigned/signed operations in chapter 4 and 5.

Chapter 4 discusses the UMAC, USMAC and SMAC architectures which are specialized in unsigned, synchronized-unsigned and synchronized-signed operations respectively for fixed-point inputs. The novel UCM architecture explained in chapter 3 is used for designing the MAC architectures. The graphical outputs of the UMAC, USMAC and SMAC architectures shows the accuracy of the designs and advantages of one over another.

Chapter 5 discusses the implementation of the SFMAC architecture, which is capable of performing signed/unsigned fixed-point or signed/unsigned floating-point MAC operation on given 8-bit inputs. The SFMAC architecture is the further extension of the MAC architectures proposed in chapter 4. The block enabling technique is deployed along with pipelining to optimize the power consumption of the proposed SFMAC architecture.

Chapter 6 consists of the detailed results and discussion of the proposed architectures. A comparative analysis is also shown in this chapter.

Finally, the conclusion of the thesis, its importance and its future works that can be adopted, are addressed in chapter 7.

CHAPTER 2: REVIEW OF LITERATURE

As mentioned in the previous chapter, the essential component of a MAC unit is a multiplier; on the other hand, the integral component of a multiplier is an adder or summer. Therefore, this section is explained in two parts; namely i) Adder and Multiplier and ii) Multiply and Accumulate unit.

2.1 ADDER AND MULTIPLIER

(Wallace, 1964): A m × n bit multiplier using combinational logic (in one gating step) is proposed. The proposed architecture promises to get the products and quotients within a time of 1 µs and 3 µs respectively if it is used in diode-transistor logic. Moreover, a rapid square-root process is also discussed [11].

(Itoh, et al., 2001): In this work, a rectangular styled Wallace-tree architecture is proposed. As stated, the partial products are segregated into two groups and summed up separately in top-down and bottom-up directions [15].

(Onomi, et al., 2001): A Wallace-tree multiplier architecture suitable for pipeline scheme is proposed in this research, where "carry-save adders are used for the addition of partial products". In this proposed work, the authors have claimed for removing the irregularity present in a conventional Wallace tree architecture [16].

(Liao, Su, et al., 2002): A CSA portioning algorithm is proposed in this paper, which is applied to the Booth-encoded Wallace-tree algorithm. As stated by the authors, "by taking into various data arrival times, a branch-and-bound algorithm is proposed and a heuristic to partition an n-bit carry-select adder into several adder blocks so that the overall delay of the design is minimized" [4].

(Guevorkian, et al., 2005): An architecture targeting mobile multimedia systems is proposed in this paper by introducing a “MAVIP”, which is a “reconfigurable extension derived from a high-radix multiplier structure". A MAVIP may be configured "either to a processing unit with DSP-specific operations such as multiplication, multiply-accumulate, parallel addition, MIN/MAX, etcetera or one/another ASIC such as a matrix-vector multiplier, FIR filter or SAD accelerator" [17].

(Kuo, et al., 2008): Low power high-performance latch adder-based Wallace tree multiplier has been proposed. The proposed techniques-based tree multiplier provides 22.3-23.7% of lesser delay and 5.5-3.3% of lesser power consumption than the conventional traditional latch-adder technique-based tree multiplier [18].

(Chen, et al., 2008): Canonical Signed digit multiplier is proposed with the help of Wallace tree adder is proposed. CSD requires to the lookup table for fetching the data from memory. Hence the speed of operation has improved. Finally, the FPGA implementation is done [19].

(Yi, et al., 2009): In this research work, a modified booth algorithm is studied and proposed which yields a variable bit-length multiplier. The proposed multiplier can perform "a 32 × 32-bit or dual 16 × 16-bit or four 8 × 8-bit multiplications, which greatly enhance the parallelism of the multiplier". The overall implementation is performed in Verilog HDL [20].

(Nachtigal, et al., 2010): In this research work, reversible design of single-precision floating-point multiplier is proposed which uses a technique called "operand decomposition approach". To design a "reversible 24 × 24-bit multiplier", the operands are partitioned into three groups consisting of 8 bits each. Therefore, the "24 × 24 bit reversible multiplication" is performed using nine "reversible 8 × 8-bit Wallace tree multipliers" and then the outputs are summed to get the final result [21].

(Singh, et al., 2012): Various logic style-based "1-bit full adders" and "AND2 function" are designed in this paper and used for designing 4 × 4 unsigned arrays and tree multiplier. The full adders and AND2 function are designed in different logic techniques such as CMOS logic, CPL logic and DPL logic style to improve the area, power, delay and PDP [13].

(Rao, et al., 2012): An improved version of tree-based Wallace tree multiplier architecture using Booth Recorder is proposed in this work. This proposed architecture reduces the latency and area of Wallace tree multiplier with the help of the Booth algorithm and compressor adders. The overall implementation is performed in Verilog HDL [12].

(Sousa, 2013): In this paper, an improved version of modulo (2^n + I) multipliers is proposed. The efficiency is achieved by "manipulating the Booth tables and by applying a simple correction term" in the existing modulo (2^n + I) multiplier algorithm. Moreover, the author states in the paper that "the proposed multiplier is almost as efficient as those for ordinary integer multiplication" [22].

(Khan, et al., 2013): The complexity of Wallace tree multiplier reduced in this research work without compromising with the delay. As full adder is used gregariously in Wallace tree multiplier (in partial product reduction as well as in the form of carry-propagation-adder), an "energy-efficient CMOS full adder" is used at the place of full adder standard cell to reduce power, area and delay [23].

(Kshirsagar, et al., 2013): For simultaneous arithmetic operation and therefore, to increase the speed of operations, a "four-stage pipelining at the intermediate nodes" is discussed in this proposed work. The architecture is designed in Verilog HDL and simulated using Cadence Spectre tool at TSMC 45nm technology. Cadence RTL Compiler is used for detailed analysis of the circuit [24].

(Jayaprakash, et al., 2013): This paper proposed a novel "low-power hybrid full adder" which consumes deficient power. The same is compared with its conventional counterpart (28T). The power consumption is found to be low in this design. The implementation is done on MOSIS 90 nm Technology [25].

(Bhattacharyya, et al., 2014): A hybrid full adder based on CMOS and transmission gate technique is proposed in this paper. The design is also extended till 32-bit full adder operation. The implementation of the circuit is done in Cadence Spectre tool in 90 nm and 180 nm CMOS technology [3].

(Paradhasaradhi, et al., 2014): The "Modified CSLA (MCSLA)" is proposed in this paper, which is designed using "Common Boolean Logic" and implemented using the Wallace Tree Algorithm. The implemented MCSLA is compared with regular CSLA architectures. The proposed work requires lesser area in comparison to normal Wallace tree multiplier [26].

(Luu, et al., 2014): An unsigned 32-bit multiplier for best timing performance with the optimized area is proposed in this research paper. The architecture uses "a modified Radix-4 Booth encoder, a modified Wallace Tree adder, and a Carry Look Ahead adder" [27].

(Reddy, et al., 2014): In this paper, a Gate Diffusion Input technique based low-power multiplier for 8-bit operation is proposed. The reduction in power and area is achieved by using “Booth encoding and Wallace tree technique” as this algorithm generates the minimal number of partial products for signed number multiplication and provides an efficient way to add the partial products [28].

(Srinitha, et al., 2015): A VHDL based high performance Fused Add-Multiply (FAM) unit architecture is proposed in this research work. The proposed architecture uses 4:2 compressor block instead of full adder/half adder [29].

(Jaiswal, et al., 2015): A MUX based full adder is proposed in this research article and then, the work is further extended for designing a Wallace tree multiplier. Because of the optimization of the adder, the performance of the multiplier has got improved. The architectural design is done in Verilog, and the functionalities are confirmed using Quartus II [30].

(Shoba, et al., 2017): A **“**CslA and Binary to Excess 1 Converter (BEC)" based multiplier is proposed in this paper. Because of the use of the BEC, the total number of adders is reduced by n/4 than orthodox addition scheme (here 'n' is the width of the input). Moreover, a Vedic multiplier is used as a base multiplier which requires lesser area and lesser delay. Additionally, Gate Diffusion Input (GDI) logic style is used for designing the proposed multiplier. The functionality of the proposed multiplier is analyzed and verified by Cadence Spectre Tool in 45 nm CMOS technology. From the comparative analysis, it is found that the proposed multiplier requires 17% lesser PDP than its close competitor. The Monte Carlo simulation is also performed to analyze the performance in extreme conditions [31].

(Ozcan, et al., 2018): A "Montgomery multiplier" which works iteratively is proposed in this work. A digit of the multiplier is multiplied by the digits of the multiplicand in every iteration. And the result is stored in an accumulator. Each time the total number of multiplier and multiplicand is reduced by the Montgomery method. As stated in the paper, the total number of iterations required to complete the multiplication process is eight cycles, and therefore it saves some hardware resources. The prototype of the architecture is implemented on the Virtex-7 FPGA board [32].

(Rose, et al., 2019): A DML multiplier which is capable of performing the mixed operation mode (i.e. a mixture of the static/dynamic mode) is proposed which promises to offer "better performance and energy trade-off" in comparison to the standard CMOS based designs. In fact, "the use of the dynamic mode for higher precision operations ensures higher performance as compared to the standard CMOS circuit (16% gain on average) at the cost of higher energy consumption". In comparison with standard CMOS implementation, the proposed DML's mixed-mode offers 15% of EDP improvement in a varied range supply voltage. A detailed PVT analysis is also carried out to ensure the performance at extreme conditions [33].

2.2 MULTIPLY AND ACCUMULATE UNIT

(Suzuki, et al., 1996): A FADD core is proposed in this design. The core has been fabricated in CMOS 0.5um technology. LZA technique is used for normalizing the numbers. HDL is used for the overall design [34].

(Pillai, et al., 2000): A floating-point low power multiply-accumulate unit is presented in this work. Transition activity and data path are simplified to reduce the power consumption. A 4 state FSM model is used to represent the switching activity. Due to data path simplification, the latency and delay are reduced [35].

(Natter, et al., 2000): A signed VHDL based MAC is proposed in this work. The design is also implemented on FPGA board. The proposed MAC algorithm uses "recursion formula in terms of new input-independent variables". The correctness of the proposed MAC is verified on MALAB and MAX plus II [36].

(William, et al., 2001): The technique proposed in the paper reduces the total number of partial product by a factor of two if applied to "signed-binay (SB) number". The work is also extended for FPGA hardware [37].

(Plessis, et al., 2002): Field Programmable Gate Arrays (FPGAs) are rapidly gaining popularity for signal processing applications. Multiplication, addition and Multiply-Accumulate (MAC) are the most important building blocks in signal processing. This paper will compare a number of structures to find the optimum configurations for minimum delay, size and cost in an FPGA [38].

(Huang, et al., 2002): A "novel limited resource scheduling (LRS) algorithm" based MAC for "DWT-processor" is proposed in this work. Given a set of architecture constraints and DWT parameters, the LRS algorithm can generate four scheduling matrices that drive the data path to perform the DWT computation, and the performance has also been investigated. Because the registers of FIR filtering are reused for the inter-octave storage, the MAClevel DWT architecture may require less extra inter-octave memory than the traditional architecture [39].

(Premkumar, et al., 2002): In this paper an alternative multiply accumulate units for the pulse shaping filters that use a new representation for their coefficients is proposed. Consequently, these new structures are fast, efficient and dissipate less power. The filters proposed take into account constraints, such as, inter symbol interference, response characteristics etc. in their design methodology [40].

(Tian, et al., 2002): In this paper, an algorithm of 32x32 multiply and MAC instructions' VLSI implementation with 32x8 multiplier-accumulator in DSP applications is presented. The 32x32 multiplication is achieved by 4 times 32x8 multiplication. The result of 32x8 multiplication serves as a partial product of the next 32x8 operation, when the result' of such four multiplication is accumulated, we get the result of 32x32. The 32x8 multiplication is only implemented by the hardware Booth multiplier [21[31. The algorithm of multiply and MAC instructions' implementation is the better trade-off between serial multiplier and, parallel multiplier [41].

(Liao, et al., 2002): A high-performance and low-power 32-bit multiply–accumulate unit (MAC) is described in this paper. In the proposed architecture, one-cycle throughput for 16-bit 16-bit and 32-bit 16-bit MAC instructions was achieved at very high frequencies. To handle media streams more efficiently, the single-instruction-multiple-data (SIMD) and the multiply-with-implicit-accumulate (MIA) features were added [42].

(Kao, et al., 2002): This research develops a theoretical model to predict how dynamic power and subthreshold power must be balanced to give an optimal operating point that minimizes total active power consumption for different workload and operating conditions. A 175-mV multiply-accumulate test chip using a triple-well technology with tunable supply and body bias values is measured to experimentally verify the tradeoffs between the various sources of power [43].

(Suvakovic, et al., 2003): A mechanism to minimize non-adiabatic dissipation in adiabatic circuit is explained in this research work. As stated, "the non-adiabatic dissipation is minimized by architectural design involving a small number of complex logic gates". For designing complex adiabatic gates "Ordered Binary Decision Diagrams (OBDD)" is used. Finally, an optimized architecture "for adiabatic parallel multipliers" is explained and its power consumption is also estimated [44].

(Shim, et al., 2003): This paper shows the usage of MAC in Very High-Speed Digital Subscriber line. A detailed analysis is also performed for DMT (Discrete Multitone) and SCM (Single-Carrier Modulation) used in VDSL (Very high-speed Digital Subscriber Line). The work is further extended to estimate the memory requirement for the proposed design in addition to conventional complexity measures [45].

(Li, et al., 2003): This paper describes a reconfigurable architecture of a high-performance pipelined 32-bit Multiply-Accumulate Unit (MAC). which is designed for a powerful embedded Digital Signal Processor (DSP). The proposed MAC unit can carry out two 16-bit multiplications in one clock cycle. The 32 × 16. 32 × 32. 32 × 16+80 and 32 × 32+80 operations can be implemented in two clock cycles. These characteristics allow the DSP being applied efficiently in different situations [46].

(Grossschadl, et al., 2003): A 32-bit MAC unit for RISC processor is presented in this research work. The proposed MAC unit can perform a variety of operations including (32 x 32)-bit signed/unsigned multiplication, (32 x 32+64)-hit signed/unsigned multiplication-accumulation, and (32 x 32+32+32)-bit multiplication-accumulation on unsigned integers [47].

(Kataeva, et al., 2005): Paper explains about RSFQ DS Processor, mainly used for removal of interferences from any signal. The author proposed MAC unit for floating point Multiplication-Addition. The Multiply-Accumulate unit comprises of three-unit namely parallel-multiplier, combiner and register or accumulator. The combiner evaluates the sum of the sums and the carries from M-MSB bits of the multiplier. The simulation is verified in VHDL [48].

(Bunyk, et al., 2005): Describes a MAC unit specific for programmable Band pass filtering. As explained in the paper, the clock frequency of the presented architecture is 20 GHz and it can perform 2.5 billion MAC instructions/sec. For doing such analysis, the data sample is considered to be of 7-bits and filter coefficient is considered of 16-bits which is arriving in bit-serial mode. The simulation is verified in VHDL. Basically, this MAC unit is application specific. It consists of a D flip flop (to act as a shift register), clocked AND gate and T flip flop for counting purposes [49].

(Cardoso, et al., 2005): In this work, minimization of Accumulator unit in MAC for block matching motion estimation is proposed. The FPGA implementation and mathematical models are discussed in this paper [50].

(Danysh, et al., 2005): This paper presents a “64-bit fixed-point vector MAC architecture capable of supporting multiple precisions”. The “vector MAC” has the ability to perform “one 64 × 64, two 32 × 32, four 16 × 16, or eight 8 × 8-bit signed/unsigned multiply accumulates” using fundamentally the same hardware as a scalar 64-bit MAC and with only a slight increment in delay. The proposed design is implemented using Verilog HDL in Synopsys tool [51].

(Vangal, et al., 2006): A “pipelined single-precision Floating-Point Multiply Accumulator (FPMAC)” consisting of accumulator in radix-32 and internal carry-save addition is explained in this research work. Additionally, an improved version of “Leading-Zero Anticipator (LZA) and overflow prediction logic” required in carry-save addition is also explained [52].

(Kataeva, et al., 2007): A “RSFQ digital signal processor design based on hybrid RSFQ-CMOS memory” is proposed in this paper. The DSP consists of an “RSFQ multiply-accumulate Unit, memory caches and synchronization block, partitioned into multiple chips, and a large CMOS memory”. The MAC unit is shown as an internal and essential unit in the RSFQ architecture [53].

(Voronenko, et al., 2007): This work provides an algorithm for fused multiply accumulate instruction. In this paper, a generalized procedure to alter any transform algorithm into an FMA algorithm is explained [54].

(Abdelgawad, et al., 2007): In this work 8-bit, 16 bit and 32-bit MAC is proposed and implemented on Xilinx ISE and on FPGA board. The design shows improvement in area and power. 4:2 compressor circuits are used to make the multiplier circuit faster [1].

(Xia, et al., 2009): The novel design is implemented in ModelSim in TSMC 90 nm CMOS technology. Here “4-pipelined high-performance split Multiply-Accumulator (MAC)” architecture is proposed. In order to achieve higher speed, a novel partial product compression technique using interleaved adders and a “modified hybrid Partial-Product-Reduction-Tree (PPRT)” is also proposed. As stated by the author, the proposed MAC can perform “1-way 32-bit, 4-way 16-bit signed/unsigned multiply or multiply-accumulate operations and 2-way Parallel Multiply Add (PMADD) operations” [10].

(Shanthala, et al., 2009): In this research work an 8-bit MAC unit is proposed using Cadence Virtuoso 180nm Technology. Clock gating scheme is used to optimize the power consumption [14].

(Shanthala, et al., 2009): In this research work an 8-bit pipelined MAC unit is proposed using Cadence Virtuoso 180nm Technology. Various adder/multiplier circuits are compared and implemented for the MAC [55].

(Hoang, et al., 2010): A Multiply-Accumulate (MAC) architecture which can operate on 2's complement numbers are explained in this paper. The author claims that the proposed architecture is a high-speed and power-efficient MAC which uses "accumulation guard bits and saturation circuitry". The implementation is done basically on VHDL and designed in 65 nm 1.1V cell library [56].

(Quan, et al., 2010): This paper presents a 32-bit vector multiply-accumulate (MAC) architecture capable of supporting multiple precisions. The vector MAC can perform one 32×32, one 32×16, two 16×16, four 8×8 bit signed/unsigned multiply-accumulate using Booth encoding algorithm and Wallace tree compressing. A reconfigurable Booth encoding array is implemented using 8×8 Booth unit as the basic element, and longer bit modes are obtained by combining these elements selectively. This MAC unit can also perform multiply between scalar and vector operands [57].

(Jain, et al., 2010): This paper describes energy efficient and reconfigurable fused/continuous Multiply-Accumulator (MAC) architecture for single-precision Floating-point and 16-bit signed integer operands. This eight-stage pipelined and single-cycle throughput MAC design contains a bit level pipelined multiplier, followed by fast sparse-tree adder and single cycle accumulator loop with delayed normalization logic [58].

(Hsieh, et al., 2011): In this paper, an APC (Adaptive Power Control) system is proposed which performs on power gated circuitries. The proposed architecture is tested using a standard MAC fabricated in UMC 90 nm standard CMOS process. The basic implementation is done on RTL compiler [2].

(Kouretas, et al., 2012): A novel low-power approach to perform addition/subtraction in LNS (Logarithmic Number System) is explained in this research work. The paper also explains the impact of such low power addition/subtraction circuit used in LNS on digital filter VLSI implementation. The implementation is done in UMC 90 nm standard CMOS process [59].

(Esmaeili, et al., 2012): A “Low-Swing Differential Conditional Capturing Flip-Flop (LS-DCCFF)” is presented in this work. The flip flop explained in this work is capable of operate in a low swing LC resonant clocking scheme and utilizes reduced swing inverters at the clock input. The verification of the operation is done using LS-DCCFF in a dual-mode MAC. The dual-mode MAC is fabricated in TSMC 90 nm CMOS technology. Here, the optimization of the MAC unit is not performed but a technique to improve the performance of the MAC using LS-DCCFF is explained [60].

(Deepak, et al., 2012): In this work a novel multiplier circuit is proposed using which a MAC unit is designed. Cadence NC Sim and RTL compiler are used for doing all these analyses [6].

(Maechler, et al., 2012): VLSI based architecture is proposed based on MAC. Basically, this paper shows the importance of MAC as its application [61].

(Zhang, et al., 2012): A pipelined architecture for discrete wavelet transform is presented. The objective of this study is to design a high-speed VLSI architecture which has a high operating frequency with smaller clock periods. The architecture also achieves an efficient utilization of the hardware by increasing the inter as well as intra-stage computational parallelism for effective usage of pipelining [62].

(Mooney, et al., 2013): An “ASIP (Application-Specific Instruction-set Processor)” is designed, implemented, and evaluated in this research work. The proposed dual MAC is implemented on FPGA and its performances are evaluated in a “closed-loop power converter system”. A dual MAC Data Path is also proposed in this design [63].

(Marr, et al., 2013): A Statistical analysis of computations/ unit energy in different processor over a period of 30 years is performed in this paper. The analysis shows that the energy efficiency improvement rate has declined sharply in the recent a few years. An energy efficient asynchronous pipeline technique is presented in this work [64].

(Jagadees, et al., 2013): In this work a novel multiplier circuit is proposed using which a MAC unit is designed for 64-bit input. The overall MAC unit design operates at a frequency of 217 MHz. The overall power dissipation found to be as 177.732 mW [7].

(Abdelgawad, 2013): In this research work an ASIC implementation of a 32-bit MAC unit is proposed, which reduces the requirement of 5.5% of the total area, 9% of the power, and 13% of the delay compared to the conventional MAC unit. The simulation is done 0.18um CMOS technology using HDL [65].

(Francis, et al., 2013): In this work a modified Braun Multiplier is used with bypassing technique to design the overall MAC. Designs are implemented in 0.13um CMOS technology. TG, DPL etcetera logics are used to design the full adders in the circuit [8].

(Amaricai, et al., 2014): A "Floating-point multiply-add fused architecture" for IEEE 16-bit or IEEE 32-bit (half precision or single precision respectively) is discussed. The architecture is designed by amalgamation of the multiplication and addition/subtraction blocks required for mantissa data calculation in a single operation. This has provided an efficient usage of DSP blocks in Field Programmable Gate Arrays (FPGAs). The architecture is also implemented on FPGA [66].

(Warrier, et al., 2014): A Baugh-Wooley algorithm based pipelined MAC architecture using a 16x16 bit multiplier is proposed. The Clock gating technique is also used at the idle pipeline stages to reduce the power consumption. The author claims that, the proposed architecture consumes 30-80% lesser power than the conventional MAC architectures. At the end various MAC units available in the literature are compared. The implementation is done in 65nm CMOS using HDL in TSMC library [9].

(Burg, et al., 2014): A novel architecture for adaptive systems is presented in this paper. The architecture mainly stresses upon the systems whose exact specifications are not known. Here a Walsh-based architecture model is proposed which is better than

MAC based architecture. But the Walsh-based architecture needed a vector table from which it refers its values. So basically, it is kind of look up table technique [67].

(Ahish, et al., 2015): A “partial product reduction block” is proposed in the work, which is used for optimizing the area, power and delay of the multiplier used. The partial product reduction block uses different multi-bit adder row wise instead of the conventional adder which performs column wise. The proposed technique has reduced the delay power and area by 46%, 39% and 17% [68].

(Akbarzadeh, et al., 2015): A modified pipelined modulo $2^n + 1$ modified booth multiplier is proposed. The design is further extended for implementing a modulo $2^n + 1$ MAC architecture. The CMOS transistor level implementation of multiplier as well as MAC has shown significant improvement in power and PDP [69].

(Chen, et al., 2015): A compact architecture for performing MAC operation for “PWM signals”. The presented architecture consists of a “dual scale counter and a 2D MAC operator”. The proposed “2D MAC” operator is compared with the MAC operator from the FPGA IP which has an 8-bit resolution. The result reveals that 2D-MAC reduces the chip area with comparable power than FPGA IP [70].

(Cini, et al., 2015): In this research, a MAC unit is proposed which is suitable for “6-input LUT” based FPGAs. No pipelining structure is deployed as the design uses “(6,3) counters” in partial product reduction. The proposed MAC takes 16x16 bit input and produces 40-bit output which has sign extended bit. Significant improvement is reported when the proposed MAC is equated with the traditional MAC algorithms and redundant carry save architectures [71].

(Gerlach, et al., 2015): The proposed work explains a real and “complex valued MAC” which uses same amount of multiplier as it is been used for implementing “complex valued SIMD MAC” and butterfly operation. The proposed architecture is evaluated in terms of power, area and performance [72].

(Kumar, et al., 2015): A “novel FPMAC” is proposed in this work which works with optimal computation to make it faster. The propose design promises for lesser power consumption. Proposed architecture is implemented in Xilinx ISE (14.5) and synthesized using CMOS 90 nm technology library using Synopsys Design Compiler [73].

(Priya, et al., 2015): This research work evaluates 3 MAC architectures consisting of array, booth and Wallace tree multiplier which leads to an incorporation in PID controller. The simulation is performed in Model SIM and it is synthesized in Xilinx ISE. The result suggests that the MAC unit with Wallace tree consumes lesser power and area [74].

(Narasimhan, et al., 2015): An “optimized co-processor unit”, targeting specifically for Digital Signal processing application is presented in this work. The co-processor hardware consists of MAC unit, control unit and a 32-bit output accumulator as the leading operative blocks. Vedic as well as booth multiplier is used for designing the proposed MAC architecture. The MAC unit takes two 16-bit inputs or one 32-bit input and produces one 32-bit output [75].

(DeBrunner, et al., 2015): For FIR filter implementation a fused MAC unit is developed which truncated multiplication techniques which uses the accumulation technique. As because of truncated multiplier, the power and area are reduced. Different types of truncated multiplication approaches have been presented in this study [76].

(Basiri, et al., 2015): In this paper, a floating-point MAC circuit is used to design the 2nd order IIR filters and thereafter the 2nd order IIR filter is used rigorously to design a configurable 6th order IIR filter. The 6th order IIR filter is used to perform “one 6th order or three 2nd order or one 4th order and one 2nd order IIR filter operations in parallel”. The performance of the proposed 6th order IIR filter is evaluated in CMOS 45 nm technology and the result shows that the proposed 6th order IIR filter requires 58.4% less power than conventional MAC based architecture [77].

(Nandal, et al., 2015): A series of LUT is used in the place of MAC in the proposed work. A technique called “Distributed Arithmetic (DA)” is used. The FPGA based implementation of FIR filter is also discussed in this work. A parallel FIR digital filter is used for high-speed and low-power operations. The DA technique calculates the partial products without using a conventional multiplier for fixed-point number. The analysis on the proposed architecture shows a high-speed and low-power design. The proposed filter is implemented in VHDL. The proposed method has reduced the number of LUT used by 60%, occupied slices by 40% and number of gates by 50% [78].

(Anitha, et al., 2015): In this work Vedic multiplier and reversible logics are implemented. Using these finally 32-bit MAC architecture has been designed. The implementations are done using Verilog HDL in Cadence RTL. Not implemented for signed fixed/floating point number [79].

(Karthikeyan, et al., 2016): A modified full adder is used in the research work which reduces the power and area requirements. For estimating the power, CNTFET technology is used in HSPICE simulation. According to the author, "a model is developed for nanoscale devices and circuits, including both CMOS technology and CNTFET technology with the aim of guiding nanoscale device and circuit design". The new design offers large device speed than conventional designs [80].

(Babu, et al., 2016): A low power high through put architecture is proposed in this work. Fixed point implementation has been done for signed number. The design has been implemented in Cadence Virtuoso 90 nm technology [5].

(Dhindsa, et al., 2016): The core design units of Multiply-Accumulate architecture are optimized for energy-efficient architecture design using clock gating scheme is presented in this work. Moreover, the MAC unit is designed with synchronization to work in single clock cycle due to which the overall speed of operation has enhanced. The implemented design in Cadence Virtuoso as well as NCSim using 90 nm CMOS technology. Finally, the design in analog platform and digital platform is compared and the result shows that the digital approach of the design offers six times more power consumption than in analog design environment [81].

(Garland, et al., 2017): A MAC unit that uses weight-sharing CNNs is explained in this research work. A binning approach is used where a counter counts the frequency of each weight and place it in a bin. The accumulated value is multiplied thereafter. The hardware requirement for multiplier is reduced as the adders and selection logic replaces the multiplier. The detailed comparison shows that the presented architecture requires lesser area and lesser power comparison [82].

(Jeon, et al., 2017): A novel architecture called “HMC-MAC” is presented in this paper. As the name suggests, a MAC architecture is implemented in the HMC. As stated by the author, “a conventional HMC works independently to maximize the parallelism, and HMC-MAC is based on the conventional HMC without modifying the architecture much. Therefore, a large number of MAC operations can be processed in parallel” [83].

(Ananthalakshmi, et al., 2017): A novel “reversible floating point fused arithmetic unit architecture” is proposed in this work. The proposed architecture is also satisfying “IEEE 754 standard”. Adiabatic logic technique along with reversible logic styles offers a power efficient proposed design. In the proposed design the hardware is reduced and latency is improved by employing fused elements and decomposing the operands in the realization respectively. To test the operation of the proposed design FFT and FIR filter are realized which the key requirements in Digital Signal Processors. The result shows that the proposed architecture utilizes a smaller number of gates, requires less quantum cost and produces lesser number of garbage output at low latency [84].

(Kamp, et al.,, 2018): Design optimization for Complex Multiply Accumulate Cell (CMAC) are presented in this research work. A novel signaling technique is used to converts a complex multiplication into single integer multiplication. The FPGA based implementation is done on Xilinx ultarscale+ which promises to save power and therefore the cost [85].

(Lv, et al., 2018): An architecture required in modern FPGA is presented in this research study where a customized 32-bit floating point data is used. The 32-bit data is used for multiplication and accumulation. The customized 32-bit floating point data representation is compared with 32-bit IEEE standard [86].

(Zhang, et al., 2018): A fixed/floating point MAC unit is proposed in this research work which can be applied for the applications such as deep learning algorithm. The said architecture supports 16-bit floating point multiplication (half precision) and 32-

bit accumulation (single precision). The presented architecture requires 4.6% more area than a half-precision MAC unit. The implementation is done using VHDL [87].

(Chen, et al., 2018): This paper implemented RPR-MAC. The paper also significantly proves that signed-integer-multiplication in 2´s complement format can make RPR much more efficient. Signed integer multiplication is further extended for MAC operation by “proposing RPR implementations” that improve the “error correction capabilities with a limited impact on circuit overhead”. The tested result of the proposed design shows that the Mean Square Error can be significantly reduced by using this technique [88].

(Ryu, et al., 2018): A “pipelining method” that eradicates some of the flip-flops for designing a MAC is proposed. In machine learning accelerator operations, MAC processing plays a vital role. A pipeline structure always helps in reducing the “length of the critical paths”. At the same time, to increase the pipelining, the flip-flop count must be increased which, consequently increase the area and power consumption. The result shows that the proposed MAC architecture requires 20% lesser power and area each than the conventional pipelined MAC [89].

(Patil, et al., 2018): In this research paper, a “radix-4 booth multiplier-based MAC unit” is proposed which improvise the delay of the MAC unit. (6,3) counter is used for reduction of the partial products. The proposed MAC unit takes 16X16 bit input produces 40-bit output. The proposed MAC is simulated in Xilinx ISE and implemented in Spartan-6 FPGA board [90].

(Patil, et al., 2019): In this review paper, a comparison study is performed on MAC unit based on different kinds of multipliers and adders. The functionality of the multiplier is to produce the result based on the multiplication of the inputs whereas, the adder unit sum up the current product with the previous result. The study gives a broader picture regarding speed of operation and power consumption of different MAC architecture available in the literature [91].

(Camus, et al., 2019): A comparison is performed for run-time configurable MAC units. The circuits are synthesized in a 28nm CMOS technology. The comparison is performed in terms of power and throughput in order to identify the optimized architecture for neural network [92].

(Zhang, et al., 2019): A MAC unit using the "posit number format" in deep learning application is presented in this paper. Additionally, a "posit MAC unit generator" is written in C language. A detailed analysis is performed for area, delay and power in ST Microelectronics 28 nm technology with varied bit width [93].

(Senthilkumar, et al., 2019): A discrete wavelet transforms which can be used in the field of biomedical signal processing is implemented using Vedic mathematics. Instead of using CMOS, FinFET and CNTFET technologies are used in this architecture. The basic architecture of DWT architecture requires adder block, multiplier block, MAC block and additionally, in-order store the co-efficient, RAM or ROM blocks. The core of the SOC is designed using Vedic mathematics sutras. The usage of CNTFET has reduces the power consumption by 95% [94].

(Tung, et al., 2020): In this paper, we propose a low-power high-speed pipeline multiply-accumulate (MAC) architecture. In the proposed MAC architecture, the addition and accumulation of higher significance bits are not performed until the PPR process of the next multiplication. To correctly deal with the overflow in the PPR process, a small-size adder is designed to accumulate the total number of carries [95].

(Nahmias, et al., 2020): In this research paper several proposed tunable photonic MAC systems are discussed, and provide a concrete comparison between deep learning and photonic hardware using several empirically validated device and system models. It also shows significant potential improvements over digital electronics in energy, speed, and compute density [96].

(Zhang, et al., 2020): In this paper, a new flexible multiple-precision multiply-accumulate (MAC) unit is proposed for deep neural network training and inference. The proposed MAC unit supports both fixed-point operations and floating-point operations. For floating-point format, the proposed unit supports one 16-bit MAC operation or sum of two 8-bit multiplications plus a 16-bit addend. Verilog HDL is used for designing the overall MAC architecture [97].

2.3 MOTIVATION & TECHNICAL GAP

MAC unit performs the essential mathematical operations in the digital signal processing systems. Since the MAC unit speed decides the DSP's speed, the primary consideration of the research done in recent times has focused mainly to enhance the speed of the MAC unit. Also, as the DSPs are inevitable in portable electronics, a constraint on power consumption forces to optimize energy efficiency. Therefore, power dissipation is another primary concern in the MAC operation. Hence, from the detailed literature review it can be summarized that:

1. As discussed earlier in this chapter, the multiplier and accumulator are the critical components of MAC architecture. As the efficiency of the MAC is dependent upon the efficiency of the multiplier (mainly), an efficient multiplier (in terms of typical design constraints) design can further improvise the efficiency of a MAC unit. Moreover, the multiplier proposed in the literature are mostly based on the Wallace tree algorithm. It is claimed that the multipliers based on the Wallace tree reduce the steps involved to add the partial products. Still, it uses half adder or full adder for the addition of partial products which increases the complexity of the circuit [4, 12, 15, 18, 23, 24, 26-28, 30]. On the modified Wallace tree multiplier proposed in the literature uses compressor-based circuits (up to 7:3 only) to reduce the steps involved to add the partial products. Therefore, if a universal compressor (N:M of any size) is applied to the multiplier for the addition of partial products, it can further improve the efficiency.
2. Any electronic circuit can be designed by two different approaches, namely the top-down approach and the bottom-up approach. In the top-down approach, the designs are implemented by focusing mainly on the output efficiency of the overall design. i.e. importance is given on the implementation of the process or algorithm, not on the optimization of primary cells. On the other hand, in the case of the bottom-up approach, the whole digital architecture is designed starting from its primary cell, i.e. importance is given on optimization of the primary cell as well as on the practical implementation of the algorithm. In the literature, full custom circuit design for the MAC unit has never been proposed & most of the available

architectures in the literature have used HDL based approach [7, 56, 60, 69, 75, 87]. Moreover, almost 99% (80 out of 81 papers) of the architectures available in the literature have neither implemented for signed operation nor floating-point designs. Therefore, the practical applicability of such design needs to be further tested. Hence a full custom IC approach can provide a much more efficient MAC in terms of power as well as delay.

3. Synchronization, clock gating techniques and pipelining can further enhance the speed of operation and minimize the power consumption [48, 81]. Simultaneously all these techniques are neither adopted nor described for any of the MAC explained in the literature. Though some architectures in the literature have used the clocking signals for the accumulation of data only (in the register or accumulator), most of the architectures haven't used any clocking signal. Any circuit in asynchronous mode can't be implemented in a real-time application.

2.4 OBJECTIVE OF THE RESEARCH

Delay and power optimization are very much essential for any kind of digital circuits. As the MAC unit is the heart of a DSP, it is always demanding to use an efficient MAC architecture. In this research work, the focus is given on the optimization of the basic building blocks. Based on the technical gap identified, the objectives of this proposed research work are defined as:

I. To design & implement a novel multiplier architecture and analyzing its performance using Cadence Virtuoso 90 nm Technology.
II. To design a novel 8 × 8 bit signed/unsigned MAC architecture for fixed-point numbers using Cadence Virtuoso 90 nm Technology.
III. To design & analyze a novel 8 × 8 bit signed/unsigned synchronous MAC architecture using clock gating scheme for fixed-point numbers using Cadence Virtuoso 90 nm Technology.
IV. To design a novel 8 × 8 bit signed/unsigned MAC architecture for floating-point numbers using Cadence Virtuoso 90 nm Technology.
V. To design & analyze a novel 8 × 8 bit signed/unsigned synchronous MAC architecture using clock gating scheme for floating-point numbers using Cadence Virtuoso 90 nm Technology.

CHAPTER 3: UCM-A NOVEL APPROACH FOR DELAY OPTIMIZATION

3.1 INTRODUCTION

Multiplication has a vast field of applications such as digital signal processing, multimedia systems, arithmetic operation, digital communication, etcetera. The process of the multiplication can be segregated into two categories, namely “partial product generator” and “final sum/carry generator using adder circuits”. Therefore, the multiplication process requires more hardware resources and processing time in comparison to the primary adder/subtractor circuit. In a simplified view, a multiplier requires AND gates (for partial product generation) and adder circuits (half adders and full adders) for the addition of partial products to yield the final result. Figure 3.1 shows the simplified operation of a multiplier. As per the literature, various multiplier algorithms/architectures are proposed in the past, such as booth encoder, Wallace tree adder, array multiplier, modified booth multiplier, etcetera [4]. All these algorithms/architectures use different approaches to make the multiplier operation more efficient. For example, booth multiplier or modified booth multipliers are algorithmic approaches where the main focus is on reducing the total number of partial products. On the other hand, as explained in [4], the efficient addition of the partial products is the key advantage in Wallace tree multiplier. Hence a combination of both can provide a better result.

There are various multiplier circuits explained in the literature, which mainly focuses on the issues of power consumption, delay of the multiplier circuit, and lesser area [11-13, 15-18, 20-24, 26-28, 30, 53]. But as per studies, it is found that area and the speed of operation are the two most conflicting design constraints. Hence increasing the speed of operation enhances the area requirement. On the other hand, as day by day, the size of the transistor is decreasing, the area cannot become a significant issue in today's digital systems. The power consumption and delay of a particular circuit depends upon the supply voltage (V_{DD}). A slight increment in the supply voltage increases the overall power consumption, but at the same time, it decreases the delay of the circuit. Hence there is always a trade-off between power consumption and delay of a circuit. Therefore, the supply voltage plays a vital role in designing a low power circuit. I.e., for a low power design, an optimized supply voltage is needed to be chosen so that the output logic is valid, and the power consumption is bare minimum with a comparable delay value. As per the literature survey, it is found that most of the multiplier design uses Wallace tree multiplier as the underlying algorithm and in the majority of the cases, the basic Wallace tree multiplier algorithm has been modified to get better results [4, 12, 15, 18, 26, 30]. The reason for the same is that the Wallace tree algorithm is the simplest way of designing multiplier with optimized delay/power consumption.

$$
\begin{array}{rrrrrrrrr}
 & & & & \mathbf{A_N} & \cdots & \mathbf{A_2} & \mathbf{A_1} & \mathbf{A_0} \\
 & & & \times & \mathbf{B_N} & \cdots & \mathbf{B_2} & \mathbf{B_1} & \mathbf{B_0} \\
\hline
 & & & & A_NB_0 & \cdots & A_2B_0 & A_1B_0 & A_0B_0 \\
 & & & A_NB_1 & \cdots & A_2B_1 & A_1B_1 & A_0B_1 & \\
 & & A_NB_2 & \cdots & A_2B_2 & A_1B_2 & A_0B_2 & & \\
 & & \vdots & \vdots & \vdots & & & & \\
A_NB_N & \cdots & A_2B_N & A_1B_N & A_0B_N & & & & \\
\hline
\mathbf{P_N} & \cdots & \cdots & \cdots & \mathbf{P_4} & \mathbf{P_3} & \mathbf{P_2} & \mathbf{P_1} & \mathbf{P_0}
\end{array}
$$

Figure 3.1: Basic multiplication operation

In this chapter, a high-speed multiplier architecture with a minimal value of supply voltage is proposed. In the implemented architecture, the supply voltage is minimized to reduce the power consumption of the circuit without compromising the speed of the multiplier circuit. The study mainly focusses on the optimization of the partial product addition. The reason behind the same is that, for partial product generation, the booth algorithm produces a better result than any other multiplication approach. Secondly, as discussed above, the majority of the multipliers use Wallace tree adder for partial product addition. Hence, an optimized and efficient partial product adder, which can replace the Wallace tree algorithm, can yield a better multiplier.

3.2 WALLACE TREE MULTIPLIER ARCHITECTURE

The conventional Wallace tree multiplier algorithm is divided into three stages:

Stage 1: partial-product generation.
Stage 2: addition of partial products which creates `sum' and `carry' terms separately.
Stage 3: a final adder, which is generally a fast adder to add the ‘sum’ terms and ‘carry’ terms together to yield the final result [27].

In stage 1, the partial products are the AND product of each multiplier bit with each multiplicand bit. It can be implemented either by using conventional two-input AND gate to find the partial product of each multiplicand and multiplier or by using advanced booth multiplier to reduce the total number of partial products. With the help of 2nd order booth algorithm, the number of the partial product is reduced to half (approx.) of the bit width of the multiplier [15].

In stage 2, the partial products are added using half adder/full adder. The partial products with `N' rows are grouped in sets of three rows each. Any rows that are not part of the group of three rows are transferred to the next level without any modification. In the groups of three rows, full adders are applied to the columns containing three partial products, and half adders are applied to the columns containing two partial products (in the groups of two rows) [13]. The columns with only one partial product are transferred to the next level without any modification. For the next level calculation, use the sum and carry output of the full adder/half adder of the previous level along with the remaining partial products. The same procedure is followed until and unless there are only two rows left.

In stage 3, the remaining two rows are added either by using an n-bit RCA or by using a fast adder such as carry look-ahead adder, carry select adder, etcetera. Figure 3.2 elaborates the operation of the Wallace tree multiplier algorithm in detail, where 'a0'-'a8' are representing the multiplicands; 'b0'-'b8' are representing the multipliers; 'q0'-

'q80' are serving the partial products; 'S_{xx}' is representing the sum; 'Cxx' is representing the carry outputs of half adder/full adder and 'CR_x' is serving the ripple carries at the final stage. Moreover, as shown in figure 3.2, the rectangles with three variables represent full adder, and the rectangles with two variables represent half adder.

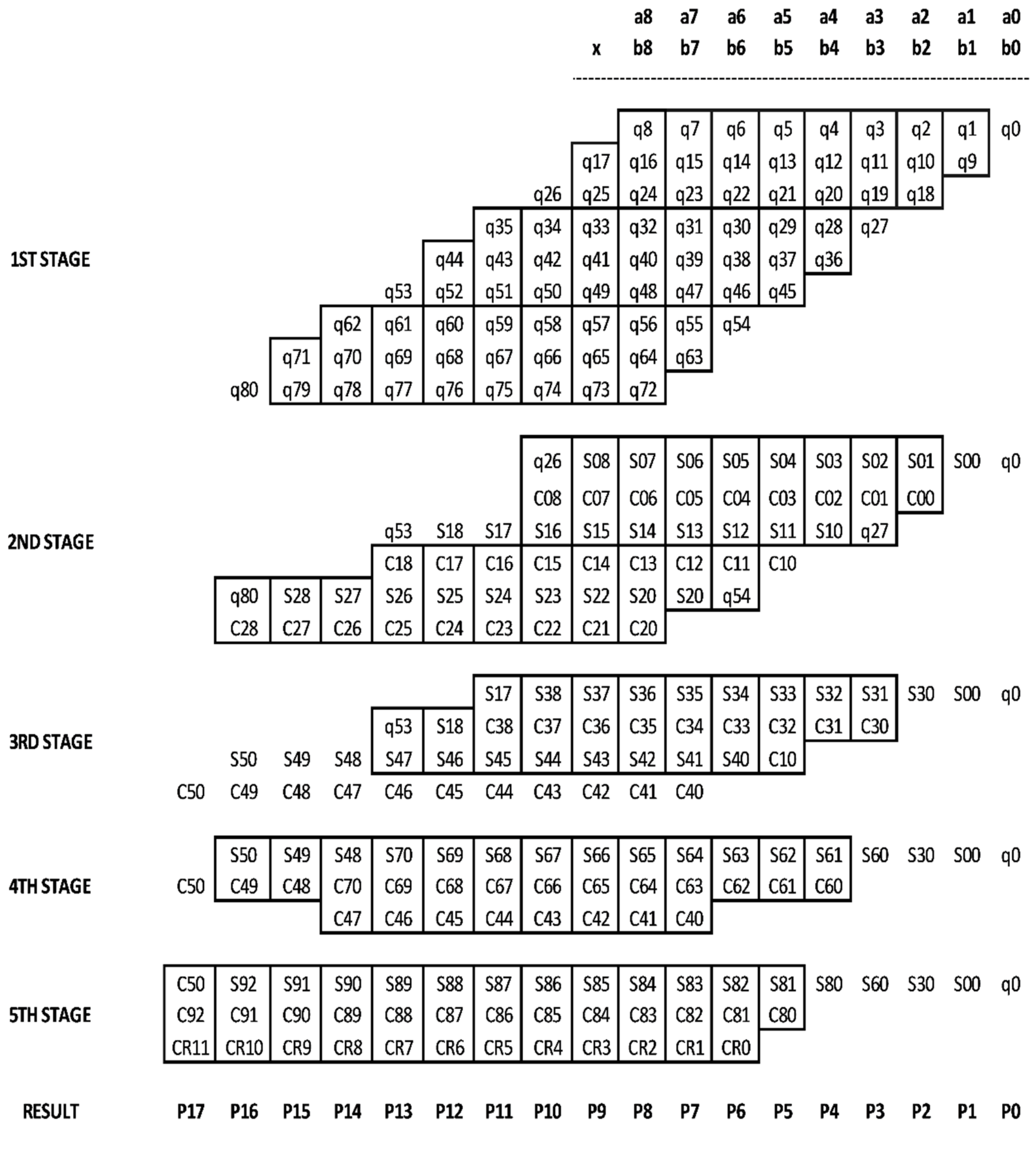

1ST STAGE
2ND STAGE
3RD STAGE
4TH STAGE
5TH STAGE
RESULT
a8 a7 a6 a5 a4 a3 a2 a1 a0
x b8 b7 b6 b5 b4 b3 b2 b1 b0
P17 P16 P15 P14 P13 P12 P11 P10 P9 P8 P7 P6 P5 P4 P3 P2 P1 P0

Figure 3.2: Wallace tree multiplier for 9 × 9 bit multiplication

3.3 DESIGN PROCESS OF UCM ARCHITECTURE

A universal N:M bit compressor-based multiplier is proposed in this research work. Where ‘N’ and ‘M’ are the number of inputs and outputs respectively. The process flow of the proposed UCM design is shown in figure 3.3. As shown in figure 3.3, the novel architecture is designed in Cadence Virtuoso 90 nm CMOS technology as well as in Verilog HDL. The power and delay analysis are carried out from virtuoso-based design, whereas the Verilog HDL program is used for FPGA prototyping.

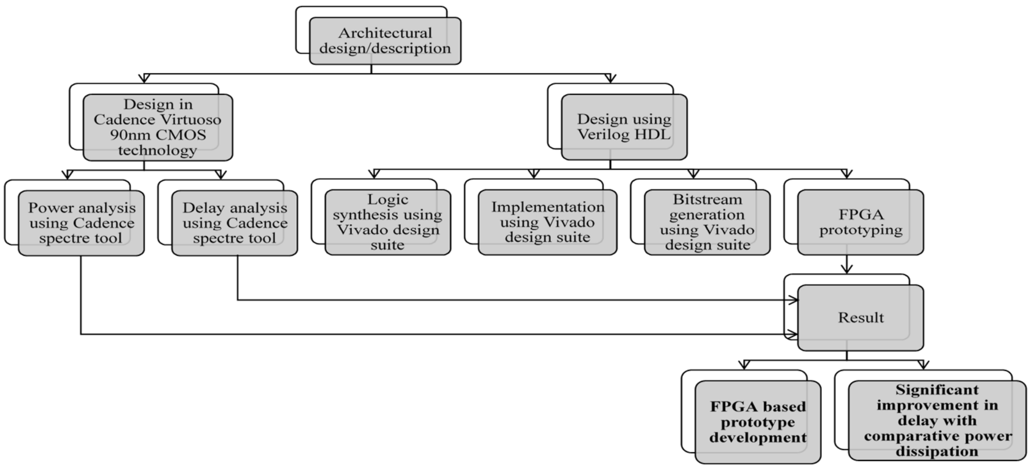
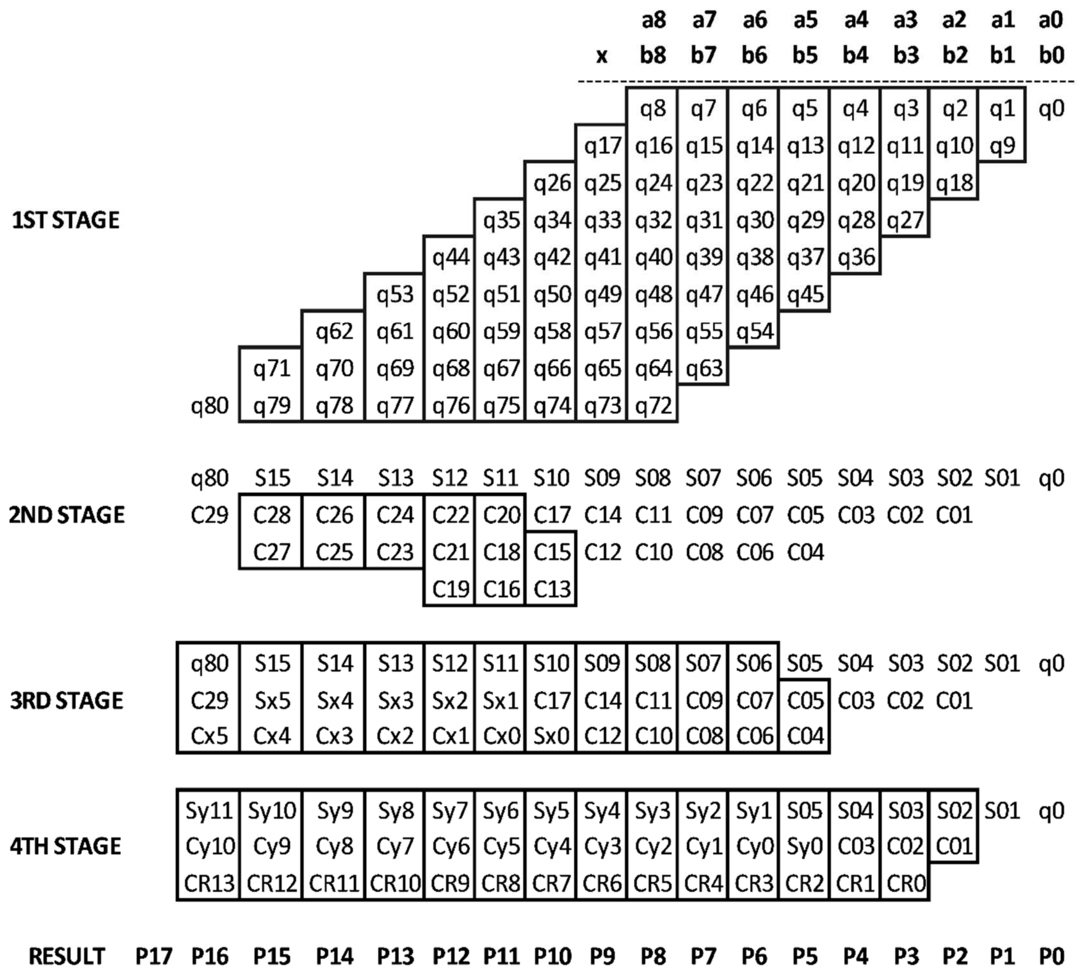

Architectural design/description
Design in Cadence Virtuoso 90nm CMOS technology
Design using Verilog HDL
Power analysis using Cadence spectre tool
Delay analysis using Cadence spectre tool
Logic synthesis using Vivado design suite
Implementation using Vivado design suite
Bitstream generation using Vivado design suite
FPGA prototyping
Result
FPGA based prototype development
Significant improvement in delay with comparative power dissipation

Figure 3.3: UCM architecture design process flow

3.3.1 UCM architecture

Although the Wallace tree multiplier is much faster than the array multiplier [30], it requires a large number of adders. Secondly, the Wallace tree multiplier is highly irregular and complicated. So, to overcome the irregular structure, several modified Wallace tree algorithms are proposed in the literature [4, 12, 15, 18, 23, 24, 26-28, 30]. All these multiplier algorithms are based upon Wallace tree algorithms. Hence replacing the Wallace tree algorithm may further improve the result of the multiplier. Another critical point here is, instead of using traditional Wallace tree adder, compressor circuits such as 3:2 compressors or 4:2 compressors, etcetera can be used for partial product addition. But as there is a possibility of using the same compressor again and again for doing addition (same as Wallace tree addition), the same wouldn't be much useful. The UCM architecture is designed as shown in figure 3.4, where the rectangles with three variables represent full adder, the rectangles with two variables represent half adder, and the rectangles with more than three variables represents a compressor circuit. The architecture of UCM is composed of three stages. The stage 1 and stage 3 of the novel UCM architecture remain the same as that of the Wallace tree algorithm, since whether it is partial product generation or the addition of intermediate ‘sum’ or ‘carry’ terms using a simple adder these can be selected according to the designer's requirement. Therefore, it is more critical to substitute stage 2, i.e., the addition of partial product, which separately produces ‘sum’ and ‘carry’ terms.



a8 a7 a6 a5 a4 a3 a2 a1 a0
x b8 b7 b6 b5 b4 b3 b2 b1 b0
q8 q7 q6 q5 q4 q3 q2 q1 q0
q17 q16 q15 q14 q13 q12 q11 q10 q9
q26 q25 q24 q23 q22 q21 q20 q19 q18
1ST STAGE
q35 q34 q33 q32 q31 q30 q29 q28 q27
q44 q43 q42 q41 q40 q39 q38 q37 q36
q53 q52 q51 q50 q49 q48 q47 q46 q45
q62 q61 q60 q59 q58 q57 q56 q55 q54
q71 q70 q69 q68 q67 q66 q65 q64 q63
q80 q79 q78 q77 q76 q75 q74 q73 q72
q80 S15 S14 S13 S12 S11 S10 S09 S08 S07 S06 S05 S04 S03 S02 S01 q0
2ND STAGE
C29 C28 C26 C24 C22 C20 C17 C14 C11 C09 C07 C05 C03 C02 C01
C27 C25 C23 C21 C18 C15 C12 C10 C08 C06 C04
C19 C16 C13
q80 S15 S14 S13 S12 S11 S10 S09 S08 S07 S06 S05 S04 S03 S02 S01 q0
3RD STAGE
C29 Sx5 Sx4 Sx3 Sx2 Sx1 C17 C14 C11 C09 C07 C05 C03 C02 C01
Cx5 Cx4 Cx3 Cx2 Cx1 Cx0 Sx0 C12 C10 C08 C06 C04
Sy11 Sy10 Sy9 Sy8 Sy7 Sy6 Sy5 Sy4 Sy3 Sy2 Sy1 S05 S04 S03 S02 S01 q0
4TH STAGE
Cy10 Cy9 Cy8 Cy7 Cy6 Cy5 Cy4 Cy3 Cy2 Cy1 Cy0 Sy0 C03 C02 C01
CR13 CR12 CR11 CR10 CR9 CR8 CR7 CR6 CR5 CR4 CR3 CR2 CR1 CR0
RESULT P17 P16 P15 P14 P13 P12 P11 P10 P9 P8 P7 P6 P5 P4 P3 P2 P1 P0

Figure 3.4: UCM architecture for 9 × 9 bit multiplication

3.3.2 Addition of partial products

While adding partial products, the partial products are arranged in such a way that the summation of multiplicand and multiplier's bit position is identical. The summation of the location of the bit can be called a 'weight' of a specific partial product. For example, in figure 3.4, 'q35', 'q43', 'q51', 'q59', 'q67' and 'q75' are aligned in a single column because of the fact that the weight for all of the partial products mentioned is eleven, i.e. 'q35'='a8'.'b3', 'q43'='a7'.'b4', 'q51'='a6'.'b5' etcetera. Thus, the summation of the bit position is either 8 + 3 or 7 + 4 or 6 + 5, which is equal to 11 in all situations. Hence, its alignment is critical for the addition of partial products. Once the partial products are properly aligned, the next move is to add all of the partial products that fall into that specific group. At first, the total number of stages and levels need to be determined for adding a specific column. Each stage consists of a pair of AND-XOR gates, and the total number of stages is counted from top to bottom in one level. The total number of first level stages is 'i-1', where 'i' is the total number of partial products to be added in a specific column.

On the other side, the horizontal AND-XOR pair count is the total number of levels needed for the design. From a different angle, it can be found that the total number of levels required in a design is the total number of AND-XOR pairs provided in the bottom-most stages, i.e., the number of AND-XOR pairs through right to left. In each level, the total number of stages required is decremented by one from its preceding level's total number of stages. The total number of levels 'n' needed in a specific column for 'i' number of partial products is given by equation 3.1 and 3.2.

$$2^n - 1 \geq i \tag{3.1}$$

$$\Rightarrow 2^n \geq i+1$$

$$\Rightarrow n(\log_{10}2) \geq \log_{10}(i+1)$$

$$\therefore n \geq \frac{\log_{10}(i+1)}{\log_{10}2}$$

$$or\, n \geq \log_2(i+1) \tag{3.2}$$

where `i' and `n' are natural numbers starting from 1, 2, 3, ..., ∞. If the 'n' value resulted in the fractional part, then its next higher natural number is to be considered. For example, for adding three partial products in a column, the total number of levels is $n \geq \log_2(3+1)$, so n=2. Similarly, suppose i=8, i.e., $n \geq \log_2(8+1)$, and it is evaluated as n=3.16. As 'n' should be a natural number, its next higher natural number is considered and therefore n=4. Figure 3.5 shows the basic block diagram for K stages and L levels. As shown in the figure, 'A_0', 'A_1', 'A_2' up to 'A_K' are the partial products; the term 'Y_0' is the sum and 'Y_1', 'Y_2', 'Y_3',....., 'Y_L' are the carries. The algorithm shown in figure 3.5 is, therefore, in simple words, an N-bit compressor circuit that generates the sum of a particular column and single or multiple carries.

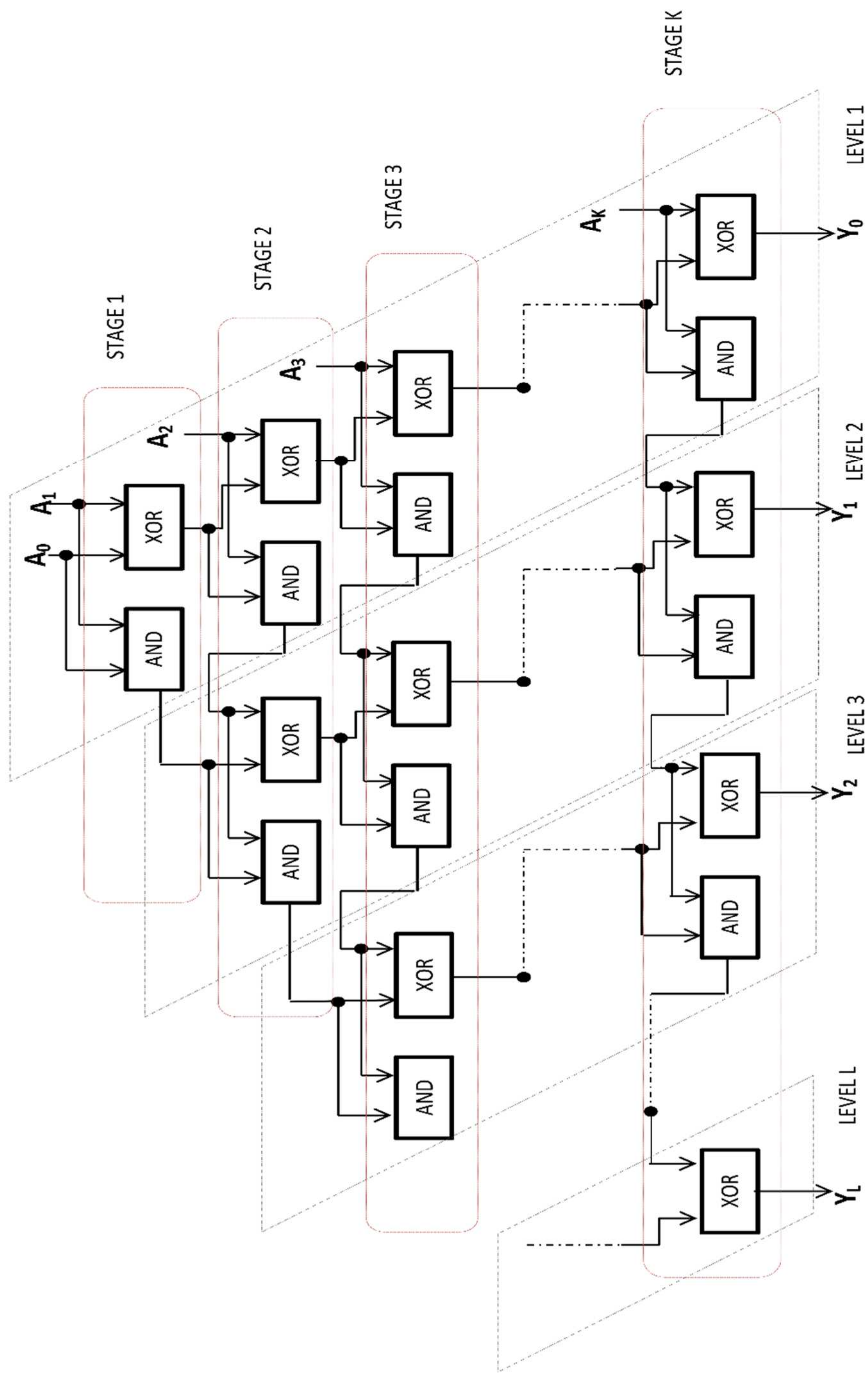


Figure 3.5: AND-XOR gate arrangement with K stages and L levels

3.3.3 Special cases

- In the last level, only the XOR gate is used instead of the AND-XOR pair
- When i=2, only one level is used to get the sum and carry. In this scenario, the carry is the data output from the AND gate.
- For i=1, the input itself is the sum (output), and it does not generate a carry.

It is worth noting that the output through level 1 is the sum of the partial products present in a particular column, and the outputs of the rest of the levels are the corresponding carry bits, i.e., level 2 to level 'L'. Upon obtaining the sum as well as carry bits of all columns, the next move is to add the sum bits with the previous column's carry bits. For this, any of the practical algorithms, such as the DADA algorithm, Wallace tree algorithm, or even ripple carry adder, can be used as the number of rows has significantly decreased.

3.4 CONCLUSIVE REMARKS

The novel UCM architecture is a universal method for compressor design, which is dominantly used in multiplier architecture. The compressor architecture is capable of N:M bit compression; therefore, it can be directly applied to a multiplier with $N \times N$ bits. Moreover, the UCM architecture has reduced the complexity of the Wallace tree multiplier because of the novel compressor algorithm. The application of the UCM on MAC architectures is shown in chapters 4 and 5. The power-delay and PVT analysis of UCM architecture is shown in chapter 6.

CHAPTER 4: UNSIGNED/SIGNED FIXED-POINT MAC ARCHITECTURE (UMAC, USMAC & SMAC)

4.1 INTRODUCTION

The DSP devices are used in many applications, such as image processing, speech encoding, audio mixing, etcetera. The MAC unit plays a critical role in these applications since the input signals must be multiplied and then added with the previous result. The primary MAC unit includes a multiplier, summer (or adder), and register. MAC's arithmetic operations can be performed on two different number systems: a) fixed point and b) floating-point. There are signed and unsigned numbers in the fixed-point representation, which are to be multiplied and then added, but at the same time, the fixed-point number system is not sufficiently efficient for performing arithmetic operations on reasonably large numbers. Therefore, there is a requirement for the floating-point number system. The floating-point number system is the combination of the mantissa term and the exponent terms. So, in general, the real numbers in a floating-point number system is represented as equation 4.1.

$$N = M \times B^E \tag{4.1}$$

where ‘M’ is the mantissa, ‘B’ is the base, and ‘E’ the is exponent. Therefore, all such design aspects of fixed, as well as floating-point numbers, must be considered when constructing a MAC unit.

On the other hand, the function of the MAC unit is termed, as shown in equation 4.2.

$$Z = \sum_{i=0}^{n-1} a_i b_i \tag{4.2}$$

The equation 4.2 represents that a MAC unit performs multiplication of two numbers and add the result with the previously stored values. The primary building block for the MAC unit, as discussed earlier, is multiplier and adder. For the MAC block to be efficient, the MAC unit's multiplier and adder blocks must be efficient in terms of power, speed, and area.

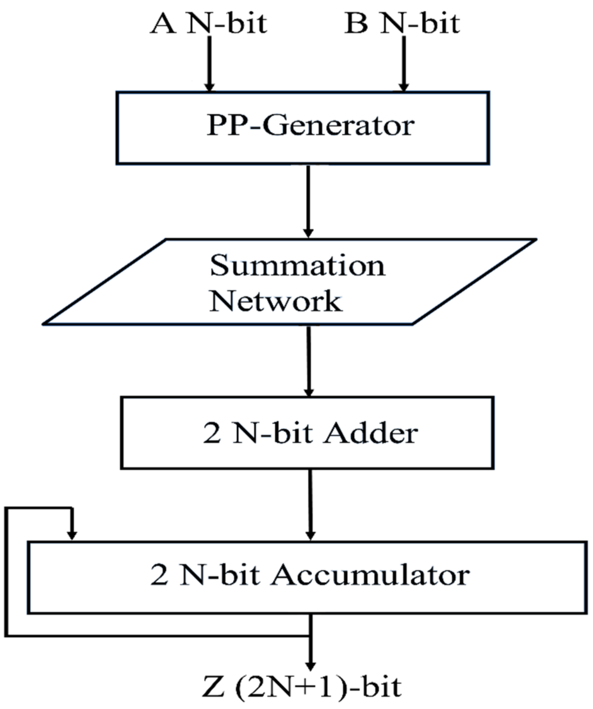


Figure 4.1: Basic MAC Unit

4.2 BASIC BUILDING BLOCKS OF MAC

The basic building block of the MAC unit is represented in figure 4.1 [14]. The multiplier block collects and multiplies two n-bit inputs, and produces the 2N-bit output, which is further processed to the register/accumulator unit. The register temporarily stores the data and sends the data to the adder as an input. The adder sums up the register unit output together with the accumulator register accumulated value, which is the result of the previous cycle. Thus, the MAC unit's overall output is taken from the accumulator register output. Hence, the MAC unit architecture consists of an N-bit multiplier, 2N bit register, (2N+1) bit adder, and two (2N+1)-bit accumulator registers (one for storing the output value and the other for reading the previous output).

4.2.1 Multiplier

As explained earlier, the processing elements of MAC mainly involve the multiplication of two numbers; therefore, in such types of processing systems, the multiplier is required. In the literature, various fast and effective multipliers are described. The Array Multiplier is a basic multiplier that follows the product generation and addition principle. But this architecture becomes bulkier with higher PDP when the total number of summation levels increases. The solution to this problem is to use the "Wallace tree multiplier based on the structure of Wallace tree". In 1964, C.S.Wallace proposed the Wallace tree multiplier, which "generates the product of two numbers using purely combinational logic, i.e., in one gating step". This work has also outlined a rapid square-root process [11], as explained in figure 4.2. However, in the Wallace tree multiplier, every partial product is added in the top to bottom direction. Therefore, the total number of adders increases in a conventional Wallace tree multiplier. A rectangular styled Wallace tree multiplier is proposed in which the "partial products are divided into two groups and added in the opposite direction to overcome this problem. The partial products in the first group are added downward, and the partial products in the second group are added upward" [15]. On the other hand, in the literature, a phase mode parallel multiplier is also proposed [16]. The presented multiplier has a "Wallace-tree structure comprising trees of carry-save-adders for the addition of partial products". This structure has avoided the use of the irregular structure of the conventional Wallace tree; therefore, it is much appropriate for pipeline operation.

A couple of architectures in the literature also focused on adder cell optimization. As adder is an essential unit in multiplier or divider, the main focus of the optimization is on the adder part. The literature proposes a carry-select-adder optimization technique in which a "carry-select-adder partitioning algorithm" is used for the Wallace tree multiplier using booth encoded techniques, which is found to be much efficient [4]. By considering different data arrival times, a "branch-and-bound algorithm" is proposed, and a generalized technique to separate an n-bit carry-select-adder in several small blocks of adder unit is introduced so that the overall delay of the design can be minimized. In a separate approach by [22], an improved version of modulo ($2^n + 1$)

multipliers is proposed in 2013. The efficiency is achieved by “manipulating the Booth tables and by applying a simple correction term” in the existing modulo ($2^n + 1$) multiplier algorithm. Moreover, the author states that "the proposed multiplier is almost as efficient as those for ordinary integer multiplication”. On the other hand, in 2012, a comparative analysis is done by [13] for designing a multiplier using “complementary MOS (CMOS) logic style, Complimentary Pass Transistor (CPL) logic style, and Double Pass Transistor (DPL) logic” style. A single-precision reversible floating-point multiplier is proposed by [21] in the year 2010. A 24-bit multiplier is proposed in this work by decomposing the whole 24 bits in three portions of 8 bit each.

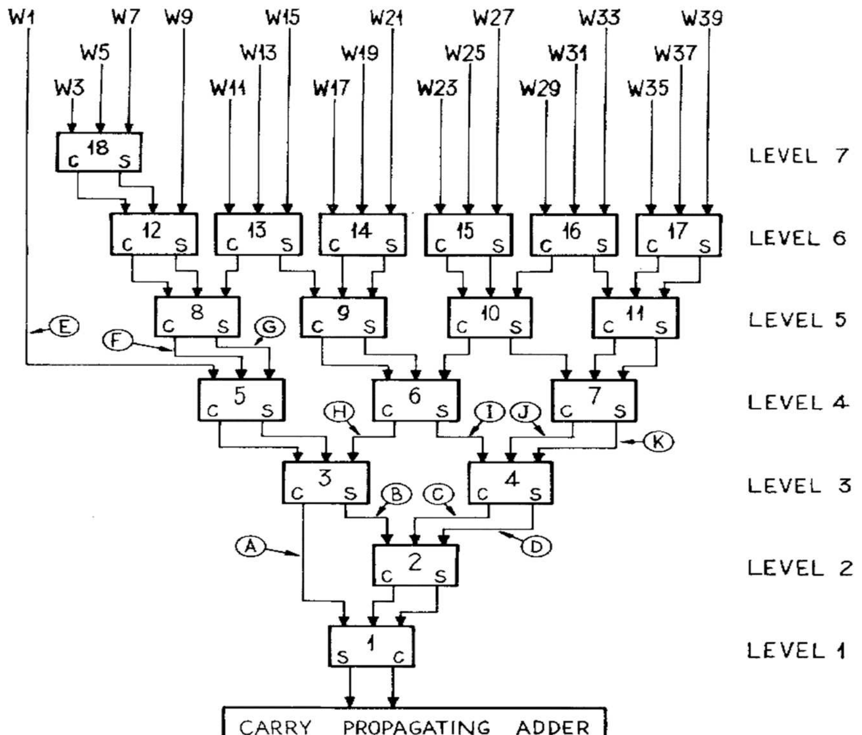


Figure 4.2: Wallace tree multiplier (addition of partial products)

4.2.2 Adder

An adder is also known as summer, is a logic circuit that adds two numbers. An adder or summer circuit is used not only for addition but also for multiplication, updating the address, increment/decrement operation, table indices, etcetera. The adder operation is performed in binary number systems, but the adder can also be applied on BCD, excess

-3, etcetera. In the literature, various full adder architectures are proposed. In 2013, a “novel low power hybrid full adder using MOSIS 90 nm technology” is proposed, which consumes meager power [25]. The design being proposed is compared to its conventional full adder, which consists of 28 transistors. A hybrid 1-bit full adder is introduced in a different approach, which uses both CMOS and TG logic styles [3]. The entire design is implemented in both 90 nm technology and 180 nm technology. The proposed design offers very little power at 1.8V supply voltage and moderately low delay. Figure 4.3 shows the adder, as described in [3].

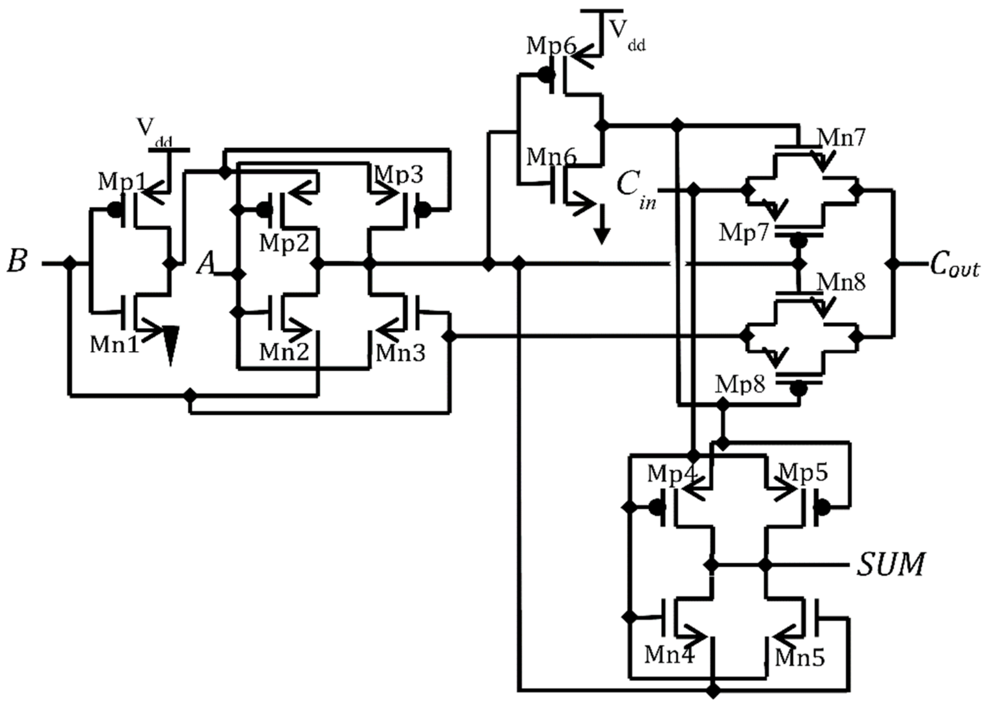


Figure 4.3: Full adder design which uses both CMOS and TG logic styles

4.3 EXISTING ARCHITECTURES OF MAC UNIT

In 2007, an 8-bit, 16-bit, and 32-bit MAC is proposed and implemented on the Xilinx ISE and FPGA boards [1]. The design shows both area and power improvements. 4:2 Compressor circuits are used for faster design of the multiplier circuit. Using Cadence Virtuoso 180 nm Technology in [55], an 8-bit MAC Unit is proposed. For the said MAC architecture, several adder/multiplier circuits are also compared and implemented. In 2012, a multiplier in which the terms are rearranged to reduce the "total number of partial products by 25%" is proposed and shown in figure 4.4. [6]. The proposed multiplier is further used to offer a MAC architecture. Also, cadence NC Sim and RTL compiler are used to do the analyses.

			a	b	c	d	
			X	w	x	y	z
0	az	az	cx&by	dw	cy	dy	dz
0	0	ax	cw^ay	az	dx	cz	0
0	0	cw&ay	by	cx^by	bz	0	0
P7	P6	P5	P4	P3	P2	P1	P0

Figure 4.4: Partial product addition matrix

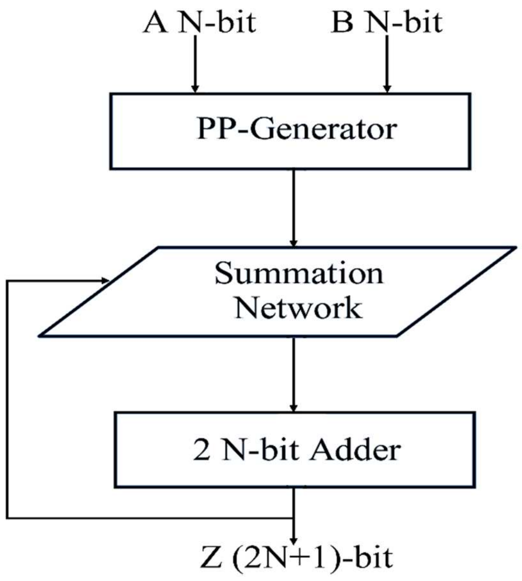


Figure 4.5: 32-bit MAC architecture and its ASIC implementation

In a different approach in [7], it suggest a multiplier circuit using a modified Wallace tree multiplier and carry-save-adder. Further, a MAC device is also designed for 64-bit input, operating at 217 MHz and consuming a total dissipation of 177,732 mW of power. Abdelgawad has proposed an ASIC implementation of the 32-bit MAC in [65]. The proposed architecture has reduced hardware complexity, thereby reducing power consumption and decreasing delay, which decreases the area by 5.5%, power by 9%, and delay by 13% compared to conventional MAC architecture. Figure 4.5 displays the block diagram of the proposed design by Abdelgawad. The simulation is performed in 180nm technology using HDL.

In 2013, a new MAC architecture based on a modified Braun multiplier with a bypass technique is proposed [8]. Designs are implemented in CMOS technology of 130 nm. The full adders in the circuit are constructed using TG logic, DPL logic style, etcetera. In [9], it is suggested that a pipeline MAC architecture consisting of a 16-bit multiplier. The multiplier that is implemented is based on the Baugh-Wooley algorithm. The proposed architecture is found to be more power-efficient, which is 30 percent to 80 percent lower than traditional MAC architectures. Implementation is performed in the TSMC library using HDL in 65 nm CMOS technology. In 2015, the authors have implemented a Vedic multiplier and various logic-based reversible designs in [79]. Using these, finally, a 32-bit MAC architecture has been designed, as shown in figure 4.6. The implementations are rendered in Cadence RTL using Verilog HDL.

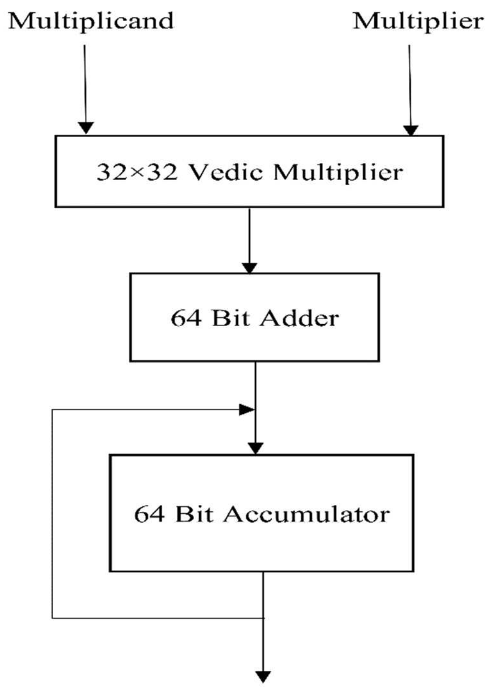


Figure 4.6: 32-bit MAC architecture

As adiabatic architecture offers little energy dissipation, a MAC unit using adiabatic logic is proposed by [44]. Using a smaller number of complex logic gates, the non-adiabatic dissipation is optimized, and the comparative study of the proposed MAC with the existing designs is also discussed. On the other hand, in 2009, the authors have implemented a novel design in ModelSim in TSMC 90 nm CMOS technology in [10]. Here "4-pipelined high-performance split Multiply-Accumulator (MAC)" architecture is proposed. To improve the architecture's operating speed, a partial product compression circuit based on "interleaved adders" and a hybrid "Partial-Product-Reduction-Tree (PPRT)" is proposed. The benefit of this MAC is that it can perform 1-way 32-bit or 4-way 16-bit signed/unsigned "multiply or MAC operation" and "2-way parallel multiply-add operations". Figure 4.7 shows the architecture discussed in [10].

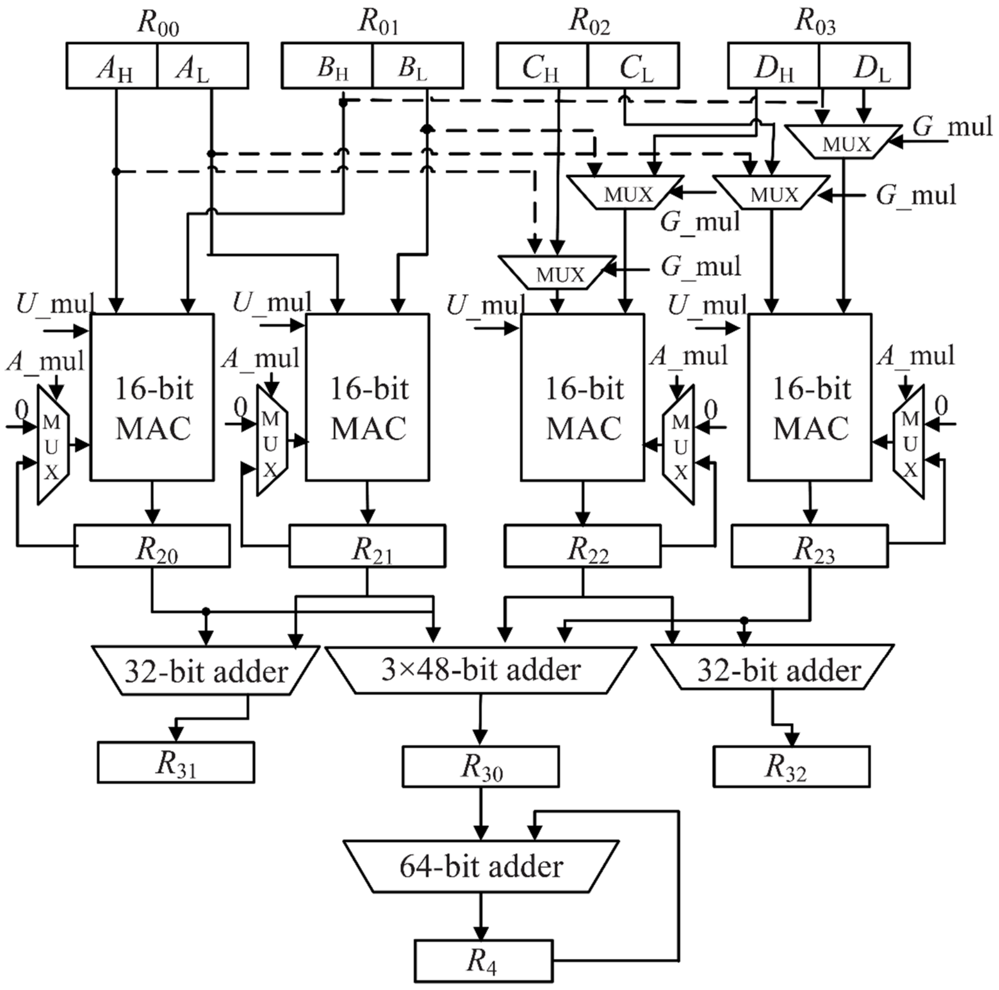


Figure 4.7: Pipelined MAC architecture

In 2009, the researchers have proposed an 8-bit MAC unit using 180 nm technology [14]. Various adder/multiplier circuits are compared and implemented for the MAC. As the circuit designed by [14] is without a clock signal, it faces a synchronization issue. In 2014, a multiply-added floating-point unit for low-precision formats is proposed [66]. To achieve this architecture, which is required in the processing of mantissa data in a single operation, the multiplication and addition/subtraction operations are fused.

The architecture is implemented on the FPGA board. In 2005, the paper by [48] have discussed regarding RSFQ DS Processor, specifically used to eliminate interference from any signal. The author suggested the incorporation of multiplication-addition in the MAC for floating-point operation. The suggested MAC module consists of three units, i.e., a parallel multiplier, combiner, and accumulator, as seen in figure 4.8. The combiner performs the “summation of sums and carries from M-MSB bits” of the multiplier. In VHDL, the simulation is verified.

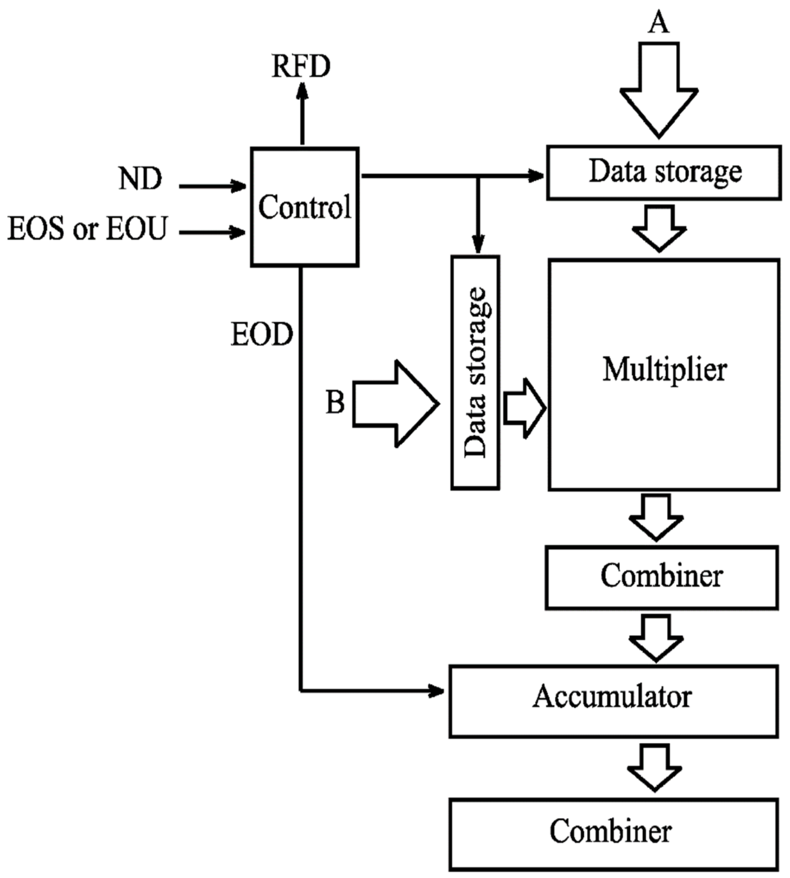


Figure 4.8: The RSFQ DS processor architecture

A MAC unit specific for “programmable bandpass filtering” is described in [49]. This MAC device is clock-able at 20 GHz frequency and can perform “2.5 billion MAC operations/second for 7-bit data”. In VHDL, the simulation is tested. Authors have suggested a "block matching motion estimation" method for the minimization of the accumulator unit in MAC [50]. This paper also explores the implementation of the MAC on FPGA and its mathematical models. In [51], an architecture of "64-bit fixed-point vector MAC" is proposed, which supports multiple precisions. The MAC vector can perform “one 64 × 64”, “two 32 × 32”, “four 16 × 16”, or “eight 8 × 8" bit signed or unsigned multiplication and accumulation using the same hardware as the scalar 64-bit MAC architecture. The proposed design is implemented using Verilog HDL in the Synopsys tool, as shown in figure 4.9. A two-cycle MAC architecture with efficient power-delay is proposed in [56]. The proposed architecture includes accumulation guard bits and saturation circuitry as well as it supports two’s complement numbers. Implementation is performed on VHDL and conceived in a cell library of 65 nm at 1.1V.

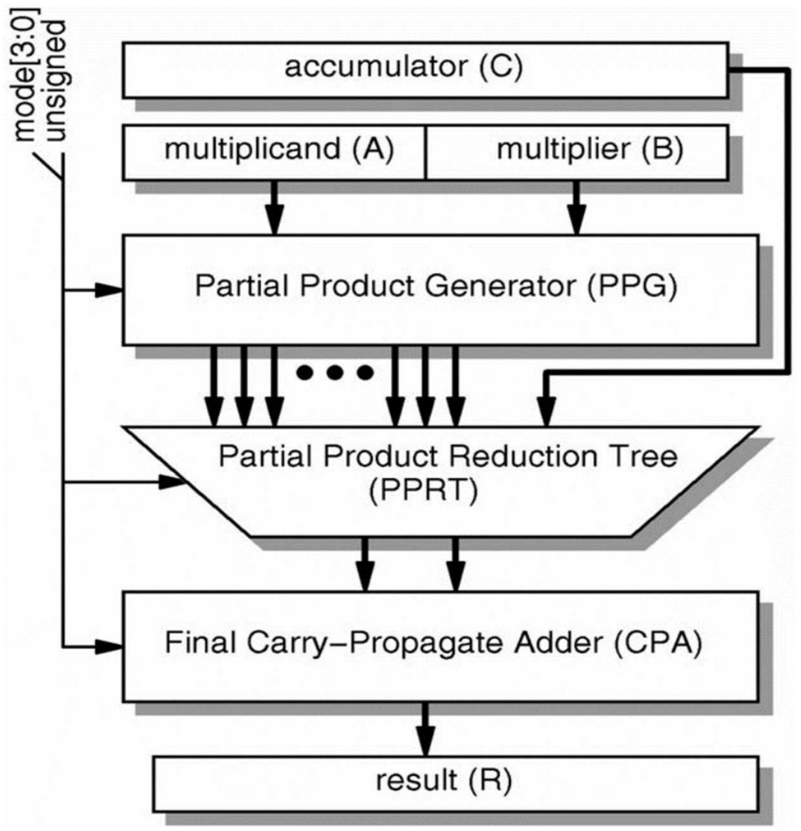

mode[3:0]
unsigned
accumulator (C)
multiplicand (A)
multiplier (B)
Partial Product Generator (PPG)
Partial Product Reduction Tree (PPRT)
Final Carry–Propagate Adder (CPA)
result (R)

Figure 4.9: Fixed-point vector MAC architecture

4.4 PROPOSED MAC ARCHITECTURES

The proposed MAC architectures focus primarily on the signed/unsigned architecture for fixed-point inputs based on the synchronized block that are enabled with proper pipelining. The block enabling is a power saver technique that temporarily triggers a circuit, and the circuit becomes disabled afterward. Most of the energy/power can be saved because of this basic phenomenon. The second reason why synchronization is introduced is to avoid unnecessary data loss. Because proper synchronization is not available, the data being processed in the preceding block may get lost while transferring the same to the next block. Third and most importantly, the processing through pipelining is the digital system's ultimate necessity as it improves the system's performance tremendously.

As the two core blocks are multiplier and adder, a detailed analysis is done while selecting the appropriate circuits. The delay-cum-power efficient design is given critical importance whilst selecting the adder. As mentioned in chapter 1, the simultaneous minimization of delay and power consumption of any circuit is not possible because of the trade-off between these two design constraints. Therefore the 'delay-cum-power efficient design’ here signifies the optimization of one of the design constraints while minimizing the other or vice versa. The adder circuit proposed in [3] is used in the proposed MAC design, as it is found to be the most suitable for delay-cum-power efficient design. On the other hand, as stated in section 4.1, it is found that although the array multiplier is considered to be the most straightforward algorithm for multipliers, it generates a very high delay compared to the Wallace tree multiplier. Whereas, the Wallace tree multiplier in rectangular style is the best option since it divides the partial products into two groups and hence faster than the conventional Wallace tree multiplier [15]. But the irregular structure is the most significant disadvantage of Rectangular styled Wallace tree multiplier. Therefore, the novel UCM architecture (proposed in the previous chapter), which has a better performance in terms of delay in comparison to the Wallace tree multiplier, is chosen as the multiplier for the proposed 8-bit MAC architectures.

4.4.1 Proposed Unsigned MAC architecture (UMAC)

The unsigned architecture is nothing more than conventional MAC architecture. For n-bit inputs, as discussed earlier, it consists of an n-bit size multiplier, a (2n+1)-bit adder, and a 2n & (2n+1)-bit register or accumulator. Figure 4.10 and figure 4.11 shows the detailed block diagram and the output waveform of the UMAC architecture, respectively.

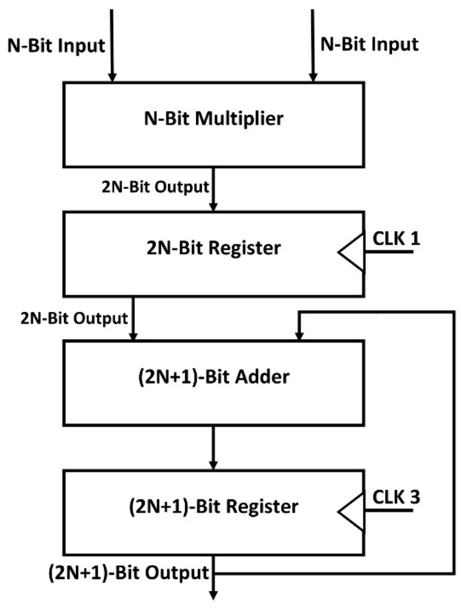


Figure 4.10: Proposed architecture of UMAC

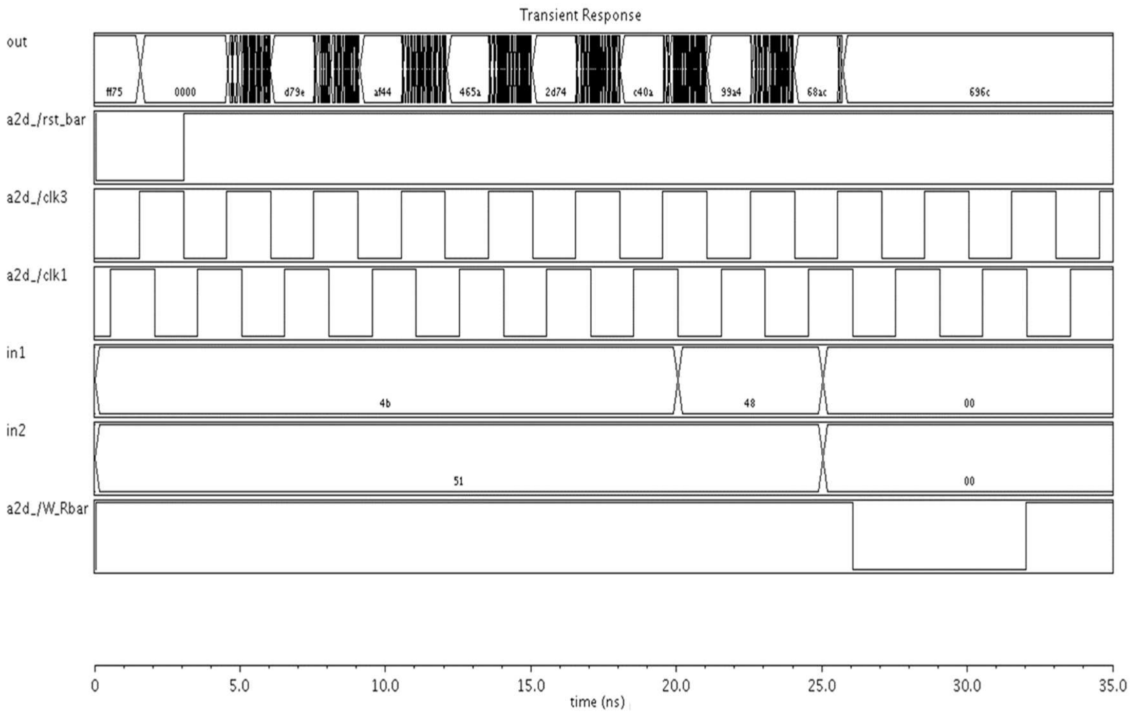


Figure 4.11: Output waveform of the proposed UMAC architecture

4.4.2 Proposed Unsigned Synchronized MAC architecture (USMAC)

The critical disadvantage of UMAC architecture is that the synchronization mechanism is not available, due to which the appropriateness of this architecture is in doubt. It creates precise results, but it is challenging to verify the output waveform. Besides, the static power consumption for the UMAC architecture is very high due to the lack of the block enabling technique. Since the UMAC architecture requires no synchronized mechanism, only the registers are synchronized with clock pulses that make the multiplier and adder blocks active throughout the simulation. Hence the rise in the consumption of static power.

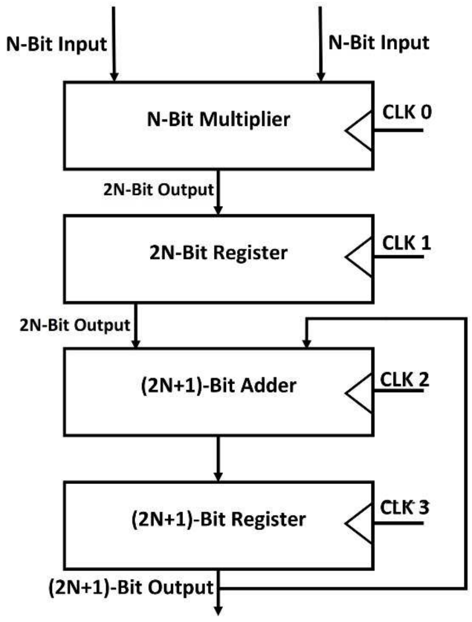


Figure 4.12: Block diagram of USMAC architecture

USMAC architecture is proposed to rectify the errors of UMAC architecture. The add-on to the USMAC architecture is that the individual design blocks are connected to the clock pulse with a PED block and a Latch block to detect the clock edges and temporarily store the processed data, as shown in figure 1.7, respectively. Due to the minor modification in the architecture, all the results can now be read and validated. Moreover, USMAC's static power consumption is also meager compared to UMAC architecture, since each block of the USMAC architecture is synchronized with the proper clock pulse. On the other side, it also adds the pipelining framework for proper data latching. The clock signal to the individual block of the USMAC architecture is provided with sufficient delay to control the pipeline process, as an incorrect data latching may result in undesirable results at the final output. A delay of 500 ps is thus maintained between the 1st and 2nd blocks and so on. Figure 4.12 and figure 4.13, respectively, display the block diagram and the output waveform of the proposed USMAC architecture.

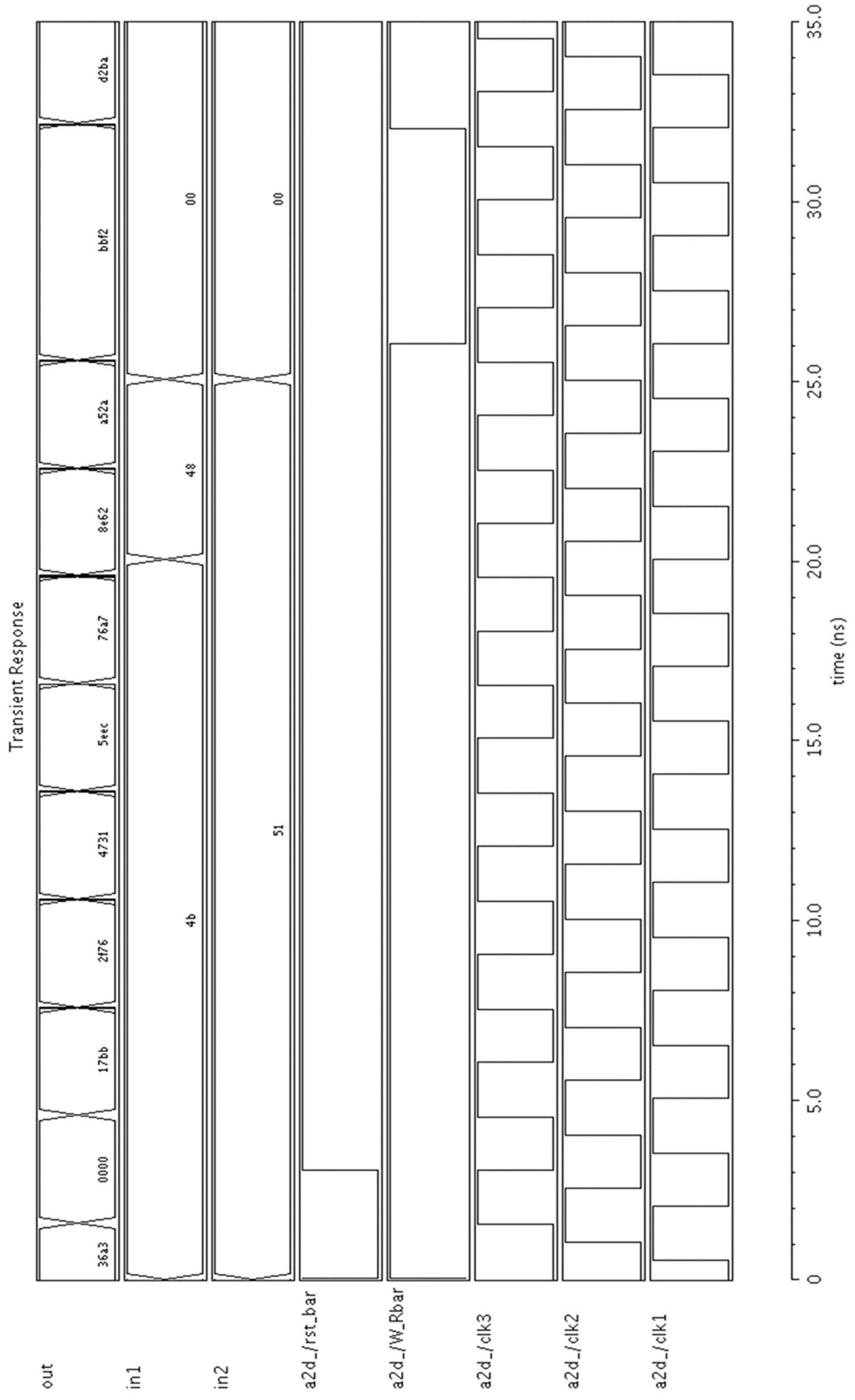


Figure 4.13: Output waveform of the USMAC architecture

4.4.3 Proposed Signed MAC architecture (SMAC)

In SMAC architecture, care is taken of the MAC process for positive, as well as the negative number. Multiplexers are used in this architecture for choosing positive and negative numbers. For the multiplication of negative numbers, the negative values are expressed in 2's complement form. For proper data latching, a gap of 500 ps is established between each block, as described in the previous sub-section. The proposed SMAC architecture block diagram and output waveform are shown in figure 4.14 and figure 4.15, respectively.

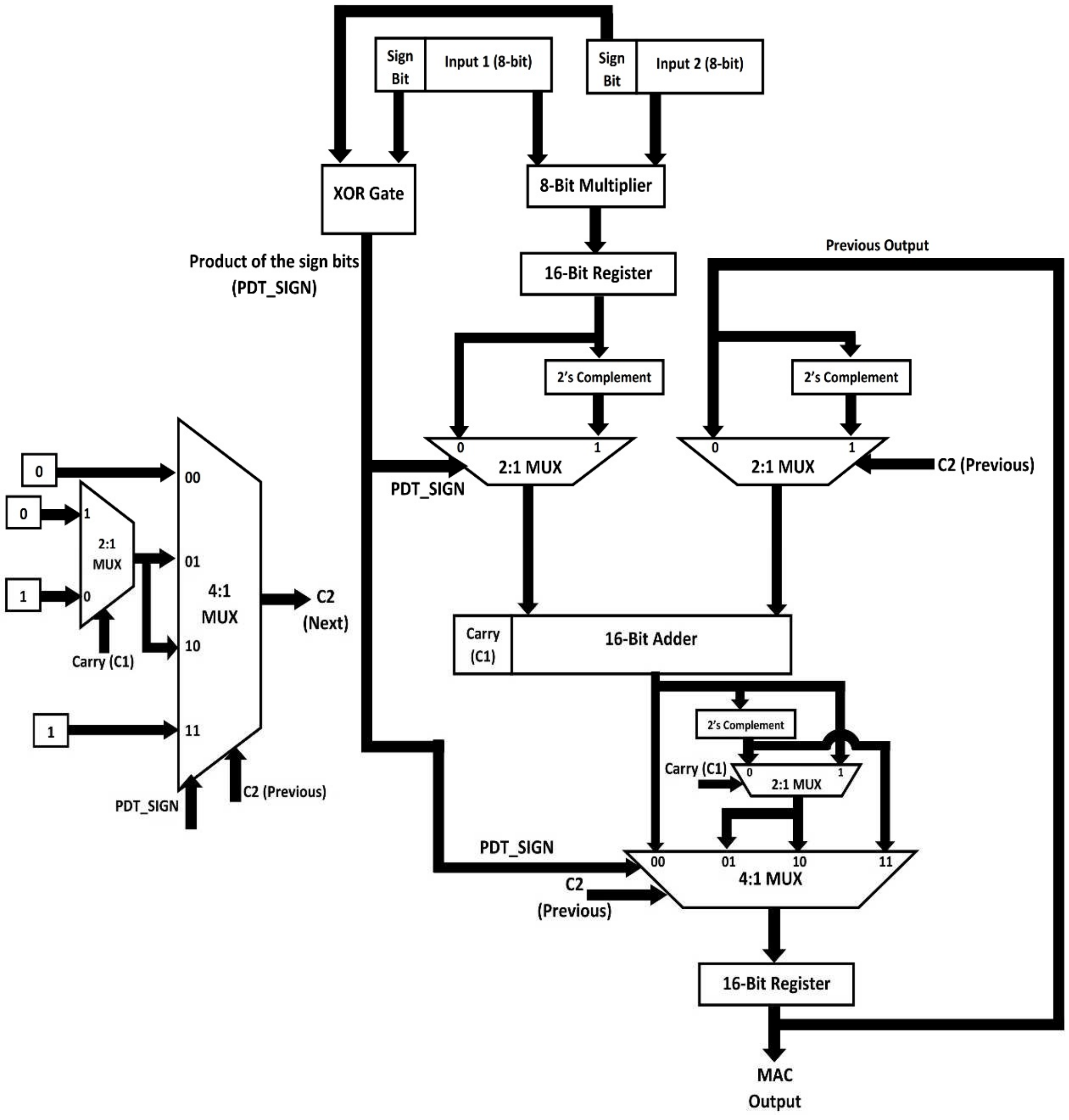


Figure 4.14: Block diagram of SMAC architecture

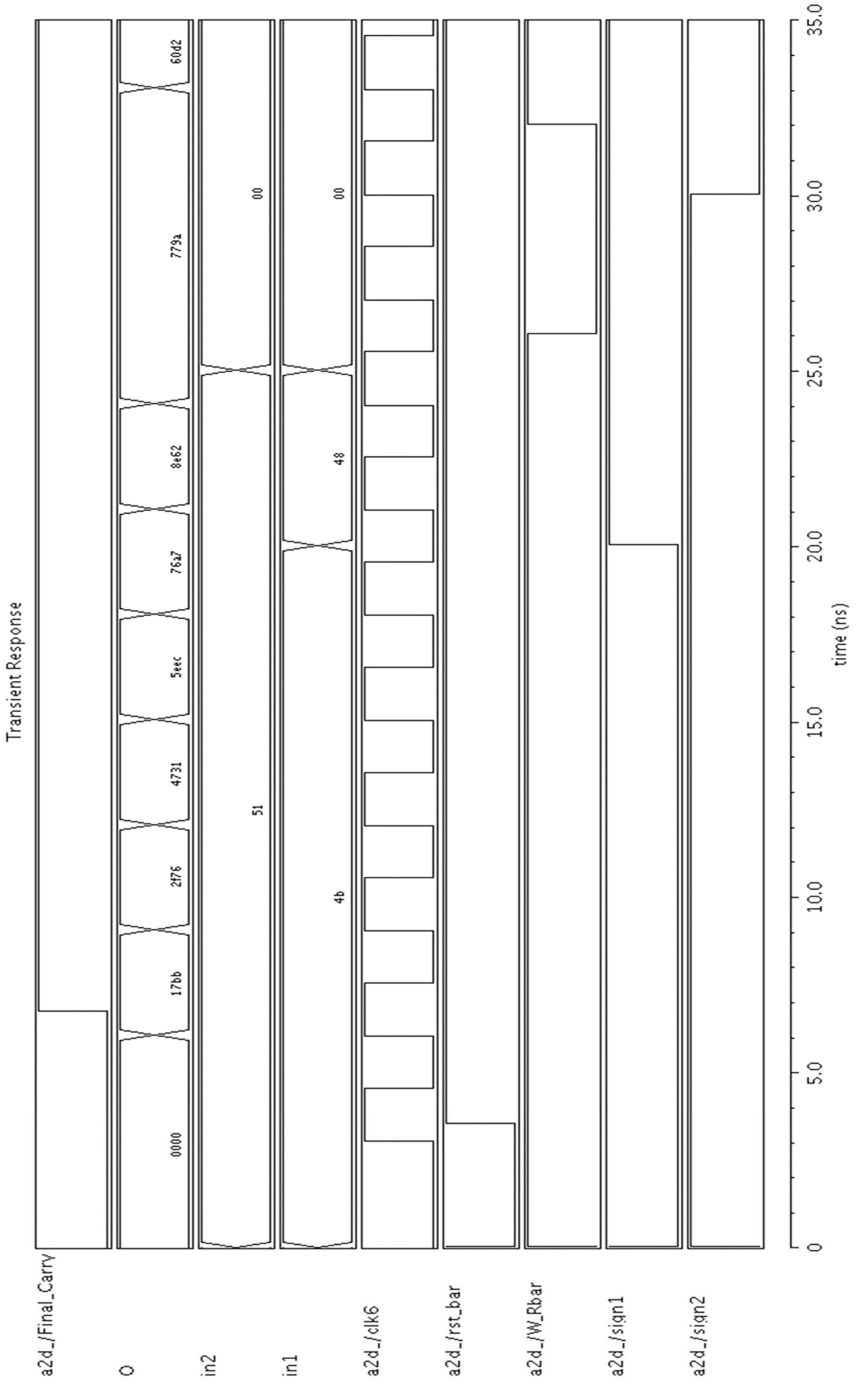


Figure 4.15: Output waveform of the SMAC architecture

4.5 CONCLUSIVE REMARKS

The novel MAC architectures for unsigned/signed fixed-point architectures are explained in this chapter. As shown in this chapter, the UMAC architecture (designed for unsigned MAC operation) has the limitation of producing the result accurately because of the non-availability of synchronization. This problem is rectified by the USMAC architecture, where the unsigned MAC is synchronized using a clock gating technique along with pipelining. Finally, for signed MAC operation, SMAC architecture is proposed, which is capable of performing the multiply-accumulate operation on positive as well as negative inputs. The MAC unit with floating-point and signed input (SFMAC architecture) is explained in chapter 5.

CHAPTER 5: MUX BASED SIGNED FLOATING-POINT MAC (SFMAC) ARCHITECTURE

5.1 INTRODUCTION

The actual MAC block is not just limited to the fixed-point number system. For applications such as image processing, speech encoding, audio mixing, etcetera, floating-point MAC architecture is much needed. In the literature, different approaches are adopted for designing effective floating-point MAC architectures. In [52], one of the notable floating-point MAC architectures is proposed, where a “pipelined single-precision Floating-Point Multiply Accumulator (FPMAC)" consisting of the accumulator in radix-32 and internal carry-save addition is explained. Additionally, an improved version of “Leading-Zero Anticipator (LZA) and overflow prediction logic” required in carry-save addition is also described. The FPMAC is shown in figure 5.1.

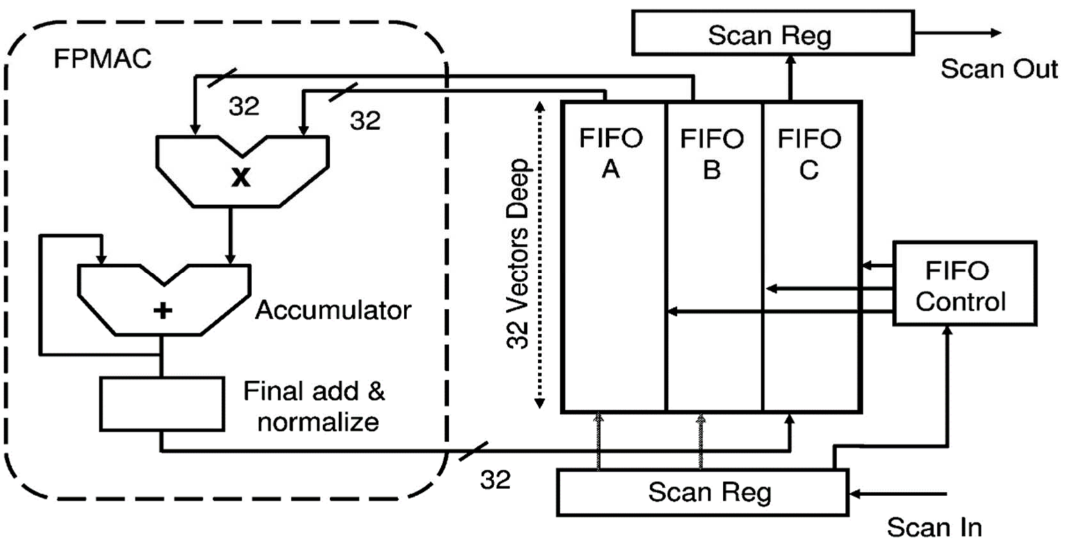


Figure 5.1: Pipelined single-precision FPMAC

A floating-point MAC architecture is proposed in the year 1996, where an LZA technique is used for a FADD core, as shown in figure 5.2. This logic carries out the simultaneous execution of the “pre-decoding for normalization” along with a summation of the significand [34]. The rounding operation, in parallel with the shifting of the normalization, is also proposed in this architecture. The CMOS logic is used for implementing the primary circuits for the design. Its area penalty of the FADD core is found to be as low as 30% of the traditional LZA method. The FADD core is fabricated by 0.5 µm CMOS technology at a supply voltage of 3.3 V.

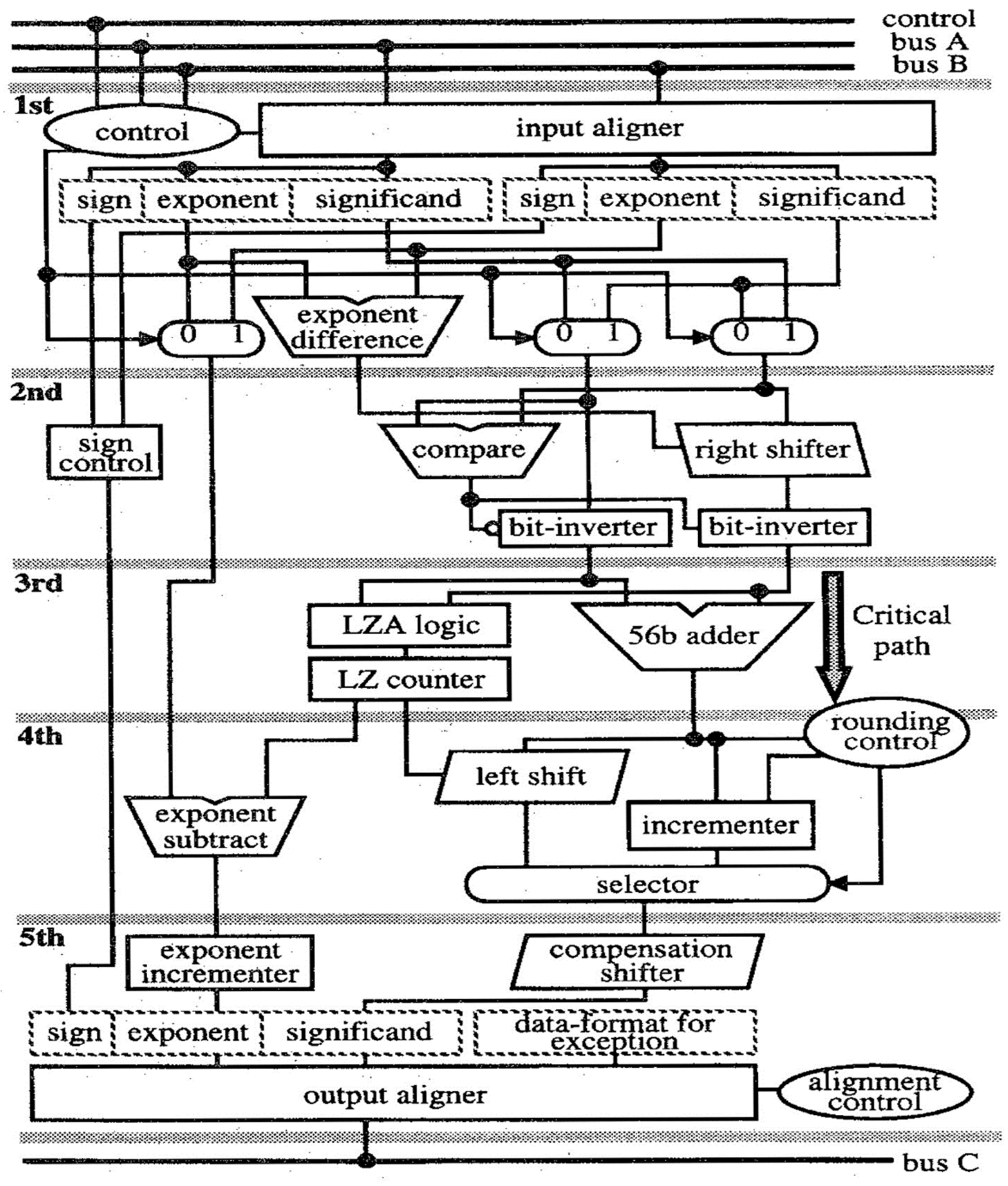


Figure 5.2: An LZA logic for floating-point addition operation

A “Floating-point multiply-add fused” architecture for IEEE 16-bit or IEEE 32-bit (half precision or single precision respectively) is discussed in [66]. The architecture is designed by the amalgamation of the multiplier and adder/subtractor required for mantissa data calculation in a single operation, which has provided an efficient usage of DSP blocks in FPGAs. The architecture is also implemented on FPGA. The architecture is shown in figure 5.3.

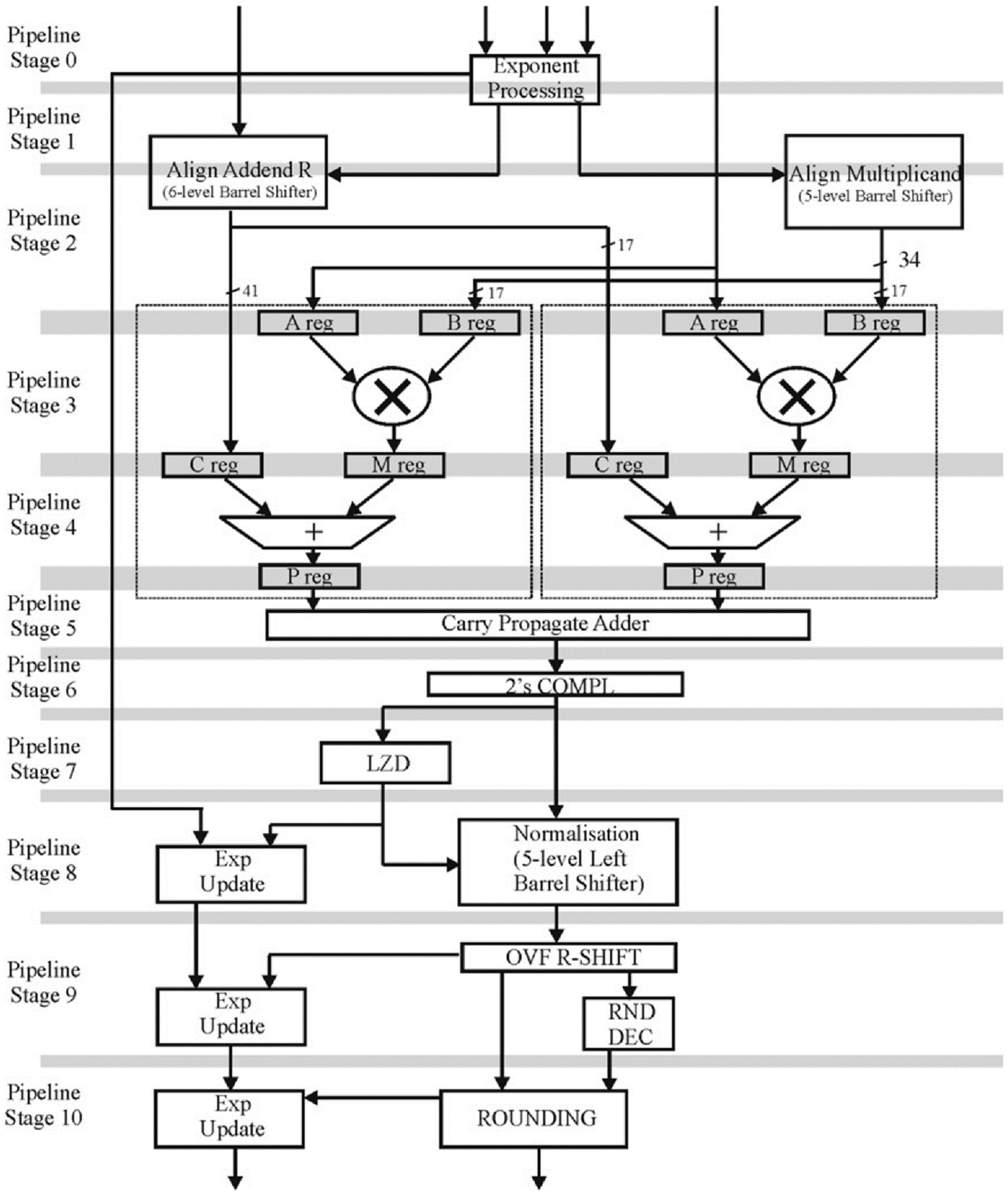


Figure 5.3: Floating point multiply-add units for IEEE 16-bit or IEEE 32-bit

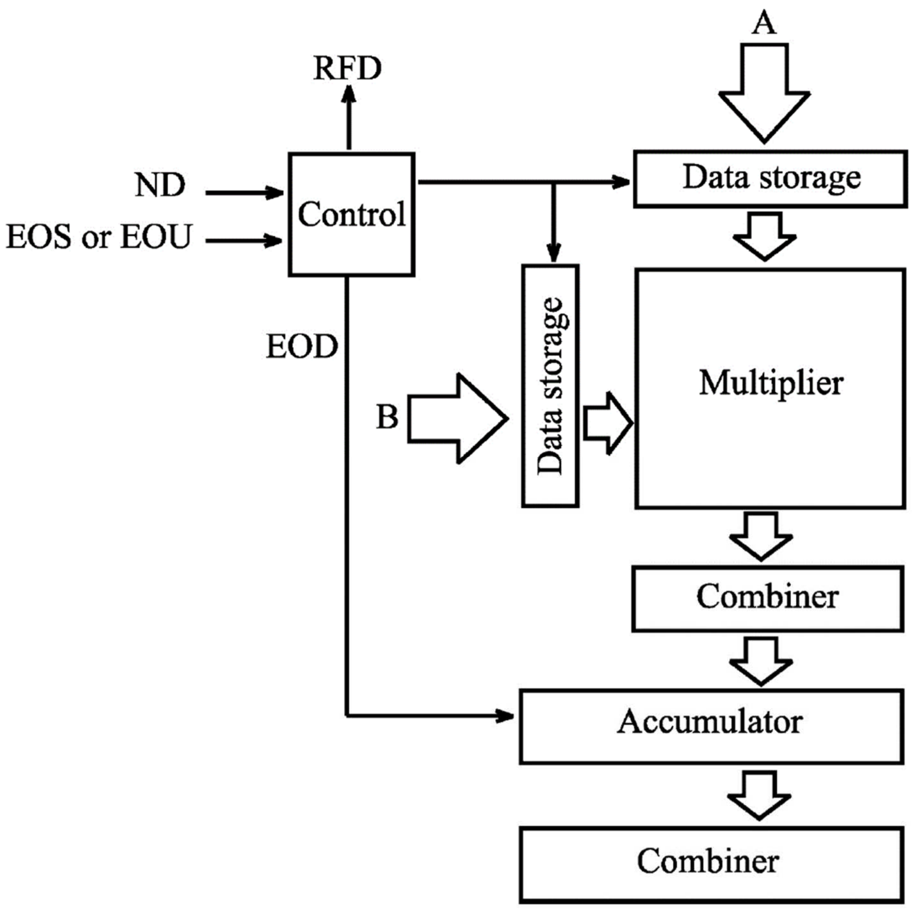


Figure 5.4: RSFQ DS Processor

In [48], as shown in figure 5.4, explains about RSFQ DS processor, mainly used for the removal of interferences from any signal. The author proposed the MAC unit for floating-point multiplication-addition. The MAC unit consists of a three-unit parallel multiplier, combiner, and accumulator. The “combiner performs a summation of sums and carries from M-MSB” bits of the multiplier. The simulation is verified in VHDL.

In this chapter, a multiplexer-based MAC architecture is proposed, which is capable of performing multiply and accumulation on signed floating-point inputs. For this, a novel input-data format is introduced, which takes 9-bit binary data with the MSB as the sign bit and 4-bit exponential input with the MSB as the exponential sign bit. Therefore, the size of the novel input-data format is 13-bits. Moreover, the SFMAC architecture uses multiplexer circuits rigorously for selecting among a positive or negative number. The next section and its sub-section explain the SFMAC architecture in detail, which mainly consists of input format representation, EA block, ECC block, ESC block, etcetera.

5.2 SFMAC ARCHITECTURE

The architecture has a separate number representation. The initial considerations for the proposed SFMAC architecture are as mentioned below:

1. The architecture uses sign-magnitude as well as 2's complement representations to represent positive as well as negative numbers (including exponent terms). The overall inputs and output of SFMAC are represented in sign-magnitude form, whereas for internal calculations, the same data are converted into 2's complement form. The final output of the proposed MAC architecture (MAC output) is 16-bits, and the sign bit of the result is identified by the ‘C2’ bit.
2. The inputs to the SFMAC are two 8-bit binary number arranged in a format, as shown in figure 5.5 below:

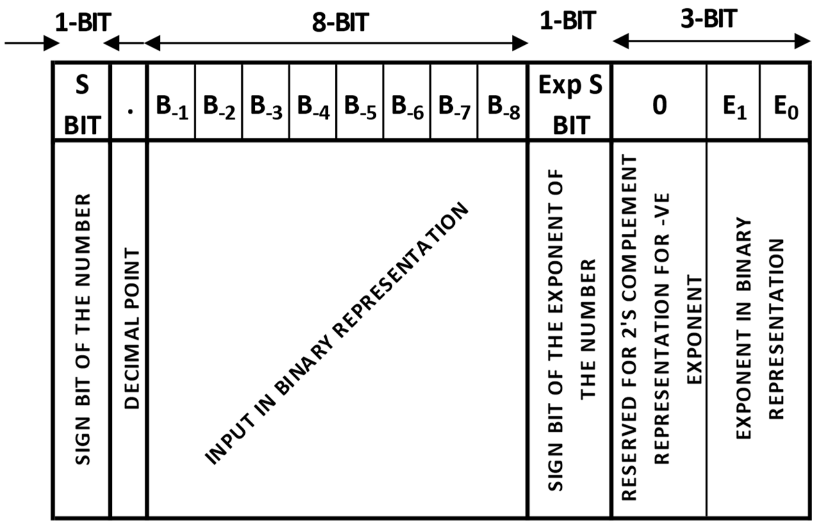

1-BIT
8-BIT
1-BIT
3-BIT
S BIT
.
B_{-1} B_{-2} B_{-3} B_{-4} B_{-5} B_{-6} B_{-7} B_{-8}
Exp S BIT
0
E_1 E_0
SIGN BIT OF THE NUMBER
DECIMAL POINT
INPUT IN BINARY REPRESENTATION
SIGN BIT OF THE EXPONENT OF THE NUMBER
RESERVED FOR 2'S COMPLEMENT REPRESENTATION FOR -VE EXPONENT
EXPONENT IN BINARY REPRESENTATION

Figure 5.5: Input format representation of SFMAC

The size of each input of the SFMAC representation is 13 bits, in which two bits are reserved for the sign bits of the number and its exponent. The sign bit can be '0' or '1' based on positive or negative number representation, respectively. Remaining eleven bits are used for 8-bit binary representation and 3-bit exponent representation in binary. One significant point here to note is that the 3rd bit of the exponent in binary representation is by default made as '0' because, to represent a 2-bit number in 2’s complement form, it requires 3 bits. The range of 2’s complement representation is given by equation 5.1.

$$-(2^{n-1})\ to + (2^{n-1}-1) \qquad (5.1)$$

Where 'n' is the number of bits.

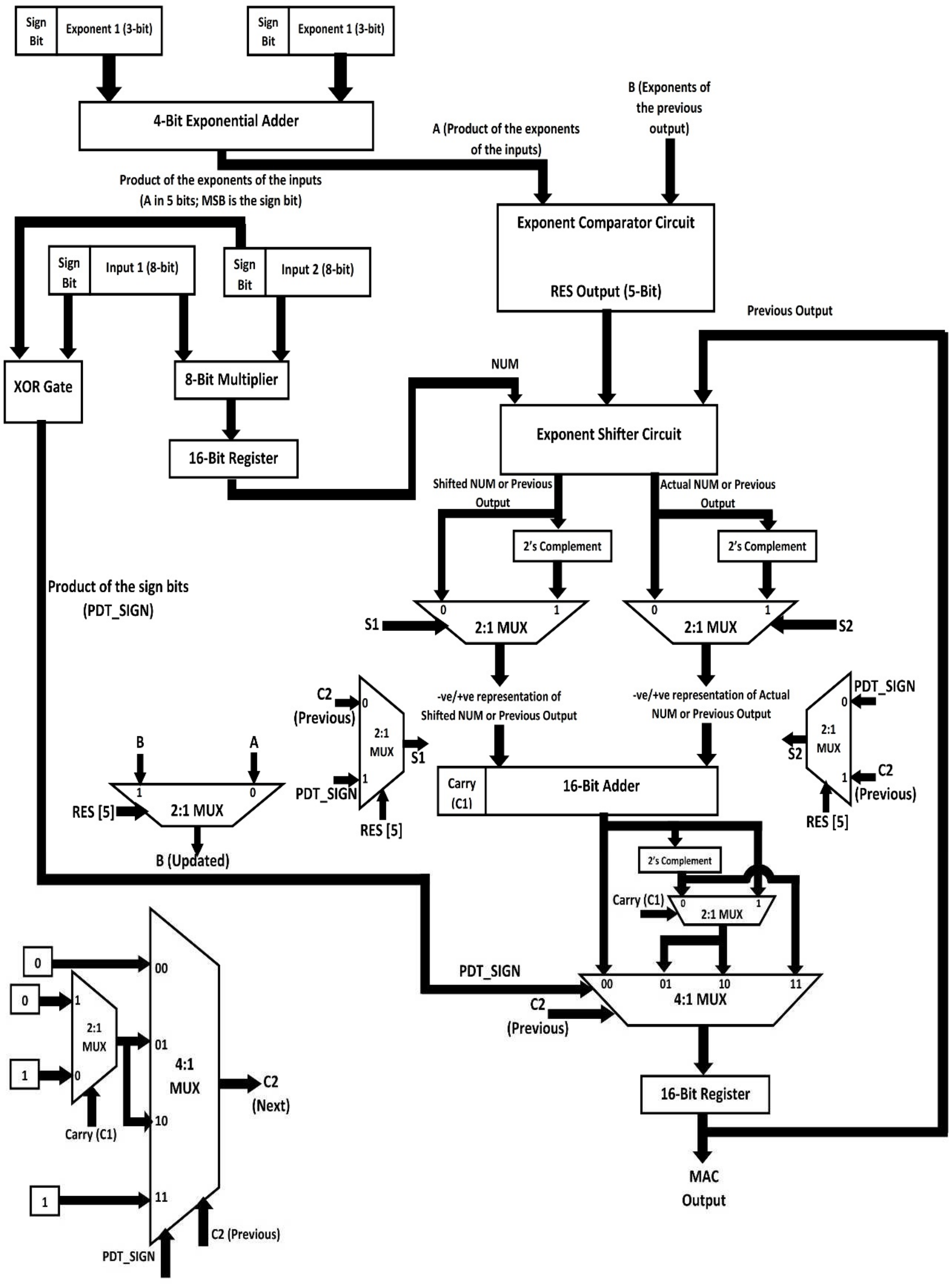


Figure 5.6: The novel SFMAC architecture

Therefore, in this architecture, the exponent term can range from ‘-4’ to ‘+3’. Hence, the input numbers can have a range from $-(0.11111111)_2 \times 2^{+3}$ to $+(0.11111111)_2 \times 2^{+3}$ and thus, the range of the inputs of the current SFMAC architecture in a decimal number system is from $-(7.96875)_{10}$ to $+(7.96875)_{10}$.

3. The inputs to the SFMAC architecture should be entered in decimal point only. For example, instead of providing the inputs to the SFMAC as $(001)_2$ and $(010)_2$, the numbers should be entered as $(0.00100000)_2 \times 2^{+3}$ and $(0.0100000)_2 \times 2^{+3}$. Similarly, $(101)_2$ and $(10)_2$ should be represented as $(0.10100000)_2 \times 2^{+3}$ and $(0.10000000)_2 \times 2^{+2}$ respectively to process it through the SFMAC.
4. The EA block performs multiplication of the exponents of the inputs. Therefore, it basically adds the exponents (as $2^n \times 2^m = 2^{(n+m)}$). Though the inputs to the EA block is of 4 bit each (including one sign bit), it produces the result in 5 bits as the addition of two 2-bit number can produce a maximum of 3-bit result and for representing a 3-bit binary number in 2's complement form, it requires 4-bits. On the other hand, the MSB bit (i.e., 5th bit) is the sign bit of the result.

The primary content of the SFMAC architecture are:

- Exponential Adder (EA)
- 8-bit multiplier
- 16-bit register
- Exponent Comparator Circuit (ECC)
- Exponent Shifter Circuit (ESC)
- 16-bit adder and
- 2:1/4:1 multiplexer of different sizes

The overall architecture of SFMAC is shown in figure 5.6.

5.2.1 Exponential Adder (EA)

As mentioned above, the EA block performs the multiplication of the exponential terms of the inputs. The size of each exponent is 4-bit, out of which one bit is reserved for sign representation. The EA architecture is shown in figure 5.7, and the following steps are followed in the EA block:

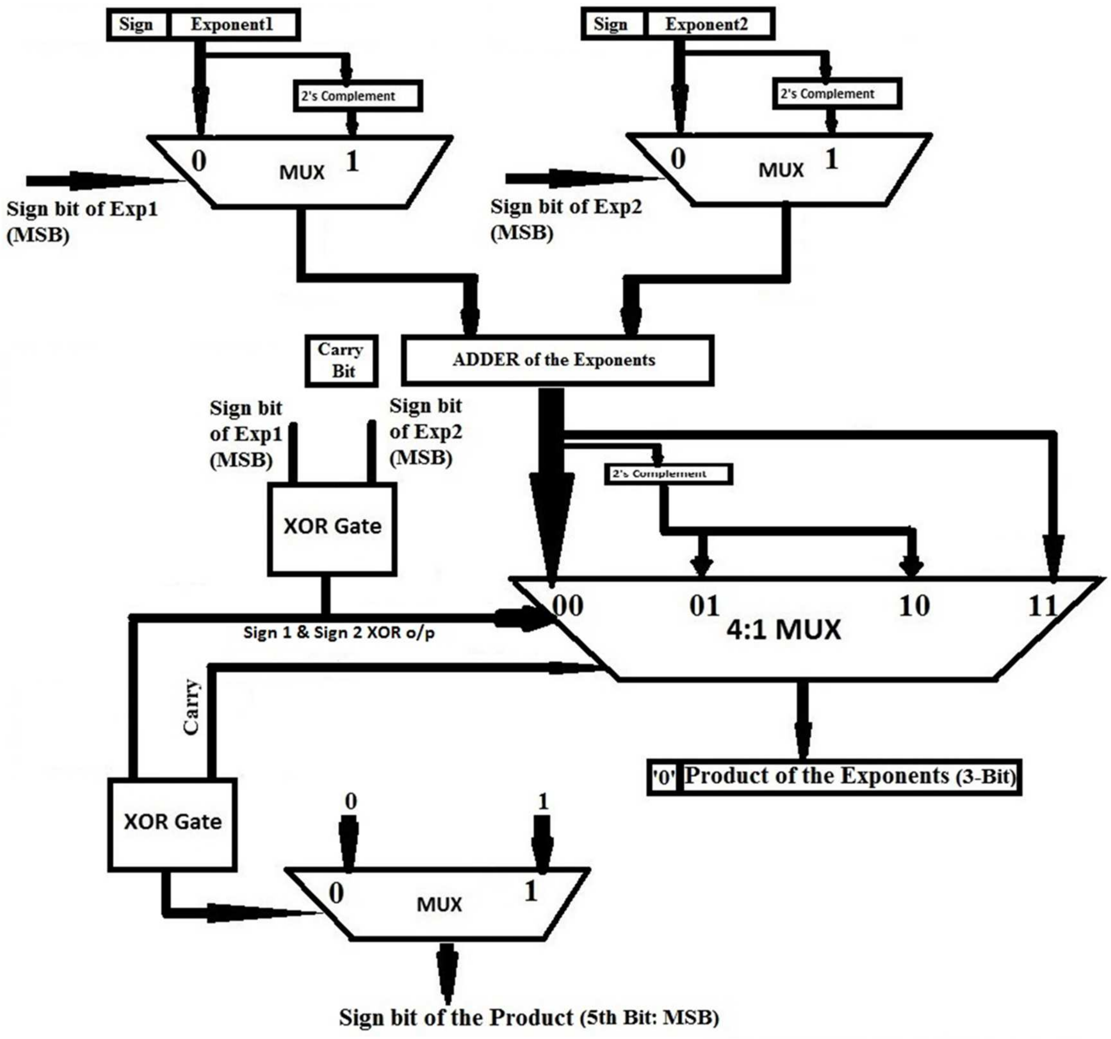


Figure 5.7: Exponential Adder (EA) architecture

i) Based on the sign bit, the exponents are represented in 2’s complement form.

ii) True/2’s complement form of both the exponents are added using a 4-bit adder block.

iii) As the inputs to the adder block is in true or 2’s complement form, the sum term of the adder doesn’t provide an exact result. Therefore, the output of the adder block is further processed through a 4-bit 4:1 multiplexer to represent the result in sign magnitude form. The outcome of the 4-bit 4:1 multiplexer is based on the following conditions:

- If the ‘XOR’ output of both the sign bit and carry a bit of the adder block is either '00' or '11' then pass, the adder output itself is the output of the 4-bit 4:1 multiplexer.

- Else the output of the 4-bit 4:1 multiplexer is the 2’s complement representation of the adder block.

iv) The output of the EA is in sign-magnitude form only. The final sign bit of the EA block is based on the ‘XOR’ value of the sign bit of the exponent 1, exponent 2, and the carry bit of the adder block.

v) A PED-latch block pair is used at the output of each output bits to make the EA block synchronized.

5.2.2 8-bit multiplier

The multiplier block used in this case is the novel UCM architecture, which is explained in detail in chapter 3. The additional circuitry that is added to the multiplier is the synchronization. As used in the EA block, a PED-latch block pair is used at the output of each output bits of the multiplier.

5.2.3 16-bit register

Generally, due to fluctuation in the inputs, the output changes, and it is almost impossible to track the output. The primary use of the register is to hold the data until the next cycle is processed. Here, 16-bit registers are used at the final output and immediately after the multiplier. The main content of the register is a D flip-flop and a data selection circuit consisting of basic gates. The basic design of the register is already elaborated in chapter 1.

5.2.4 Exponent Comparator Circuit (ECC)

As shown in figure 5.8, the inputs to the ECC are the product of the exponents (EA output, i.e., 5-bit) and the output exponent of the previous cycle (5-bit in size). The major point to consider here is that if both the input terms to the ECC block carry the same sign, then the actual difference among the two is the arithmetic difference between the numbers. Whereas, if both the inputs carry different signs, then the actual difference among the two is the arithmetic sum of the two numbers. For example, the actual difference between '+a' and ‘+b’ is ‘a-b’ or ‘b-a’. Whereas, for ‘-a’ and '-b', the actual difference is 'a-b' or 'b-a' only. But if the inputs are '+a' and ‘-b’ or ‘-a’ and ‘+b’ then the actual difference is going to be ‘a+b’ or ‘b+a’. The operation of the ECC block is as follows:

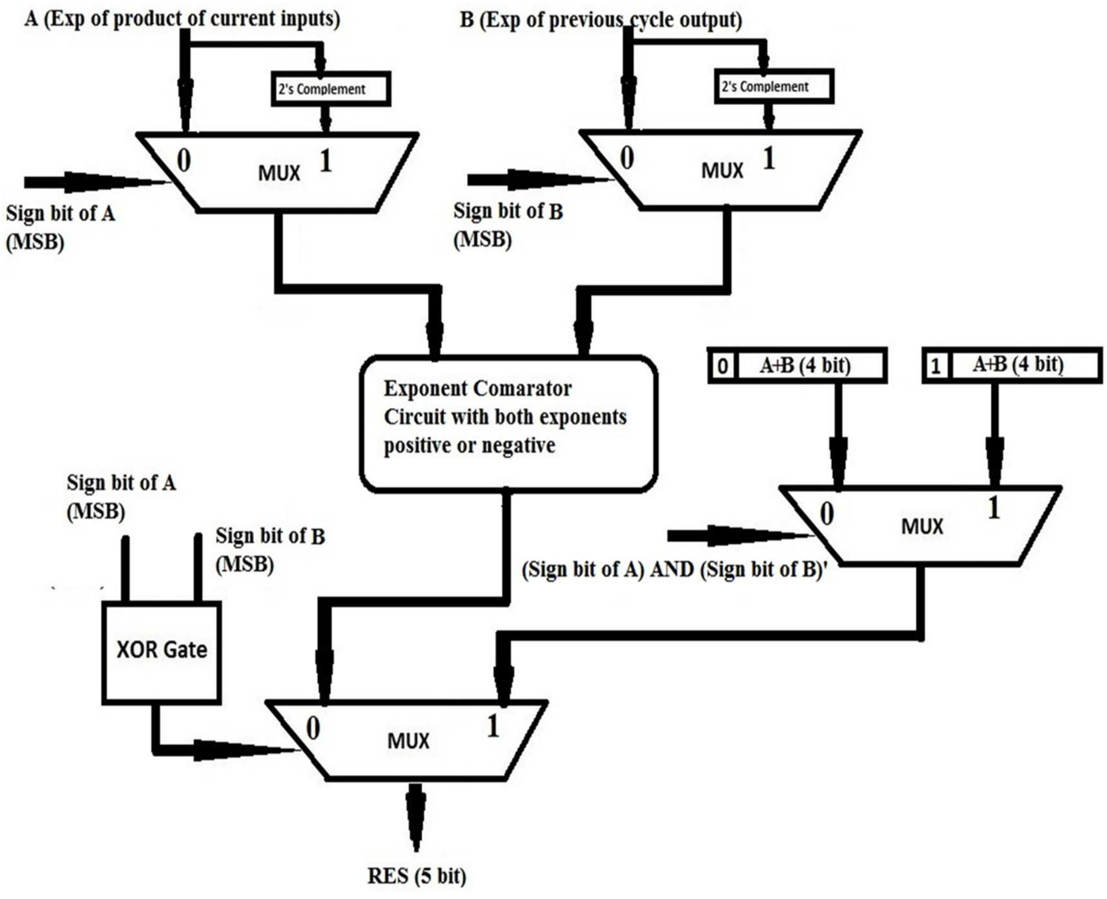

A (Exp of product of current inputs)
2's Complement
B (Exp of previous cycle output)
2's Complement
0
MUX
1
Sign bit of A (MSB)
0
MUX
1
Sign bit of B (MSB)
Exponent Comarator Circuit with both exponents positive or negative
0 A+B (4 bit)
1 A+B (4 bit)
0
MUX
1
(Sign bit of A) AND (Sign bit of B)'
Sign bit of A (MSB)
Sign bit of B (MSB)
XOR Gate
0
MUX
1
RES (5 bit)

Figure 5.8: The ECC architecture

i) Based on the sign bit, the inputs to the ECC are represented in 2’s complement form.

ii) The operation of the ECC is further segregated based on the sign bits of the inputs as follows:

a. If both the sign bits are different, then add the inputs of the ECC to produce a 4-bit output (i.e., discard the carry bit) but introduce the 5th bit as '1' if the product of the exponents of the inputs is negative, but the previous exponent is positive. Make the 5th bit as '0' in the other circumstances.

b. If both the sign bits of the inputs to the ECC are same then find out the input which is higher among the two and find the difference between the inputs as per the following procedure:

- For finding the higher number, compare both the numbers bit by bit, i.e., start comparing from MSB to LSB, as shown in figure 5.9.
- For finding the difference, use the 2’s complement approach. The difference produces a 4-bit output (i.e., discard the borrow bit) but introduces the 5th bit as '0' if the product of the exponents of the inputs is higher than the previous cycle exponent. Make the 5th bit as ‘1' in the other circumstances.
- In this architecture, multiplexers are used to compare the inputs.

iii) This operation produces a 5-bit output, which is further used for performing the binary shifts.

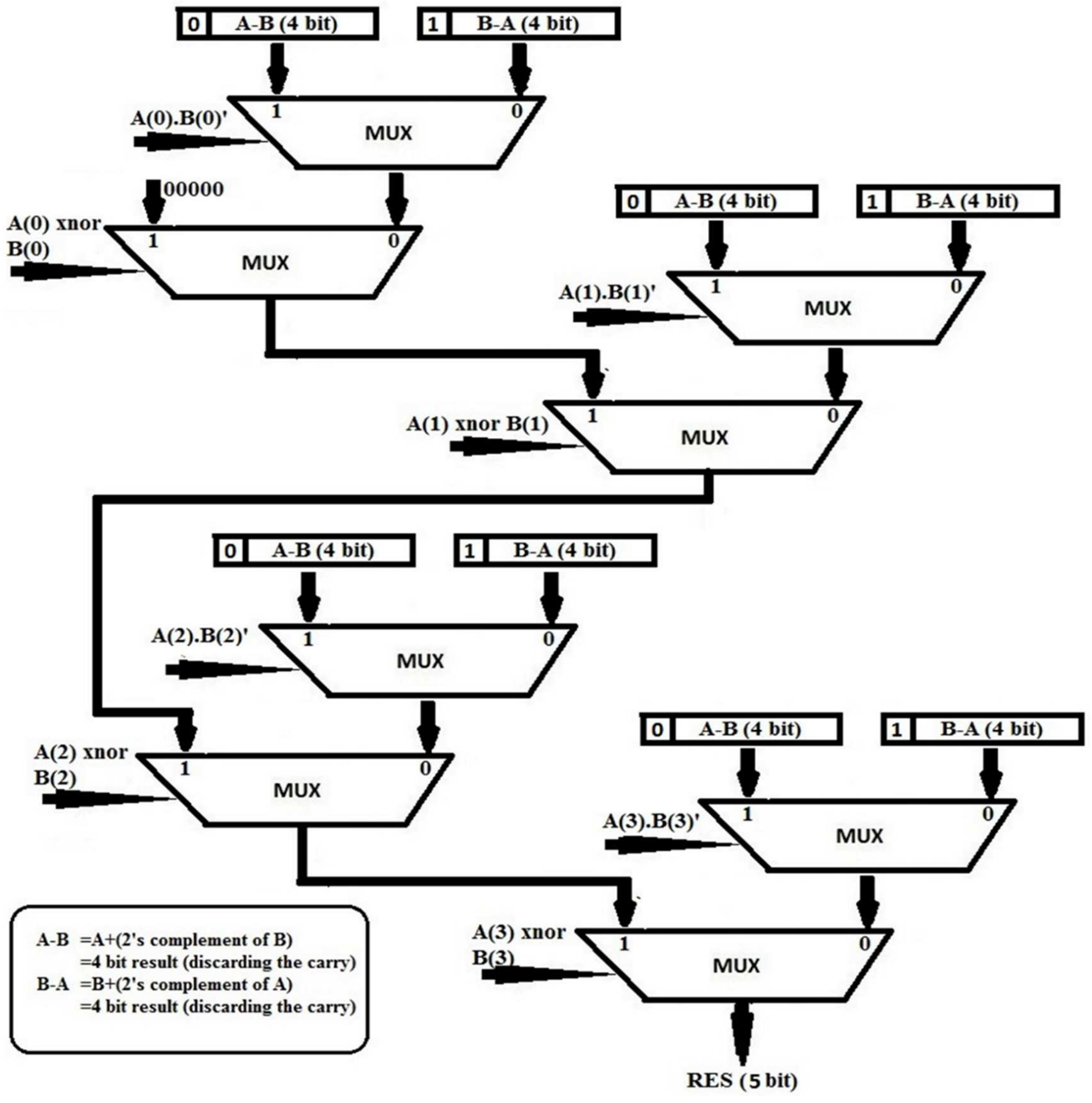
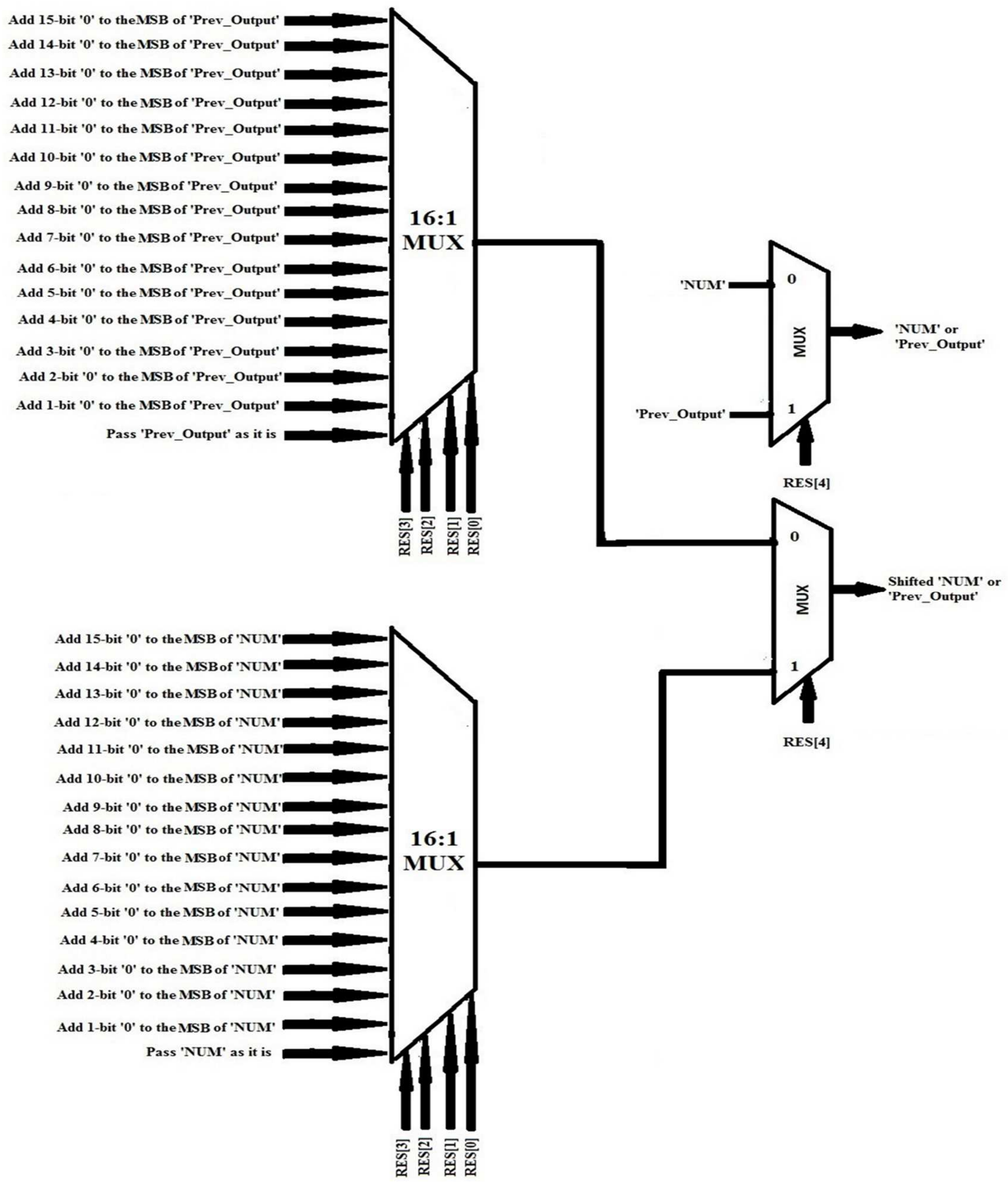


Figure 5.9: The ECC with same sign bit

5.2.5 Exponent Shifter Circuit (ESC)

The ESC block is responsible for shifting the smaller number (either the product of the 8-bit inputs or the previous cycle MAC output) by the amount of difference between the exponents of these two. The inputs to the ESC block are the 5-bit output of the ECC block, a 16-bit product of the inputs, and 16-bit value of the previous cycle output. The step by step procedure is as follows:



Add 15-bit '0' to the MSB of 'Prev_Output'
Add 14-bit '0' to the MSB of 'Prev_Output'
Add 13-bit '0' to the MSB of 'Prev_Output'
Add 12-bit '0' to the MSB of 'Prev_Output'
Add 11-bit '0' to the MSB of 'Prev_Output'
Add 10-bit '0' to the MSB of 'Prev_Output'
Add 9-bit '0' to the MSB of 'Prev_Output'
Add 8-bit '0' to the MSB of 'Prev_Output'
Add 7-bit '0' to the MSB of 'Prev_Output'
Add 6-bit '0' to the MSB of 'Prev_Output'
Add 5-bit '0' to the MSB of 'Prev_Output'
Add 4-bit '0' to the MSB of 'Prev_Output'
Add 3-bit '0' to the MSB of 'Prev_Output'
Add 2-bit '0' to the MSB of 'Prev_Output'
Add 1-bit '0' to the MSB of 'Prev_Output'
Pass 'Prev_Output' as it is
16:1 MUX
RES[3]
RES[2]
RES[1]
RES[0]
'NUM'
'Prev_Output'
0
1
MUX
'NUM' or 'Prev_Output'
RES[4]
0
1
MUX
Shifted 'NUM' or 'Prev_Output'
RES[4]
Add 15-bit '0' to the MSB of 'NUM'
Add 14-bit '0' to the MSB of 'NUM'
Add 13-bit '0' to the MSB of 'NUM'
Add 12-bit '0' to the MSB of 'NUM'
Add 11-bit '0' to the MSB of 'NUM'
Add 10-bit '0' to the MSB of 'NUM'
Add 9-bit '0' to the MSB of 'NUM'
Add 8-bit '0' to the MSB of 'NUM'
Add 7-bit '0' to the MSB of 'NUM'
Add 6-bit '0' to the MSB of 'NUM'
Add 5-bit '0' to the MSB of 'NUM'
Add 4-bit '0' to the MSB of 'NUM'
Add 3-bit '0' to the MSB of 'NUM'
Add 2-bit '0' to the MSB of 'NUM'
Add 1-bit '0' to the MSB of 'NUM'
Pass 'NUM' as it is
16:1 MUX
RES[3]
RES[2]
RES[1]
RES[0]

Figure 5.10: The ESC architecture

i) As shown in figure 5.10, the identification of the smaller number is made based on the ECC output (5-bits). If the MSB of the ECC block output is '1', then the product of the inputs is shifted towards the right by the equivalent decimal value of the remaining 4-bit binary of the ECC block output. On the other hand, if the MSB of the ECC block output is '0', then the previous output is shifted towards the right by the equivalent decimal value of the remaining 4-bit binary of the ECC block output.
ii) The input to the ESC block, which need not be shifted, is identified by the same MSB of the ECC block output.

5.2.6 16-bit adder

The adder block is again a synchronized block (i.e., it is clocked). The outputs of the ESC block are processed through a 2's complement block and a 2:1 Multiplexer for representing a positive or negative value. For example, if the shifted output of the ESC block is negative, then the 2’s complement value of the shifted output of the ESC block is considered. Similarly, the non-shifted output of the ESC block is negative, then the 2’s complement value of the non-shifted output of the ESC block is considered. The shifted or non-shifted number can be the product of the inputs or the previous output. Therefore, to distinguish the same, the 5th bit of the ECC block output is considered. The rest of the adder block is the same, as explained in chapter 1. Additionally, the PED and latch pair is used for synchronization.

5.2.7 2:1/4:1 multiplexers of different sizes

As the algorithm doesn’t use any programming approach, for solving the conditions, multiplexers of various sizes with multiple or single bits is considered.

5.2.8 Explanation of SFMAC using binary values

Let us consider an example to elaborate on the operation of the proposed architecture. The input numbers are as follows:

Input1=**0**01001011, Input1 exponent=**0**000

Input2=**1**01010001, Input2 exponent=**1**001

The MSB (9th bit) of input1 and input2 is the sign bit, which is highlighted in bold. Similarly, the MSB (4th bit) of input1-exponent and input2-exponent is the sign bit, which is highlighted in bold as well. In this example, input1 and input1-exponent are positive & input2 and input2-exponent are negative. On the other hand, the previous output **0**0000000000000000 with exponent as **0**000. Therefore, the previous output, as well as its exponent, is positive. The execution steps as per the example mentioned above are as follows:

1. Based on the inputs, the product of the two inputs (NUM) is calculated as **1**0001011110111011 (in 16 bit). As one of the input numbers is negative, the resultant is negative.
2. The exponent of the NUM is the addition of the exponents of the inputs. The NUM exponent result is **1**0001 (-1). The NUM exponent is represented in 5 bits because the addition of two 2-bit numbers can produce a result in 3-bit. Moreover, a negative 3-bit number requires 4 bits to represent. Additionally, the 5th bit is used to signify the sign bit.
3. If the exponents of the NUM and previous outputs are compared, then it can be observed that the exponent of NUM is -1 and exponent of previous output is +0. Therefore, the NUM is smaller than the last output, and hence, the NUM is shifted by 1 bit from the left to get the updated NUM as **1**0000101111011101.
4. The shifted NUM (i.e., **1**0000101111011101) is added with previous output **0**0000000000000000 which produces a result as **1**0000101111011101 with exponent as **0**0000. The same is shown in HEX code as -0BDD × 2^{+0} in the output curve at the 2nd rising edge of clock 8, as shown in figure 5.11.
5. In the next cycle, as the input doesn't change, the NUM remains the same, i.e., **1**0001011110111011. On the other hand, the latest value of input1-exponent and input2-exponent are **1**011 and **0**010. Therefore, it produces the NUM exponent as **1**0001 (-1).
6. As the NUM in this cycle is smaller than the previous cycle output (as the last output's exponent is more significant than NUM exponent), the NUM is shifted by 1 bit towards its right, which produces the updated NUM as **1**0000101111011101 with updated NUM exponent as **0**0000.

7. The updated NUM and previous output are added and produce the result as **1**0001011110111010 with the exponent as **0**0000. The same is shown in HEX code as -17BA × 2^{+0} in the output curve at the 3rd rising edge of clock 8. The simulation waveform is shown in figure 5.11.
8. The clock in this SFMAC architecture is applied in a pipelined manner, as mentioned below:

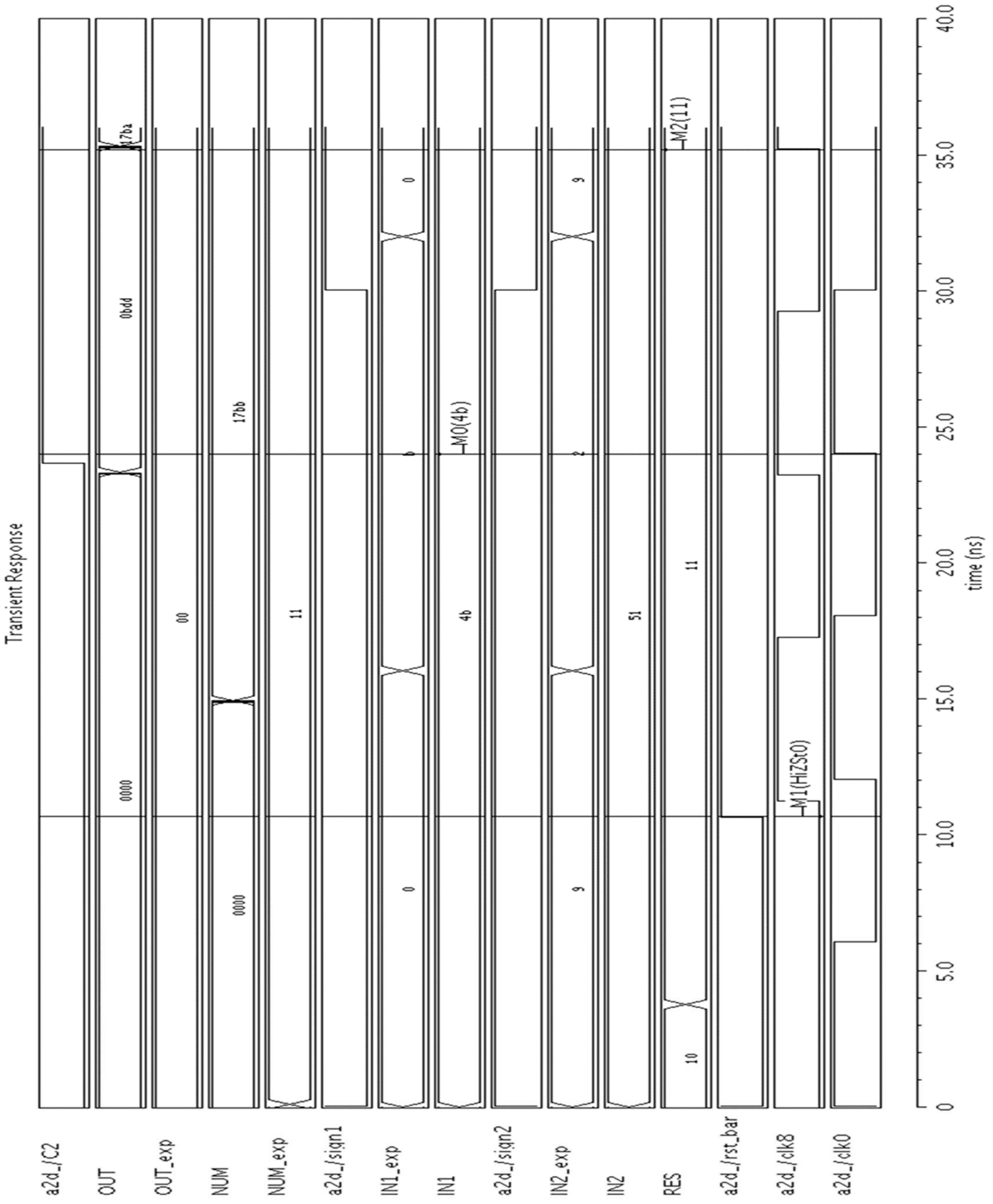


Figure 5.11: The simulation waveform of the SFMAC architecture

a. The pipeline mechanism using the clock is ensured by activating the consecutive blocks. A single clock signal is applied with a fixed clock period. But the consecutive clocks are differed by a delay of 1.4 ns. The reason for the delay is to latch the previous block's output effectively as the input for the next block. The delay of clock signals is calculated by the maximum propagation delay of the individual blocks of the SFMAC architecture, which is given by the equation 5.2.

$$\tau_{Clock\_delay} = max(\tau_{Delay\_EA}, \tau_{Delay\_UCM}, \dots \dots \dots \dots, \tau_{Delay\_reg2}) \quad (5.2)$$

τ_{delay_EA} is the propagation delay of the EA block; τ_{delay_UCM} is the propagation delay of the UCM and so on. The propagation delay of all blocks of SFMAC is shown in table 5.1.

Table 5.1: Propagation delay of the internal blocks of SFMAC architecture

Block	**Delay (in ps)**	**Inference**
Multiplier	433.7	The maximum delay from A_0 to P_{15}, considering all inputs as high.
Register	123.6	Delay from the positive edge of the clock to any of the output
Full Adder	22.4	Delay from A_0 to OUT_{15}, considering all inputs as high
EA Block	268.9 ps	Delay from $Exp2_0$ to $ExpOUT_2$, considering Exp1 as positive & Exp2 as negative
ESC Block (along with ECC block)	1367.5	With same sign bits of both the exponents (as negative or positive) in the ECC block and maximum bit shift in the ESC block
2:1 MUX	12.6	With the critical path from 'S' to 'Y'
4:1 MUX	18.9	Maximum delay occurred either in 'S_0' to 'Y', 'S_1' to 'Y' or 'S_2' to 'Y'

b. As there are a total of nine clocked blocks in this architecture, the amount of total delay required is eight times 1.4 ns (1.4 ns × 8 = 11.2 ns). This means a set of inputs latched at time 0 ns is evaluated and produces the output only after 11.2 ns. Therefore, the clock period is fixed at 12 ns (or 83.333 MHz operational frequency), so the execution of the last clock and latching on the first clock doesn’t get overlapped.
c. The EA block is enabled with clock 0.
d. The multiplier block is enabled with the clock 1 signal.
e. Clock 2 signal is used as a clock signal for the 16-bit register for the multiplier.
f. There is no clock applied to the ECC block, which produces 5-bit output.
g. Clock 3 is applied to the ESC block, which yields the shifted/non-shifted NUM or previous output. Parallelly the same clock is used to the 2:1 MUXs for updating the select line of the 16-bit 2:1 MUXs to update the true or complemented NUM/Previous output.
h. Clock 4 is applied to the 16-bit 2:1 MUXs to update the true or complemented NUM/Previous output.
i. For adding the true or 2’s complement form of shifted/non-shifted 16-bit inputs, Clock 5 is applied.
j. The 16-bit 2:1 MUX block is activated on the edges of clock 6, which choose between the true output of the full adder output or the 2’s complement output of the full adder output. The carry bit of the output of the full adder is applied as the select line.

Table 5.2: The operation of the 16-bit 4:1 MUX based on the two select lines

XOR of I/P sign bits	**Sign bit of the previous output**	**Operation**
0	0	No change or true form
0	1	Pass the output of the 16-bit 2:1 MUX as such
1	0	Pass the output of the 16-bit 2:1 MUX as such
1	1	2’s complement

k. The output of the MAC block is based on the selection of XOR of the sign bit of the inputs and sign bit in the last output. The input for the 4:1 MUX (16-bit) is the output of the 2:1 MUX (16-bit). The inputs for the 4:1 MUX (16-bit) is latched at the positive edges of clock 7. The operation of the 4:1 MUX is explained in table 5.2.

l. Finally, a 16-bit register is used at the output so that the internal glitches doesn't change the output value. The 16-bit register block is enabled with clock 8.

5.3 CONCLUSIVE REMARKS

The novel SFMAC architecture for signed-floating point MAC operation is explained in this chapter. The circuit is implemented on Cadence Virtuoso CMOS 90 nm as well as in TSMC 130 nm technology. The internal building blocks of SFMAC are also explained in detail. The step-by-step working procedure of EA block, ECC block and ESC block are also explained in this chapter. The operation of SFMAC is also explained with the help of an example also. Earlier in the literature, a full-custom based approach has never been adopted to design the floating-point synchronized MAC architecture from the primary or leaf cell. The proposed SFMAC architecture shows the simplicity of the design which primarily uses multiplexers of different sizes.

CHAPTER 6: RESULTS & DISCUSSION

The multiplier/MAC unit shown in chapter 1-5 is implemented on Cadence Virtuoso. Additionally, the UCM architecture is also prototyped on the Nexys-4 Artix-7 FPGA board. Based on the performance of the architectures, its detail analysis is done in this chapter.

6.1 IMPLEMENTATION OF UCM ARCHITECTURE & FPGA PROTOTYPING

The existing multipliers in the literature are mostly based on the Wallace tree algorithm [4, 12, 15, 18, 23, 24, 26-28, 30]. It is claimed that the multipliers based on the Wallace tree reduce the steps involved to add the partial products. Still, it uses half adder or full adder for the addition of partial products which increases the complexity of the circuit. To overcome the shortcoming of Wallace tree multiplier, the UCM architecture is proposed which uses universal compressor of N-bit size. This has ensured the proposed UCM architecture as much faster than the Wallace tree multiplier. Moreover, the proposed UCM architecture is implemented in Cadence Virtuoso 90 nm technology. This has customized the internal building blocks of the UCM and hence, highly efficient. To compare the implemented UCM with Wallace tree multiplier and array multiplier, the architectures are designed on Cadence Virtuoso 90 nm technology as well as Verilog HDL (for implementing it on Nexys-4 Artix-7 FPGA board). The result shows that the UCM is much more efficient in supply voltage as low as 600 mV for 5-bit as well as a 9-bit multiplier. The reason for implementing a 5-bit and 9-bit multiplier is to show the complexity and accuracy handling capacity of the algorithm (for which an odd number of inputs are taken). Due to lowering the supply voltage, not only the speed of operation is improved in comparison with the Wallace tree algorithm, but the power consumption has dropped substantially.

6.1.1 Power & delay analysis of novel UCM architecture

The tabular comparison of UCM and Wallace tree multiplier for 5-bit and 9-bit is shown in table 6.1 and table 6.2, respectively. As there is always a trade-off between power and delay, the average power consumption of the UCM is slightly higher than the Wallace tree multiplier. For example, the average power (a total of static as well as dynamic) consumption of the UCM at 600 mV supply voltage and for 5 × 5-bit operations is 20.32 µW, whereas, for Wallace tree multiplier, the same is recorded as 19.54 µW. Similarly, at 900 mV and for 9 × 9-bit operations, the average power consumption for implemented UCM is 355.8 µW, whereas, for the Wallace tree multiplier, it is 299.9 µW.

Table 6.1: Delay comparison of UCM versus Wallace tree for 5 × 5-bit operation

Multiplier Algorithms	**Delay at different V_{DD}**			
	0.6V	**0.7V**	**0.8V**	**0.9V**
UCM	2.769 ns	2.701 ns	2.664 ns	2.641 ns
Wallace tree	2.789 ns	2.717 ns	2.677 ns	2.652 ns
Array multiplier	Invalid outputs			

Table 6.2: Delay comparison of UCM versus Wallace tree for 9 × 9-bit operation

Multiplier Algorithms	**Delay at different V_{DD}**			
	0.6V	**0.7V**	**0.8V**	**0.9V**
UCM	2.281 ns	2.21 ns	2.171 ns	2.147 ns
Wallace tree	2.401 ns	2.298 ns	2.241 ns	2.205 ns
Array multiplier	Invalid outputs			

At the same time, there is a significant improvement of delay for the implemented UCM in comparison to the Wallace tree. The irregular structure of the Wallace tree algorithm is the leading cause of the lagging in delay. As per the Elmore formula, the wire delay is proportional to the square of its length, and the relationship is expressed by equation 6.1.

$$\tau_d=(R\times C\times L^2)/2 \tag{6.1}$$

Where 'R','C' and 'L' are the wire resistance, capacitance, and length, respectively. Hence with an irregular structure with an increased length of wire can affect the speed of operation of the circuit. On the other hand, the array multiplier could not produce any result in such low supply voltages (below 1.0V) due to which its power and delay analysis could not be performed.

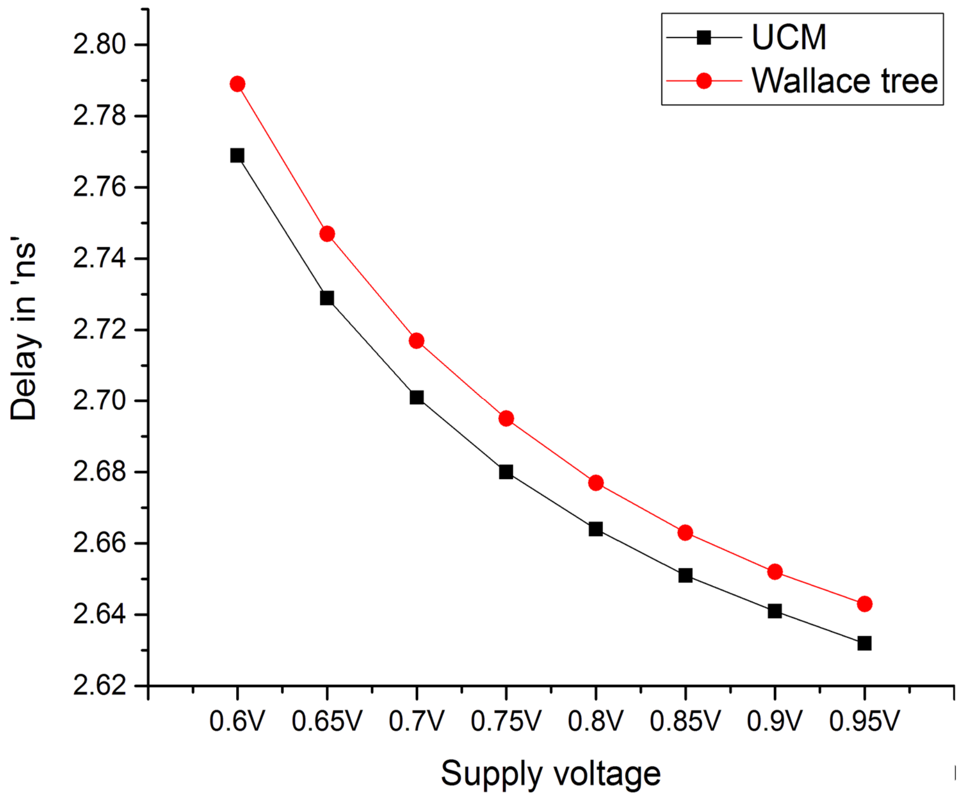


Figure 6.1: UCM and Wallace tree for 5 × 5-bit operations at voltages below 1V

The graphical representation of the delay analysis of 5 × 5 bit as well as 9 × 9-bit multipliers is shown in figure 6.1 and figure 6.2. It is clear from the graphical analysis that in 5 × 5-bit as well as 9 × 9-bit multiplication operation, the implemented UCM takes lesser time to pass the signal from input to the output (critical path). As the supply voltage drops further, the difference between the delay values of UCM and Wallace tree multiplier is significant, and it is much evident in 9 × 9-bit multiplier. For example, at 600 mV supply voltage and 9 × 9-bit multiplication, the difference in delay between Wallace tree and implemented UCM is 120 ps, on the other hand, for 5 × 5-bit multiplication, the difference in delay between the two is 20 ps. Hence it can be summarized that, as the multiplier size increases, the delay of the UCM is significantly low than the Wallace tree multiplier at ultra-low supply voltage (as low as 600 mV).

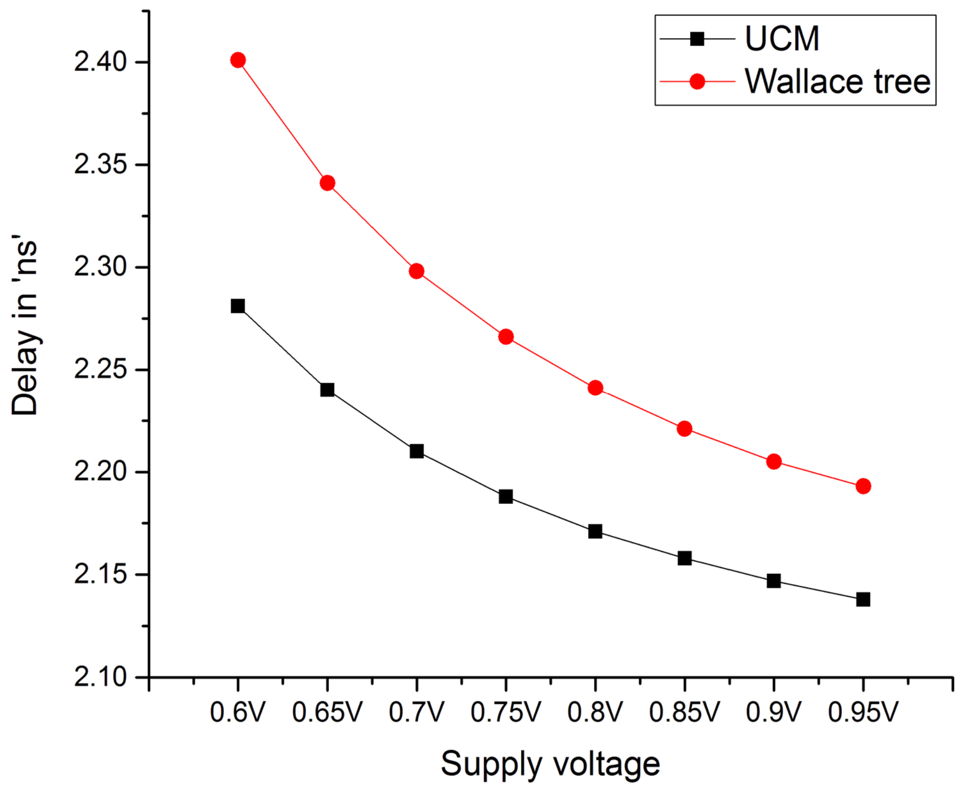


Figure 6.2: UCM and Wallace tree for 9 × 9-bit operation at voltages below 1V

6.1.2 Nexys-4 Artix-7 based FPGA Implementation

The FPGA implementation of the UCM on the Nexys-4 Artix-7 FPGA board is shown in figure 6.3. The FPGA realization is done for 5 bits as well as 9 bits. Switches along with buttons are used as the 18-bit inputs, whereas the LEDs are used as 18-bit outputs for verification of the implemented UCM. For 9-bit multiplier realization, 213 out of 63400 (approximately 0.33%) LUTs are used as logic units, whereas 36 input-output buffers (IOB) are used out of which 18 are input buffers, and 18 are output buffers. On the other hand, for 5-bit multiplier realization, 42 (approximately 0.06%) LUTs are used as logic units, and 20 input-output buffers (IOB) are used. The total on-chip power for 9-bit, as well as 5-bit UCM implementation, is 40.62 mW with junction temperature as 25.2^{o} C.

6.1.3 PVT analysis of UCM architecture

VLSI is an art of chip design, which turns specification into usable hardware. Cadence offers software for both the front end and back end projects, where the GDS-II file is eventually sent for fabrication after comprehensive design steps. But the yield of the fabricated designs is found to be very low due to process complexity (i.e., pressure, supply voltage, temperature, etcetera). The main reason for the loss of yield is the variation of the fabrication parameter between wafer and wafer. To improve design yield, the IC should be in a position to sustain extreme variation. Validation of the design cycle through PVT and 3-sigma variation becomes, therefore, essential before fabrication.

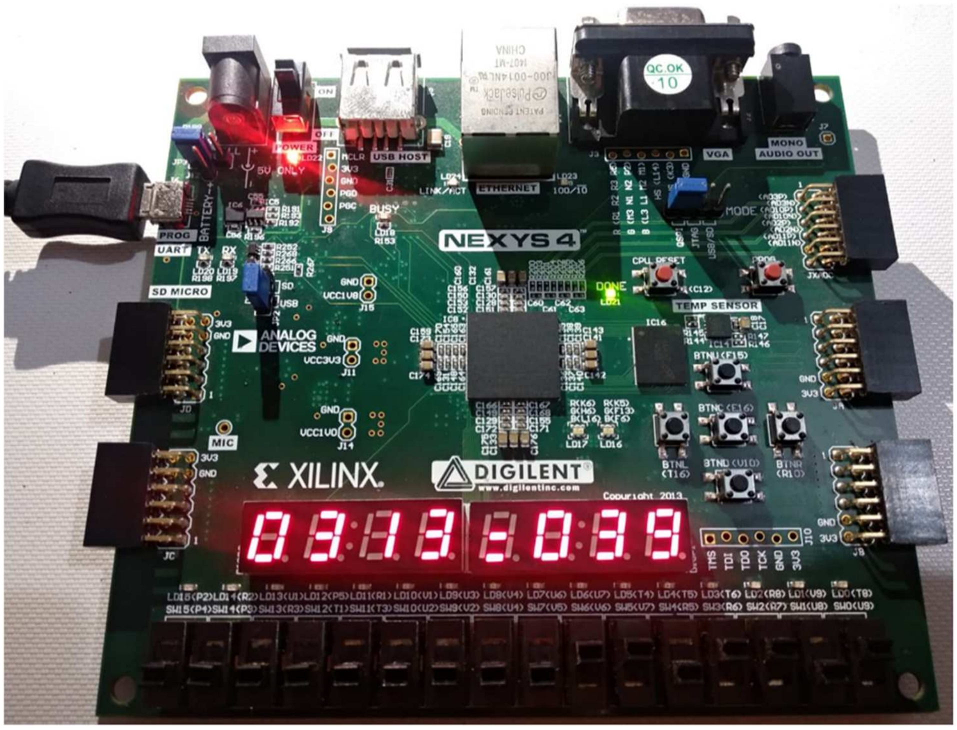


Figure 6.3: FPGA realization of the 9 × 9 UCM

Table 6.3: Delay comparison of UCM versus Wallace tree for 5 × 5-bit operations in different corners

Corners in -40°, 0° & +50° Celsius	**UCM (in ns @ 600 mV)**	**Wallace tree (in ns @ 600 mV)**	**UCM (in ns @ 900 mV)**	**Wallace tree (in ns @ 900 mV)**
Nominal (27)	2.769	2.789	2.641	2.652
FF_0 (-40)	2.665	2.677	2.59	2.597
FF_1 (0)	2.684	2.698	2.601	2.61
FF_2 (+50)	2.709	2.725	2.616	2.626
FS_0 (-40)	2.75	2.766	2.623	2.632

FS_1 (0)	2.782	2.801	2.64	2.651
FS_2 (+50)	2.822	2.845	2.663	2.676
NN_0 (-40)	2.72	2.735	2.613	2.622
NN_1 (0)	2.749	2.767	2.629	2.64
NN_2 (+50)	2.786	2.809	2.651	2.663
SF_0 (-40)	2.728	2.746	2.617	2.627
SF_1 (0)	2.76	2.782	2.635	2.647
SF_2 (+50)	2.802	2.829	2.658	2.673
SS_0 (-40)	2.826	2.849	2.656	2.668
SS_1 (0)	2.875	2.902	2.682	2.697
SS_2 (+50)	2.937	2.97	2.716	2.734

A PVT analysis is performed at different corners (Fast-Fast, Fast-Slow, Normal-Normal, Slow-Fast, and Slow-Slow) and three different extreme temperatures (-40^{o}, 0^{o} and +50^{o} Celsius) to validate the performance of the UCM architecture further. Table 6.3 and table 6.4 shows the delay comparison of UCM and Wallace tree 5 × 5-bit and 9 × 9-bit architecture respectively at 0.6V and 0.9V supply voltage in different corners along with variation in temperature (-40^{o}, 0^{o} and +50^{o} Celsius)

Table 6.4: Delay comparison of UCM versus Wallace tree for 9 × 9-bit operations in different corners

Corners in -40^{o}, 0^{o} & +50^{o} Celsius	**UCM (in ns @ 600 mV)**	**Wallace tree (in ns @ 600 mV)**	**UCM (in ns @ 900 mV)**	**Wallace tree (in ns @ 900 mV)**
Nominal (27)	2.281	2.401	2.147	2.205
FF_0 (-40)	2.171	2.239	1.138	1.195
FF_1 (0)	2.192	2.27	1.153	1.222
FF_2 (+50)	2.218	2.31	1.247	1.257
FS_0 (-40)	2.258	2.353	2.126	2.171
FS_1 (0)	2.291	2.402	2.145	2.198
FS_2 (+50)	2.334	2.463	2.169	2.233
NN_0 (-40)	2.228	2.322	1.235	1.252
NN_1 (0)	2.259	2.369	2.134	2.187
NN_2 (+50)	2.3	2.43	2.157	2.221
SF_0 (-40)	2.239	2.351	2.123	1.259
SF_1 (0)	2.274	2.406	1.421	1.289
SF_2 (+50)	2.32	2.479	2.168	1.439
SS_0 (-40)	2.339	2.484	2.162	2.227
SS_1 (0)	2.391	2.561	2.19	2.268
SS_2 (+50)	2.456	2.659	2.227	2.323

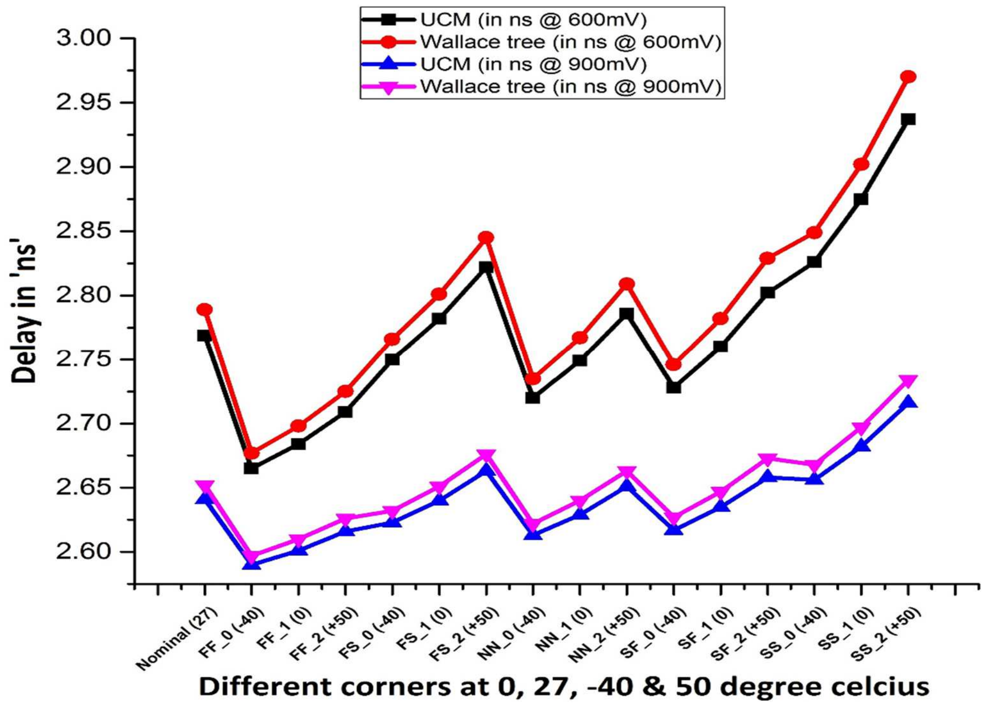


Figure 6.4: PVT comparison of delay of UCM and Wallace tree for 5 × 5-bit operations at 0.6V and 0.9V in different corners

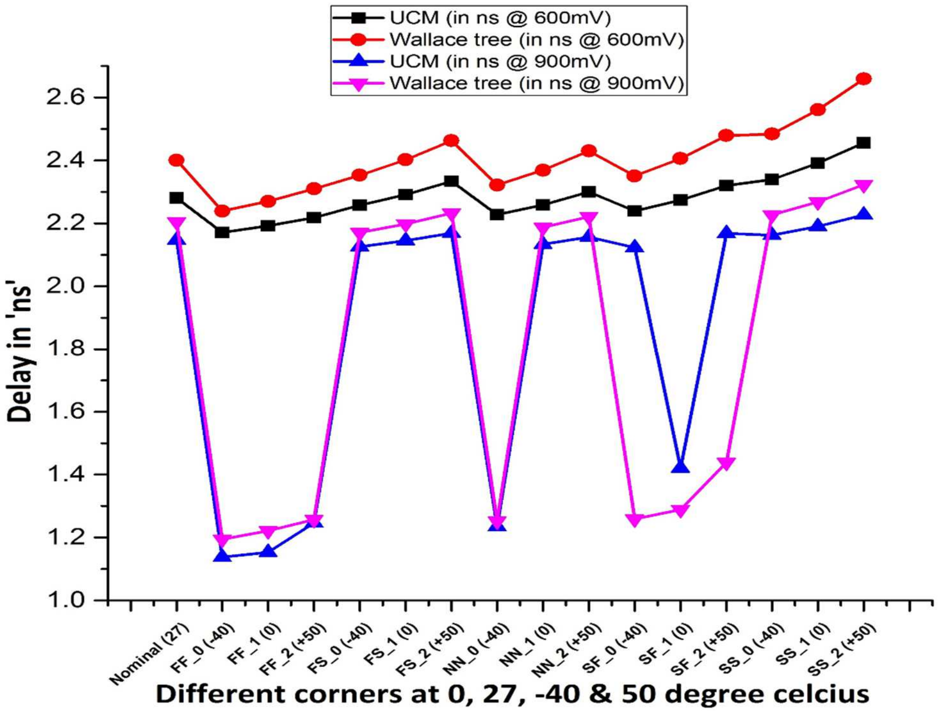


Figure 6.5: PVT comparison of delay of UCM and Wallace tree for 9 × 9-bit operations at 0.6V and 0.9V in different corners

The graphs in figure 6.4 and figure 6.5 clearly show that the delay in UCM architecture is significantly improved compared to the Wallace tree architecture for 5 x 5 bit as well as 9 x 9-bit multiplication. Most importantly, the UCM architecture proves to be the better performer than Wallace tree architecture at ultra-low supply voltages for 5-bit multiplication at different corners and extreme temperatures. On the other hand, the delay of UCM has a much more significant drop compared to the Wallace tree at 600 mV (at different corners and extreme temperatures) for 9-bit multiplication. While the UCM architecture delay appears to be slightly higher than the Wallace tree in the slow-fast (SF) corner at -40^o, 0^o, and +50^o Celsius for 9-bit multiplication at 900 mV. The reason for the same might be the use of different processes at the SF corner. The minimum and maximum delay for 5 × 5-bit multiplication using UCM architecture at 600 mV are 2.665 ns and 2.937 ns, respectively, as shown in table 6.3. Whereas 2.677 ns and 2.97 ns are the same for Wallace tree, respectively. Likewise, 2.59 ns and 2.716 ns are the minimum and maximum delay for 5 × 5-bit multiplication using UCM architecture at 900 mV, which are 2.597 ns and 2.734 ns respectively for Wallace tree. It can be observed from table 6.4 that for 9 × 9-bit multiplication using UCM architecture at 600 mV, the minimum and maximum delays are 2.171 ns and 2.456 ns respectively. In contrast, for the Wallace tree, the values are 2.239 ns and 2.659 ns. On the other hand, for 9 × 9-bit multiplication using UCM architecture at 900 mV, the minimum and maximum delays are 1.138 ns and 2.227 ns, respectively. In contrast, for the Wallace tree, the values are 1.195 ns and 2.323 ns.

6.2 POWER, DELAY & AREA COMPARISON OF NOVEL UMAC, USMAC, SMAC & SFMAC ARCHITECTURES

The proposed UMAC, USMAC, SMAC (fixed-point), and SFMAC (floating-point) architectures are implemented at the Cadence Virtuoso 90 nm technology. The power consumption of the proposed designs is measured using the Cadence Spectra tool. The detailed report of the static power, average power, and area are shown in table 6.5. Static power is assessed for 2V supply voltage, while average power is measured for 20 ns simulation period and 333.33 MHz frequency. The area, on the other hand, is measured in terms of total transistor count.

Table 6.5: Comparison of UMAC, USMAC, SMAC and SFMAC architectures with 2V supply voltage and 20 ns simulation period

Architecture	**Static power (at V_{DD}=2V)**	**Average power (at V_{DD}=2V and simulation period=20 ns)**	**Area (Total number of transistors)**
UMAC	3072 µW	2253 µW	4556
USMAC	758 µW	2905 µW	5744
SMAC	1721 µW	7317 µW	10928
SFMAC	4854 µW	26950 µW	25783

As discussed earlier, UMAC architecture's static power consumption is the highest as block enabling is not being used in this architecture. On the other hand, since the SMAC architecture area is about two times greater (in terms of the number of transistors) than the USMAC architecture, the static and thus the average power consumption of SMAC architecture is higher than that of the USMAC architecture. Moreover, the static and the average dynamic power are the highest for SFMAC architecture, as shown in the comparison table 6.5. The reason for the same is the total number of transistor count in SFMAC is almost five times higher than that of USMAC and 2.5 times higher than that of SMAC architectures. The graphical analysis is shown in figure 6.6.

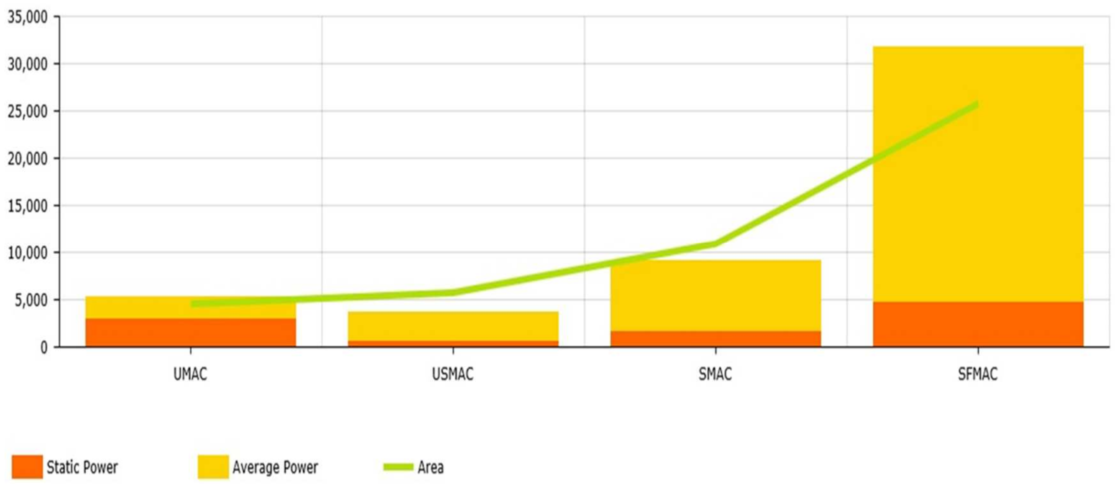


Figure 6.6: Graphical comparison of UMAC, USMAC, SMAC and SFMAC architectures with 2V supply voltage and 20 ns simulation period

Furthermore, table 6.6 shows a power comparison of SFMAC architecture at different CMOS technologies in a specific input vector. The simulation period is kept as 40 ns because:

1. the reset signal (active low) is low till 10.8 ns
2. the clock signals have a time period of 12 ns

Table 6.6: Comparison of SFMAC at supply voltage 2V and simulation period 40 ns in CMOS GPDK 90 nm and TSMC 130 nm technology

Architecture	**Static Power in µW (for V_{DD}=2V)**	**Average power in µW (for V_{DD}=2V and simulation period=40 ns)**	**Area (Total number of transistors)**
GPDK 90 nm CMOS Technology			
SFMAC	476.94	7980	25783
TSMC 130 nm CMOS Technology			
SFMAC	2398.76	25990	25783

Therefore, till 23.2 ns, the output signal remains at '0'. The SFMAC architecture is not only implemented in GPDK 90 nm but also in TSMC 130 nm CMOS Technology. The power consumption of the implemented designs is calculated using Cadence Spectra Tool.

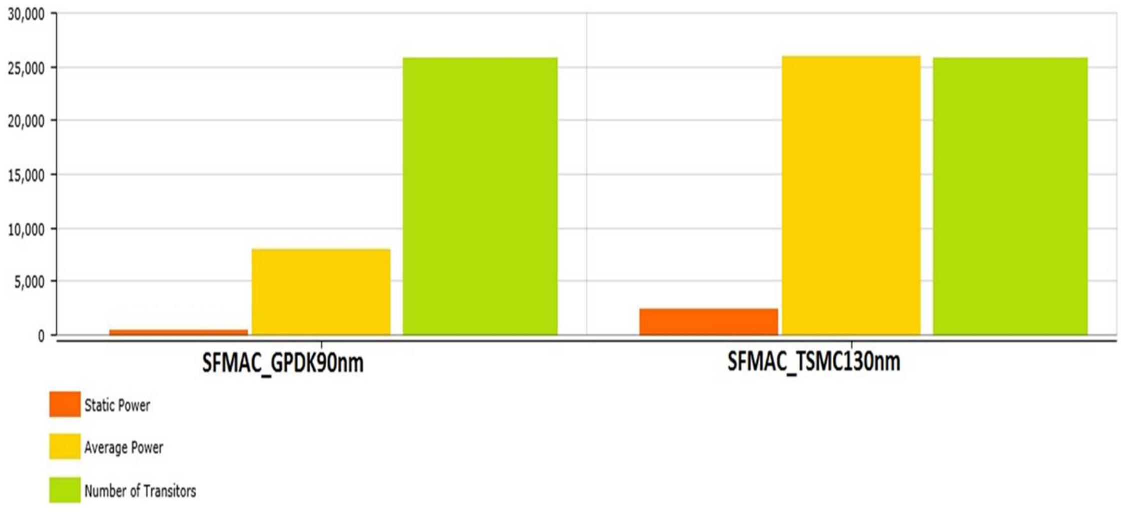


Figure 6.7: Graphical analysis of the static power, average power and area of SFMAC in CMOS GPDK 90 nm and TSMC 130 nm technology

Figure 6.7 shows the graphical analysis of the static power, average power, and area of SFMAC in CMOS GPDK 90 nm and TSMC 130 nm technology. The static power is evaluated for 2V supply voltage, whereas the average power is measured for a simulation period of 40 ns and at a frequency of 83.33 MHz. The average dynamic power consumption of the SFMAC in TSMC 130 nm is higher than GPDK 90 nm because the transistor sizing is higher in 130 nm technology, which affects the load capacitance (C_{load}). The average dynamic power of a CMOS circuit is given by equation 6.2.

$$P_{avg} = \alpha_T C_{load} {V_{DD}}^2 f_{CLK} \qquad (6.2)$$

Similarly, the static power consumption is also a function of device geometry. Therefore, a circuit consisting of a higher device dimension has higher static power consumption.

6.2.1 Comparison with existing architectures

It is challenging to compare the proposed MAC architectures with those that are already available in the literature because most of the available architectures in the literature have used HDL based approach. On the other hand, the proposed architectures are implemented in Cadence Virtuoso 90 nm environment. Moreover, almost 99% (80 out of 81 papers) of the architectures available in the literature have neither implemented for signed operation nor floating-point designs. For example, [1, 6, 8, 9, 79] etcetera are unsigned-fixed-point MAC architectures, which are implemented on HDLs such as Verilog or VHDL. Moreover, some of the architectures haven’t even specifies the technology used in the design. Therefore, the comparison of proposed MAC architectures with the existing MAC architecture becomes challenging.

Though some architectures in the literature have used the clocking signals for the accumulation of data only (in the register or accumulator), most of the architectures haven't used any clocking signal. For example, [8, 71, 74] etcetera are existing MAC architecture but without proper synchronization. Any circuit in asynchronous mode can't be implemented in a real-time application. Therefore, the practical applicability of such design needs to be further tested.

From the literature, a few existing MAC architectures are found suitable to compare with the proposed MAC architectures. Though all parameters of comparison (such as technology specified, operating frequency, supply voltage, tool used, size and type of MAC etcetera) are not matching but most of the parameters are common while comparing the existing and proposed MAC architectures. Table 6.7 shows the comparison, where most of the architectures are compared with UMAC architecture, and only one architecture (i.e. [87]), which is implemented for floating-point signed operation is compared with proposed SFMAC architecture. The differences are visible from table 6.7 that the architectures in [55], [7] and [69] have a significantly higher static as well as average power (in mW) than proposed UMAC architecture. In [56] and [60], the performance is evaluated in 65nm and 90 nm technologies for 16-bit operations at 1.1V and 1V, respectively. It is clear from table 6.7 that the power consumption of [56] is significantly higher than UMAC. On the other hand, the architecture in [60] operates in 1V supply voltage with operating frequency 100 MHz, and therefore, a direct comparison can't be made with UMAC. Though the architecture in [75] is implemented in 180nm technology and 1.8V supply voltage for 16-MAC operation, the power consumption is way more than the UMAC architecture. For [80], the implementation is done for 1-bit MAC operation in 32nm CMOS and CNTFET technology, and hence, comparison with 8-bit UMAC is not relevant. The architecture in [87] is compared with proposed SFMAC architecture, and the analysis shows that the performance of SFMAC is much better in terms of power consumption.

Table 6.7: Performance comparison of Proposed MAC architecture with existing architectures

Sl No.	**Existing work in**	**Existing architecture description**	**Implementation on**	**Power Consumption**
1	[55]	Pipelined Multiply Accumulate Unit (fixed-point) in 180nm technology, 1.8V at 83.3 MHz & 8 × 8 bit operation	Cadence Virtuoso	50.26 mW
2	[7]	Multiply Accumulate Unit (fixed-point) in 180nm technology, 1.8V at 217 MHz & 64 × 64 bit operation	Verilog HDL	177.732 mW
3	[56]	Pipelined Multiply Accumulate Unit (fixed-point) in 65nm technology, 1.1V at 591 MHz & 16 × 16 bit operation	VHDL	8.2 mW

4	[60]	Multiply Accumulate Unit (fixed-point) in 90nm technology, 1V at 100 MHz & 16 × 16 bit operation	HDL in Cadence’s HSPICE simulator	1.506 mW	
5	[69]	Pipelined Multiply Accumulate Unit (fixed-point) in 180nm technology, 1.8V & 8 × 8 bit operation	HDL in Synopsys Design Compiler	**Dynamic Power**	**Static Power**
				3.627 mW	2.010 mW
6	[75]	Multiply Accumulate Unit (fixed-point) in 180nm technology, 1.8V at 5 MHz & 16 × 16 bit operation	Verilog HDL	**MAC using Booth**	**MAC using Vedic**
				493.648 mW	1765.241 mW
7	[80]	Multiply Accumulate Unit(fixed-point) in 32nm CMOS & CNTFET technology & 1 × 1 bit operation	**-----------**	**CMOS Tech**	**CNTFET Tech**
				0.9902 mW	0.6335 mW
8	[87]	Fixed/Floating-Point Multiply Accumulate Unit in 90nm technology for 16-bit half-precision multiplication	VHDL	14.07 mW	
Sl No.	**Proposed architecture**	**Proposed architecture description**	**Implementation on**	**Power Dissipation**	
1	UMAC	Unsigned MAC architecture in 90nm tech., 2V at 333.33 MHz & 8x8 bit operation	Cadence Virtuoso 90nm CMOS	**Static Power**	**Average Power**
				3.072 mW	2.253 mW
2	USMAC	Unsigned Synchronized MAC architecture in 90nm tech., 2V at 333.33 MHz & 8x8 bit operation	Cadence Virtuoso 90nm CMOS	**Static Power**	**Average Power**
				0.758 mW	2.905 mW
3	SMAC	Signed MAC (synchronous) architecture in 90nm tech., 2V at 333.33 MHz & 8x8 bit operation	Cadence Virtuoso 90nm CMOS	**Static Power**	**Average Power**
				1.721 mW	7.317 mW
4	SFMAC	Signed Floating-Point MAC architecture in 90nm tech., 2V at 83.33 MHz & 8x8 bit operation	Cadence Virtuoso 90nm CMOS	**Static Power**	**Average Power**
				0.476 mW	7.98 mW

CHAPTER 7: CONCLUSION & FUTURE WORK

7.1 CONCLUSION

The summary of this research work along with new possibilities for further improvement is presented in this chapter. The objective of this research work is to design suitable low power high-speed MAC unit for signed-floating operation. For achieving the said objective, the Unsigned MAC (UMAC) architecture, the Unsigned Synchronized MAC (USMAC) architecture, Signed MAC (SMAC) architecture with synchronization and the Signed Floating-point MAC (SFMAC) architecture with synchronization are designed and implemented. As the multiplier is said to be the heart of the MAC architecture, a novel Universal Compressor based Multiplier (UCM) is designed and implemented. This work examines, discusses and uses fast adder and multiplication schemes in the design and development of proposed novel UCM architecture. The prototype of the proposed UCM architecture has been implemented on Nexys-4 Artix-7 FPGA board using Xilinx Vivado 17.4 and Xilinx ISIM simulator for simulation.

Further, a detailed analysis of the novel UCM along with Wallace tree multiplier and array multiplier at ultra-low supply voltages (as low as 600 mV) is performed on Cadence Spectre tool in GPDK 90 nm technology. A significant improvement in terms of delay of the proposed UCM in comparison to Wallace tree multiplier is sighted. The irregular structure of the Wallace tree algorithm is the leading cause for the lagging in delay, as an asymmetrical structure with increased length of wire can affect the speed of operation of the circuit. On the other hand, the array multiplier could not produce any result in such low supply voltages (below 1V) due to which its power and delay analysis could not be performed. The UCM architecture is further analyzed by performing a PVT analysis at different corners (Fast-Fast, Fast-Slow, Normal-Normal, Slow-Fast and Slow-Slow) and three different extreme temperatures (-40^o, 0^o and +50^o Celsius).

The block enabling schemes along with pipelining is used to model the MAC design power-efficient. Block enabling technique facilitates saving of electrical power, used by digital signal processors, by reducing the switching activity ‘α’. It ensures power saving of the MAC architecture, by turning on a functional logic block only when required. For achieving pipelining, the time between each clock signal is set such that when the registers are clocked, the data written to them is the final result of the preceding stage.

For proper implementation of floating-point MAC implementation, each input of the Signed Floating-point MAC (SFMAC) is represented in 13 bits, in which two bits are reserved for the sign bits of the number, and its exponent. The sign bit has a provision to be represented in the form of '0' or '1' based on positive or negative number representation, respectively. Remaining eleven bits are used for 8-bit binary representation and 3-bit exponent representation in binary. Therefore, the input numbers have a range from $–(0.11111111)_2 \times 2^{+3}$ to $+(0.11111111)_2 \times 2^{+3}$ and hence, the range of the inputs of the current SFMAC architecture in a decimal number system is from $–(7.96872)_{10}$ to $+(7.96872)_{10}$. Different MAC architectures are designed and analyzed using Cadence Spectre tool to validate its power/delay performance. The CMOS 90 nm technology and TSMC 130 nm technologies are used for different MAC architecture designs.

The comparison of power for the proposed UMAC, USMAC, SMAC and SFMAC architectures are analyzed for a fixed input vector with 2V supply voltage and 20 ns simulation period. The clock frequency is maintained at 333.33 MHz. It is analyzed from the comparison that the static power consumption for UMAC architecture outperforms other proposed architectures as no block enabling is used in the UMAC architecture. On the other hand, as the area of the SMAC architecture is approximately two times larger (in terms of the number of transistors) than the USMAC architecture, the static and therefore the average power consumption of SMAC architecture is higher than the USMAC architecture. The static and the average dynamic power of SFMAC are maximum among all other proposed architectures because the total number of transistor count in SFMAC is almost five times higher than that of USMAC and 2.5 times higher than that of SMAC architectures.

Furthermore, a power comparison of SFMAC architecture at different CMOS technologies (TSMC 130 nm and GPDK 90 nm) in a specific input vector is studied at a frequency of 83.33 MHz. It is analyzed from the comparison that the average dynamic power, as well as the static power consumption of the SFMAC in TSMC 130 nm, is higher than GPDK 90 nm because of the transistor sizing. A detailed comparison is depicted in chapter 6 to analyze the efficiency of the MAC architectures over the existing architectures. The comparison shows a significant improvement in terms of static as well as average power for UMAC, USMAC, SMAC and SFMAC architectures over the existing architectures.

7.2 FUTURE WORK DIRECTIONS

At different abstraction level, a detailed power calculation can be done and hence the possibility of power reduction can be considered. Furthermore, a parametric analysis, along with Monte-Carlo analysis, will give a detailed picture of the proposed architectures at extreme corners and extreme supply voltage as well as temperature. Much more optimization at the abstract level can improve the performance of the proposed MAC architectures in terms of delay, power and PDP. An optimized layout design would provide an opportunity for post-layout simulation and hence ASIC fabrication.

REFERENCES

[1] A. Abdelgawad and M. Bayoumi, "High speed and area-efficient multiply accumulate (MAC) unit for digital signal processing applications," in *IEEE International Symposium on Circuits and Systems*, New Orleans, LA, pp. 3199-3202: IEEE, 2007.

[2] W. C. Hsieh and W. Hwang, "Adaptive power control technique on power-gated circuitries," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 7, pp. 1167-1180, 2011.

[3] P. Bhattacharyya, B. Kundu, S. Ghosh, V. Kumar, and A. Dandapat, "Performance analysis of a low-power high-speed hybrid 1-bit full adder circuit," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 10, pp. 2001-2008, 2014.

[4] M. J. Liao, C. F. Su, C. Y. Chang, and A. C. H. Wu, "A carry-select-adder optimization technique for high-performance booth-encoded wallace-tree multipliers," in *IEEE International Symposium on Circuits and Systems, ISCAS-2002*, Phoenix-Scottsdale, AZ, USA, pp. 81-84, 2002.

[5] N. J. Babu and R. Sarma, "A novel low power multiply–accumulate (MAC) unit design for fixed point signed numbers," *Advances in Intelligent Systems and Computing*, vol. 394, no. 1, pp. 675-690, 2016.

[6] S. Deepak and B. J. Kailath, "Optimized MAC unit design," in *IEEE International Conference on Electron Devices and Solid State Circuit (EDSSC)*, Bangkok, Thailand, pp. 1-4: IEEE, 2012.

[7] P. Jagadees, S. Ravi, and K. H. Mallikarjun, "Design of a high performance 64 bit MAC unit," in *International Conference on Circuits, Power and Computing Technologies*, Nagercoil, India, pp. 782-786: IEEE, 2013.

[8] T. Francis, T. Joseph, and J. K. Antony, "Modified MAC unit for low power high speed DSP application using multiplier with bypassing technique and optimized adders," in *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, Tiruchengode, India, pp. 1-4, 2013.

[9] R. Warrier, C. H. Vun, and W. Zhang, "A low-power pipelined MAC architecture using baugh-wooley based multiplier," in *IEEE 3rd Global Conference on Consumer Electronics (GCCE),* Tokyo, Japan, pp. 505-506, 2014.

[10] B. J. Xia, P. Liu, and Q. D. Yao, "New method for high performance multiply-accumulator design," *Journal of Zhejiang University Science,* vol. 10, no. 7, pp. 1067-1074, 2009.

[11] C. S. Wallace, "A suggestion for a fast multiplier," *IEEE Transactions on Electronic Computers,* vol. 13, no. 1, pp. 14-17, 1964.

[12] M. J. Rao and S. Dubey, "A high speed and area efficient booth recoded wallace tree multiplier for fast arithmetic circuits," in *Asia Pacific Conference on Postgraduate Research in Microelectronics & Electronics (PRIMEASIA),* Hyderabad, India, pp. 220-223, 2012.

[13] A. K. Singh, B. P. De, and S. Maity, "Design and comparison of multipliers using different logic styles," *International Journal of Soft Computing and Engineering (IJSCE),* vol. 2, no. 2, pp. 374-379, 2012.

[14] S. Shanthala and S. Y. Kulkarni, "VLSI design and implementation of low power MAC unit with block enabling technique," *European Journal of Scientific Research,* vol. 30, no. 4, pp. 620-630, 2009.

[15] N. Itoh, Y. Naemura, H. Makino, Y. Nakase, T. Yoshihara, and Y. Horiba, "A 600-MHz 54-bit multiplier with rectangular-styled wallace tree," *IEEE Journal of Solid-State Circuits,* vol. 36, no. 2, pp. 249-257, 2001.

[16] T. Onomi, K. Yanagisawa, M. Seki, and K. N. Ima, "Phase-mode pipelined parallel multiplier," *IEEE Transactions on Applied Superconductivity,* vol. 11, no. 1, pp. 541-544, 2001.

[17] D. Guevorkian, A. Launiainen, V. Lappalainen, P. Liuha, and K. Punkka, "A method for designing high-radix multiplier-based processing units for multimedia applications," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 15, no. 5, pp. 716-725, 2005.

[18] T. Y. Kuo and J. S. Wang, "A low-voltage latch-adder based tree multiplier," in *IEEE International Symposium on Circuits and Systems,* Seattle, WA, pp. 804-807, 2008.

[19] L. Chen, X. Y. Tian, and X. J. Zhao, "Improved multiplier of CSD used in digital signal processing," in *International Conference on Machine Learning and Cybernetics*, Kunming, pp. 2905-2908: IEEE, 2008.

[20] Q. Yi and H. Jing, "An improved design method for multi-bits reused booth multiplier," in *4th International Conference on Computer Science & Education,* Nanning, China, pp. 1914-1916: IEEE, 2009.

[21] M. Nachtigal, H. Thapliyal, and N. Ranganathan, "Design of a reversible single precision floating point multiplier based on operand decomposition," in *10th IEEE conference on Nanotechnology*, Kintex, Korea, pp. 233-237, 2010.

[22] L. A. Sousa, "Algorithm for modulo (2"+ 1) multiplication," *Electronics Letters,* vol. 39, no. 9, pp. 752-754, 2013.

[23] S. Khan, S. Kakde, and Y. Suryawanshi, "VLSI implementation of reduced complexity wallace multiplier using energy efficient CMOS full adder," in *IEEE International Conference on Computational Intelligence and Computing Research*, Enathi, India, pp. 1-4: IEEE, 2013.

[24] R. D. Kshirsagar, E. V. Aishwarya, A. S. Vishwanath, and P. Jayakrishnan, "Implementation of pipelined booth encoded wallace tree multiplier architecture," in *International Conference on Communication and Green Computing Conservation of Energy (ICGCE)*, Chennai, India, pp. 199-204, 2013.

[25] M. Jayaprakash, A. Shanmugam, and M. Mohamed, "Design and analysis of low power hybrid adder," *Journal of Theoretical and Applied Information Technology,* vol. 58, no. 3, pp. 618-622, 2013.

[26] D. Paradhasaradhi, M. Prashanthi, and N. Vivek, "Modified wallace tree multiplier using efficient square root carry select adder," in *International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE)*, Coimbatore, India, pp. 1-5: IEEE, 2014.

[27] X. V. Luu, T. T. Hoang, T. T. Bui, and A. V. Dinh-Duc, "A high-speed unsigned 32-bit multiplier based on booth-encoder and wallace-tree modifications," in *International Conference on Advanced Technologies for Communications (ATC'14)*, Hanoi, Vietnam, pp. 739-744, 2014.

[28] B. N. M. Reddy, H. N. Sheshagiri, and S. Shanthala, "Implementation of low power 8-Bit multiplier using gate diffusion input logic," in *IEEE 17th International Conference on Computational Science and Engineering,* Chengdu, China, pp. 1868-1871, 2014.

[29] S. Srinitha and B. Sargunam, "Area effective and speed optimized fused add-multiply unit," in *2nd International Conference on Innovations in Information Embedded and Communication Systems,* Coimbatore, India, pp. 1-6, 2015.

[30] K. B. Jaiswal, N. Kumar, P. Seshadri, and G. Lakshminarayanan, "Low power wallace tree multiplier using modified full adder," in *3rd International Conference on Signal Processing, Communication and Networking (ICSCN),* Chennai, India, pp. 1-4: IEEE, 2015.

[31] M. Shoba and R. Nakkeeran, "Energy and area efficient hierarchy multiplier architecture based on vedic mathematics and GDI logic," *International Journal on Engineering Science and Technology,* vol. 20, no. 1, pp. 321-331, 2017.

[32] E. Ozcan and S. S. Erdem, "A fast digit based Montgomery multiplier designed for FPGAs with DSP resources," *Microprocessors and Microsystems,* vol. 62, no. 1, pp. 12-19, 2018.

[33] R. D. Rose, P. Romero, and M. Lanuzzaa, "Double-precision dual mode logic carry-save multiplier," *Integration,* vol. 64, no. 1, pp. 71-77, 2019.

[34] H. Suzuki, H. Morinaka, H. Makino, Y. Nakase, and K. Mashiko, "Leading-zero anticipatory logic for high-speed floating point addition," *IEEE Journal of Solid-State Circuits,* vol. 31, no. 8, pp. 1157-1164, 1996.

[35] R. V. K. Pillai, D. Al-Khalili, and A. J. Al-Khalili, "Low power architecture for floating point MAC fusion," *IEE Proceedings-Computers and Digital Techniques,* pp. 288-296, 2000.

[36] W. G. Natter and B. Nowrouzian, "A novel algorithm for signed-digit online multiply-accumulate operation and its purely signed-binary hardware implementation," in *IEEE International Symposium on Circuits and Systems,* vol. 5, pp. 329-332: IEEE, 2000.

[37] W. G. Natter and B. Nowrouzian, "A novel multiplier recoding technique and its application to the development of a high-speed parallel online multiply-

accumulate architecture," in *IEEE International Symposium on Circuits and Systems*, vol. 2, pp. 713-716: IEEE, 2001.

[38] W. P. du Plessis, "Optimal MAC structures in an FPGA," in *IEEE AFRICON 6th Africon Conference in Africa*, pp. 333-336: IEEE, 2002.

[39] S. R. Huang and L. R. Dung, "VLSI implememtation for MAC-level DWT architecture," in *IEEE Computer Society Annual Symposium on VLSI. New Paradigms for VLSI Systems Design. ISVLSI 2002*, pp. 101-106: IEEE, 2002.

[40] A. B. Premkumar, A. S. Madhukumar, and C. T. Lau, "MAC units for matched filters in DS-CDMA systems," *IEEE Transactions on Broadcasting,* vol. 48, no. 1, pp. 52-57, 2002.

[41] Z. Tian, D. S. Yu, and Y. L. Qiu, "A high effective algorithm of 32-bit multiply and MAC instructions' VLSI implementation with 32 x 8 multiplier-accumulator in DSP applications " in *6th International Conference on Signal Processing*, vol. 1, pp. 5-8: IEEE, 2002.

[42] Y. Liao and D. B. Roberts, "A high-performance and low-power 32-bit multiply-accumulate unit with single-instruction-multiple-data (SIMD) feature," *IEEE Journal of Solid-State Circuits,* vol. 37, no. 7, pp. 926-931, 2002.

[43] J. T. Kao, M. Miyazaki, and A. R. Chandrakasan, "A 175-mV multiply-accumulate unit using an adaptive supply voltage and body bias architecture," *IEEE journal of solid-state circuits,* vol. 37, no. 11, pp. 1545-1554, 2002.

[44] D. Suvakovic, C. Salama, and T. Andre, "Energy efficient adiabatic multiplier-accumulator design," *Journal of VLSI Signal Processing,* vol. 33, no. 1, pp. 83-103, 2003.

[45] B. Shim and N. R. Shanbhag, "Complexity analysis of multicarrier and single-carrier systems for very high-speed digital subscriber line," *IEEE Transactions on Signal Processing,* vol. 51, no. 1, pp. 282-292, 2003.

[46] Y. L. Y. Li and J. C. J. Chen, "A reconfigurable architecture of a high performance 32-bit MAC unit for embedded DSP," in *5th International Conference on ASIC,* vol. 2, pp. 1285-1288: IEEE, 2003.

[47] J. Grossschadl and G. A. Kamendje, "A single-cycle (32 x 32+ 32 + 64)-bit multiply/accumulate unit for digital signal processing and public-key

cryptography," in *10th IEEE International Conference on Electronics, Circuits and Systems*, vol. 2, pp. 739-742: IEEE, 2003.

[48] I. Kataeva, H. Zhao, H. Engseth, E. Tolkacheva, and A. Kidiyarova-Shevchenko, "RSFQ digital signal processor for interference cancellation," *IEEE Transactions on Applied Superconductivity*, vol. 15, no. 2, pp. 405-410, 2005.

[49] P. I. Bunyk, Q. P. Herr, and M. W. Johnson, "Demonstration of multiply-accumulate unit for programmable band-pass ADC," *IEEE Transactions on Applied Superconductivity*, vol. 15, no. 2, pp. 392-395, 2005.

[50] J. S. Cardoso and C.-R. L., "Accumulator size minimization for a fast cumulant-based motion estimator," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 12, pp. 1660-1664, 2005.

[51] A. Danysh and D. Tan, "Architecture and implementation of a Vector/SIMD multiply-accumulate unit," *IEEE Transactions on Computers*, vol. 54, no. 3, pp. 284-293, 2005.

[52] S. R. Vangal, Y. V. Hoskote, N. Y. Borkar, and A. Alvandpour, "A 6.2-GFlops floating-point multiply-accumulator with conditional normalization," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 10, pp. 2314-2323, 2006.

[53] I. Kataeva, H. Engseth, and A. Kidiyarova-Shevchenko, "Scalable matrix multiplication with hybrid CMOS-RSFQ digital signal processor," *IEEE Transactions on Applied Superconductivity*, vol. 17, no. 2, pp. 486-489, 2007.

[54] Y. Voronenko and M. Püschel, "Mechanical derivation of fused multiply–add algorithms for linear transforms," *IEEE Transactions on Signal Processing*, vol. 55, no. 9, pp. 4458-4473, 2007.

[55] S. Shanthala, C. P. Raj, and S. Y. Kulkarni, "Design and VLSI implementation of pipelined multiply accumulate unit," in *Second International Conference on Emerging Trends in Engineering and Technology*, Nagpur, India, pp. 381-386: IEEE, 2009.

[56] T. T. Hoang, M. Själander, and P. Larsson-Edefors, "A high-speed, energy-efficient two-cycle multiply-accumulate (MAC) architecture and its application to a double-throughput MAC unit," *IEEE Transactions on Circuits and Systems-I: Regular Papers*, vol. 57, no. 12, pp. 3073-3081, 2010.

[57] H. Quan, R. Xiao, K. You, X. Zeng, and Z. Yu, "A novel vector/SIMD multiply-accumulate unit based on reconfigurable booth array," in *10th IEEE International Conference on Solid-State and Integrated Circuit Technology*, pp. 524-526: IEEE, 2010.

[58] S. Jain *et al.*, "A 90mW/GFlop 3.4 GHz reconfigurable fused/continuous multiply-accumulator for floating-point and integer operands in 65nm," in *23rd International Conference on VLSI Design*, pp. 252-257: IEEE, 2010.

[59] I. Kouretas, C. Basetas, and V. Paliouras, "Low-power logarithmic number system addition/subtraction and their impact on digital filters," *IEEE Transactions on Computers,* vol. 62, no. 11, pp. 2196-2209, 2012.

[60] S. E. Esmaeili, A. J. Al-Kahlili, and G. E. R. Cowan, "Low-swing differential conditional capturing flip-flop for LC resonant clock distribution networks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 20, no. 8, pp. 1547-1551, 2012.

[61] P. Maechler *et al.*, "VLSI design of approximate message passing for signal restoration and compressive sensing," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems,* vol. 2, no. 3, pp. 579-590, 2012.

[62] C. Zhang, C. Wang, and M. O. Ahmad, "A pipeline VLSI architecture for fast computation of the 2-D discrete wavelet transform," *IEEE Transactions on Circuits and Systems-I: Regular Papers,* vol. 59, no. 8, pp. 1775-1785, 2012.

[63] J. Mooney, A. E. Mahdi, and M. Halton, "Application-specific instruction-set processor for control of multi-rail DC-DC converter systems," *IEEE Transactions on Circuits and Systems-I: Regular Papers,* vol. 60, no. 1, pp. 243-254, 2013.

[64] B. Marr, B. Degnan, P. Hasler, and D. Anderson, "Scaling energy per operation via an asynchronous pipeline," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 21, no. 1, pp. 147-151, 2013.

[65] A. Abdelgawad, "Low power multiply accumulate unit (MAC) for future wireless sensor networks," in *IEEE Sensors Applications Symposium Proceedings,* Galveston, TX, USA, pp. 129-132: IEEE, 2013.

[66] A. Amaricai, O. Boncalo, and C. E. Gavriliu, "Low-precision DSP-based floating-point multiply-add fused for field programmable gate arrays," *The Institution of Engineering and Technology*, vol. 8, no. 4, pp. 187-197, 2014.

[67] A. Burg and O. Keren, "Universal hardware for systems with acceptable representations as low order polynomials," *IEEE Transactions on Circuits and Systems-I: Regular Papers,* vol. 61, no. 10, pp. 2878-2887, 2014.

[68] S. Ahish, Y. B. N. Kumar, D. Sharma, and M. H. Vasantha, "Design of high performance multiply-accumulate computation unit," in *IEEE International Advance Computing Conference (IACC)*, Banglore, India, pp. 915-918: IEEE, 2015.

[69] N. Akbarzadeh, S. Timarchi, and A. A. Hamidi, "Efficient multiply-add unit specified for DSPs utilizing low-power pipeline modulo 2n+ 1 multiplier," in *9th Iranian Conference on Machine Vision and Image Processing*, Shahid Beheshti University, Tehran, Iran, pp. 120-123, 2015.

[70] Y. C. Chen, C. T. Tang, H. C. Wu, and H. Chen, "A compact multiply-accumulate architecture for clustering pulse-width coded biomedical signals," in *International Symposium on Bioelectronics and Bioinformatics (ISBB)*, Beijing, China, pp. 83-86, 2015.

[71] U. Cini and O. Kurt, "A high performance multiply-accumulate unit with double carry-save scheme for 6-input LUT based reconfigurable systems," in *9th International Conference on Electrical and Electronics Engineering (ELECO)*, Bursa, Turkey, pp. 940-944, 2015.

[72] L. Gerlach, G. Payá-Vayá, and H. Blume, "An area efficient real and complex-valued multiply-accumulate SIMD unit for digital signal processors," in *IEEE Workshop on Signal Processing Systems (SiPS)*, Hangzhou, China, pp. 1-6, 2015.

[73] R. Kumar and M. Pattanaik, "A novel dual multiplier floating point multiply accumulate architecture," in *19th International Symposium on VLSI Design and Test*, Ahmedabad, India, pp. 1-2, 2015.

[74] V. Priya and V. Kavitha, "Design of efficient multiply-accumulate block for PID controllers," in *2nd International Conference on Electronics and Communication Systems (ICECS)*, Coimbatore, India, pp. 322-325, 2015.

[75] A. Rahul Narasimhan and R. S. Subramanian, "High speed multiply-accumulator coprocessor realized for digital filters," in *IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, Coimbatore, India, pp. 1-4, 2015.

[76] L. S. DeBrunner, D. Williams, and C. Riker, "Truncated multiply-and-accumulate units for FIR filter implementation with reduced coefficient length," in *51st Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, USA, pp. 457-461, 2015.

[77] M. M. A. Basiri and S. N. Mahammad, "Configurable folded IIR filter design," *IEEE Transactions on Circuits and Systems II: Express Briefs,* vol. 62, no. 12, pp. 1144-1148, 2015.

[78] A. Nandal, T. Vigneswarn, A. K. Rana, and A. Dhaka, "An efficient 256-tap parallel FIR digital filter implementation using distributed arithmetic architecture," in *Eleventh International Multi-Conference on Information Processing-2015*, Banglore, India, pp. 605-611, 2015.

[79] R. Anitha, N. Deshmukh, S. K. Sahoo, P. S. Karthikeyan, and I. J. Reglend, "A 32 bit MAC unit design using vedic multiplier and reversible logic gate," in *2015 International Conference on Circuit, Power and Computing Technologies [ICCPCT]*, Nagercoil, India, pp. 1-6: IEEE, 2015.

[80] K. V. Karthikeyan, R. Babu, N. Mathan, and B. Karthick, "Performance analysis of an efficient MAC unit using CNTFET technology," in *Recent Advances In Nano Science And Technology 2015*, Chennai, Tamilnadu, India, vol. 3, pp. 2525-2531, 2016.

[81] A. K. Dhindsa and R. Sarma, "Pipelined and clock gated MAC architecture design and implementation," *Far East Journal of Electronics and Communications,* vol. 16, no. 1, pp. 607-621, 2016.

[82] J. Garland and D. Gregg, "Low complexity multiply accumulate unit for weight-sharing convolutional neural networks," *IEEE Computer Architecture Letters,* vol. 16, no. 2, pp. 132-135, 2017.

[83] D. I. Jeon, K. B. Park, and K. S. Chung, "HMC-MAC: processing-in memory architecture for multiply-accumulate operations with hybrid memory cube," *IEEE Computer Architecture Letters,* vol. 17, no. 1, pp. 5-8, 2017.

[84] A. V. Ananthalakshmi and G. F. Sudha, "A novel power efficient 0.64-GFlops fused 32-bit reversible floating point arithmetic unit architecture for digital signal processing applications," *Microprocessors and Microsystems,* vol. 51, no. 1, pp. 366-385, 2017.

[85] W. Kamp, N. Abel, and G. Comoretto, "Complex multiply accumulate cells for the square kilometre array correlators," in *International Conference on ReConFigurable Computing and FPGAs (ReConFig),* Cancun, Mexico, Mexico, pp. 1-6: IEEE, 2018.

[86] A. Lv, C. Wang, L. Hou, Z. Zeng, J. Guo, and N. Jiang, "An arithmetic unit and multiplying accumulation unit of a custom floating point data format," in *IEEE 3rd International Conference on Integrated Circuits and Microsystems (ICICM),* Shanghai, China, pp. 282-285, 2018.

[87] H. Zhang, H. J. Lee, and S. B. Ko, "Efficient fixed/floating-point merged mixed-precision multiply-accumulate unit for deep learning processors," in *IEEE International Symposium on Circuits and Systems (ISCAS),* Florence, Italy, pp. 1-5: IEEE, 2018.

[88] K. Chen, L. Chen, P. Reviriego, and F. Lombardi, "Efficient implementations of reduced precision redundancy (RPR) multiply and accumulate (MAC)," *IEEE Transactions on Computers,* vol. 68, no. 5, pp. 784-790, 2018.

[89] S. Ryu, N. Park, and J. J. Kim, "Feedforward-cutset-free pipelined multiply–accumulate unit for the machine learning accelerator," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 27, no. 1, pp. 138-146, 2018.

[90] P. A. Patil and C. Kulkarni, "Multiply accumulate unit using radix-4 booth encoding," in *Second International Conference on Intelligent Computing and Control Systems (ICICCS),* Madurai, India, pp. 1076-1080: IEEE, 2018.

[91] P. A. Patil and C. Kulkarni, "A survey on multiply accumulate unit," in *Fourth International Conference on Computing Communication Control and Automation (ICCUBEA),* Pune, India, pp. 1-5, 2019.

[92] V. Camus, C. Enz, and M. Verhelst, "Survey of precision-scalable multiply-accumulate units for neural-network processing," in *IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS),* Hsinchu, Taiwan, pp. 57-61, 2019.

[93] H. Zhang, J. He, and S. B. Ko, "Efficient posit multiply-accumulate unit generator for deep learning applications," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, Sapporo, Japan, pp. 1-5: IEEE, 2019.

[94] V. M. Senthilkumar, S. Ravindrakumar, D. Nithya, and N. V. Kousik, "A vedic mathematics based processor core for discrete wavelet transform using FinFET and CNTFET technology for biomedical signal processing," *Microprocessors and Microsystems,* vol. 71, no. 1, pp. 16-32, 2019.

[95] C. W. Tung and S. H. Huang, "A high-performance multiply-accumulate unit by integrating additions and accumulations into partial product reduction process," *IEEE Access,* vol. 8, pp. 87367-87377, 2020.

[96] M. A. Nahmias, T. F. De Lima, A. N. Tait, H. T. Peng, B. J. Shastri, and P. R. Prucnal, "Photonic multiply-accumulate operations for neural networks," *IEEE Journal of Selected Topics in Quantum Electronics,* vol. 26, no. 1, pp. 1-18, 2019.

[97] H. Zhang, D. Chen, and S. Ko, "New flexible multiple-precision multiply-accumulate unit for deep neural network training and inference," *IEEE Transactions on Computers,* vol. 69, no. 1, pp. 26-38, 2019.

LIST OF PUBLICATIONS

PATENT & COPYRIGHT:

1. "**N-Bit Compressor Based Multiplier Circuits**", by Rajkumar Sarma, Cherry Bhargava, Sandeep Dhariwal, & Shruti Jain. (June 20th, 2018). Patent Application Number: 201811022936. Accessed on: July 26th, 2019. [Online]. Available:http://ipindiaservices.gov.in/PatentSearch/PatentSearch//ViewApplicationStatus.
2. "**A Multiplexer Based MAC Architecture Implementation on FPGA**", by Rajkumar Sarma, Cherry Bhargava & Shruti Jain (June 25th, 2020). Provisional Patent Application Number: 202011026964.
3. "**A Handbook to Use Digital Circuit Simulation Tools**", by Rajkumar Sarma, Cherry Bhargava & Shruti Jain. Copyright Registration Number: L-83855/2019. Registered on: July 8th, 2019.
4. "**Novel Signed Floating-Point MAC (SFMAC) Architecture**", by Rajkumar Sarma, Cherry Bhargava & Shruti Jain. Copyright Registration Number: L-91179/2020. Registered on: May 6th, 2019.

INTERNATIONAL JOURNALS:

1. R. Sarma, C. Bhargava, and S. Jain, "**A MUX based signed-floating-point MAC architecture using UCM algorithm**", *Bulletin of The Polish Academy of Sciences: Technical Sciences*, In-press. [SCI: 1.385 IF]
2. R. Sarma, C. Bhargava, S. Dhariwal, and S. Jain, "**UCM: A novel approach for delay optimization**", *International Journal of Performability Engineering*, vol. 15, no. 4, pp. 1190-1198, 2019. [SCOPUS: 0.16 SJR]

3. R. Sarma, C. Bhargava, and S. Jain, “**Accelerated PVT analysis of UCM architecture using cadence ADE-XL**”, *International Journal of Engineering and Advanced Technology*, vol. 8, no. 5, pp.1913-1919, 2019. [SCOPUS: 0.1 SJR]
4. R. Sarma, C. Bhargava, and S. Jain, “Application of Ameliorated Harris Hawks Optimizer for Designing of Low-Power Signed Floating-point MAC Architecture”, *Neural Computing and Applications (NCAA)*, submitted & under review. [SCI]

BOOK CHAPTER & CONFERENCES:

1. R. Sarma, C. Bhargava, and S. Jain, “**PVT Variability Check on UCM Architectures at Extreme Temperature-Process Changes**”, *AI Techniques for Reliability Prediction for Electronic Components, IGI Global*, ISBN: 9781799814641, pp. 238-251, 2019.
2. R. Sarma, S. Dhariwal, and S. Jain, “**Design and analysis of a novel 8 × 8 bit signed/unsigned synchronous MAC architecture using clock gating scheme for fixed-point arithmetic**”, *2nd International Conference on Intelligent Circuits and Systems, ICICS 2018*, Punjab, India, pp. 423-429, 2018.
3. R. Sarma, S. Dhariwal, and S. Jain, “**A novel normalization architecture for floating-point arithmetic**”, *International Conference on Data Science & Machine Learning, ICDSML-2020,* accepted for publication.