BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 68, No. 4, 2020
DOI: 10.24425/bpasts.2020.134182

A MUX based signed-floating-point MAC architecture using UCM algorithm

R. SARMA[1], C. BHARGAVA[1*], and S. JAIN[2]

[1] VLSI Design, SEEE, Lovely Professional University, GT Road, Phagwara, Punjab, India
[2] Department of Electronics and Communication Engineering, Waknaghat, Solan, HP, India

Abstract. Digital system algorithms such as FFT algorithms, convolution, image processing algorithm, etc. deploy Multiply and Accumulate (MAC) unit as an evaluative component. The efficiency of a MAC typically relies on the speed of operation, power dissipation, and chip area along with the complexity level of the circuit. In this research paper, a power-delay-efficient signed-floating-point MAC (SFMAC) is proposed using Universal Compressor based Multiplier (UCM). Instead of having a complex design architecture, a simple multiplexer-based circuit is used to achieve a signed-floating output. The 8×8 SFMAC can take 8-bit mantissa and 3-bit exponent and therefore, the input to the SFMAC can be in the range of $-(7.96875)_{10}$ to $+(7.96875)_{10}$. The design and implementation of the proposed architecture is executed on the Cadence Spectre tool in GPDK 90 nm and TSMC 130 nm CMOS, which proves as power and delay efficient.

Key words: floating-point MAC, UCM, cadence, TSMC 130 nm, GPDK 90 nm.

1. Introduction

Today's portable devices can do image filtering to facial recognition, an audio signal enhancement to biometric authentication and voice recognition to gesture-based control. All of those functionalities are digital signal processing (DSP) applications. The DSP algorithms perform a large number of mathematical operations repeatedly and quickly on a set of data samples. Most operating systems and microprocessors of general use can execute DSP algorithms successfully. Still, they are not suitable for use in portable devices such as PDAs and mobile phones because of power efficiency constraints. Nevertheless, the rapid growth of portable electronics has presented VLSI design engineers with the significant challenges of low power and high throughput. Multiply and Accumulate (MAC) unit plays a vital role in assessing the output of a DSP module, among the other digital circuits. It is always preferred to use an efficient MAC module when performing convolution, image filtering, or any other DSP operations. The efficiency of a MAC unit is measured in terms of two factors:

1. speed of operation;
2. overall power consumption [1, 2].

The basic MAC architecture contains the main functional blocks as a 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. The basic operation of 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, as shown in (1)

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

where 'i' denote the range of the values. The generalized block diagram of 8×8 bit MAC is shown in Fig. 1.

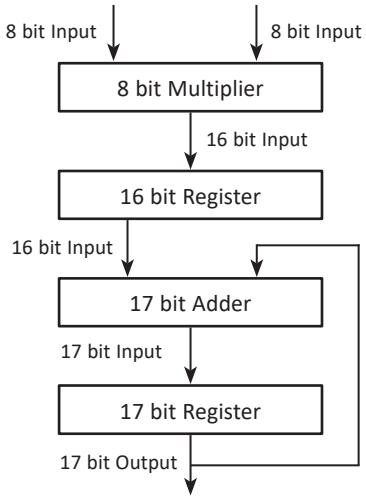

Fig. 1. Generalized block diagram of 8×8 bit MAC

In recent years, several works have been done by different researchers ([3, 4] etc.). A high-speed MAC architecture that promises with the optimized area is proposed in 2007 by Abdelgawad et al. [1]. In 2012, Deepak et al. [5] proposed a novel architecture for the multiplier. In 2013, Jagadees et al. proposed a novel architecture using a modified Wallace tree multiplier

*e-mail: cherry.bhargava@lpu.co.in

Manuscript submitted 2020-01-30, revised 2020-04-18, initially accepted for publication 2020-04-27, published in August 2020

[6]. The implementation is done for 64 bits. In 2013 Francis et al. used a modified Braun Multiplier to implement a basic MAC unit [7]. The implementation is done on NCSim and RTL Compiler. Warrier et al. proposed a low power Baugh-Wooley multiplier-based unit in 2014 [8]. A pipelined based architecture has been proposed in this work. Split MAC architecture is explained by Xia et al. in 2009 [9]. There are several architectures explained in the past by various designers. However, most of the architectures explained in the literature, are designed with the help of hardware descriptive languages such as Verilog or VHDL. Further, a MIFGMOS based circuit design is presented by Topor-Kaminski et al. [10], where add/differential voltage amplifier is proposed using MIFGMOS. In 2019, an FPGA based implementation is done of time to digital converter. The testing is done on the Kintex 7 FPGA board [11]. In 2014, Marciniak et al. proposed a speech signal processing in a system with a TMS320C5515 processor [12]. Whereas a pipelined A to D converter and self-calibrated D to A is proposed in [13] that targets the application of DSP systems. In a different approach, a floating-point accumulator architecture is proposed in 2018 by Jamro et al. [14] that uses a unique floating-point standard other than IEEE-754.

The main disadvantage of using HDL is that the basic blocks that are to be used while designing any architecture using predefined building blocks (standard PMOS-NMOS implementation). The reason is 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. In this research work, a full custom IC approach is adopted for designing the internal blocks of the MAC architecture, yielding an efficient MAC in terms of power as well as delay. Moreover, the proposed SFMAC architecture also uses the Universal Compressor based Multiplier (UCM).

The remaining part of this paper is divided into the following sections: Section 2 describes the UCM architecture and its advantages over Wallace tree multiplier. Section 3 describes the proposed SFMAC architecture in detail with the initial considerations and various essential building blocks. Section 4 describes the results and discussion along with the comparison with the existing architectures. Finally, the conclusion and future work is explained in Section 5.

2. The UCM Architecture

Although the Wallace tree multiplier is much faster than the array multiplier [15], it requires a large number of adders. Moreover, the Wallace tree multiplier is highly irregular and complicated. So, in order to overcome the irregular structure, several modified Wallace tree multipliers are proposed in the literature [4, 15–23]. All these multiplier architectures are based upon the Wallace tree algorithm. Hence, replacing the Wallace tree algorithm may further improve the result of the multiplier. Another important point here is, instead of using traditional Wallace tree adder, compressor circuits such as 3:2 compressors or 4:2 compressors, etc. can be used for partial product addition. But as there is a possibility of using the same compressor again for doing addition (same as Wallace tree addition), the same would not be much effective. Therefore, the Universal Compressor based Multiplier (UCM) architecture is used [24] for the design of the proposed SFMAC architecture, where an N:1 bit size universal compressor algorithm is used for the multiplication operation. It is claimed that the UCM produces the result with better efficiency than any other multiplier proposed in the literature [24].

3. Proposed Signed Floating-point MAC (SFMAC) architecture

The proposed MAC architecture focuses mainly on the signed architecture based on the synchronized block, which allows effective pipeline technique. The block enabling is a power saver technique that momentarily activates a circuit for a small period. Most of the energy/power can be saved because of this simple phenomenon. Additionally, the main reason to implement synchronization 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. On the other hand, pipeline processing is the digital system's ultimate necessity since it dramatically increases the system's performance. As multiplier and the adder are the two core blocks, during the selection of the appropriate circuits, a detailed analysis is made. The delay, as well as the power-efficient design, are given critical importance during the selection of the adder. The adder circuit proposed by Bhattacharyya et al. [3] in 2014 is used in the proposed SFMAC design to tackle the issue of power dissipation. Additionally, as discussed earlier, it is found that although the array multiplier is found to be the most straightforward algorithm for multipliers, it produces a very high delay compared to the Wallace tree multiplier. The rectangular Wallace tree multiplier, on the other hand, is a better option since it divides the partial products into two groups and is, therefore, faster than the conventional Wallace tree multiplier [20]. But the irregular structure is the biggest disadvantage of rectangular styled Wallace tree multiplier. Therefore, the novel UCM architecture, which has a better performance in terms of delay in comparison to the Wallace tree multiplier, is chosen as the multiplier for the proposed design. A multiplexer-based MAC architecture is proposed in this paper, which is capable of performing MAC operation on signed floating-point inputs. For this, a novel input-data format is proposed, 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, with its various basic blocks, uses multiplexer circuits rigorously for selecting between a positive or a negative number.

3.1. Initial Considerations. 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

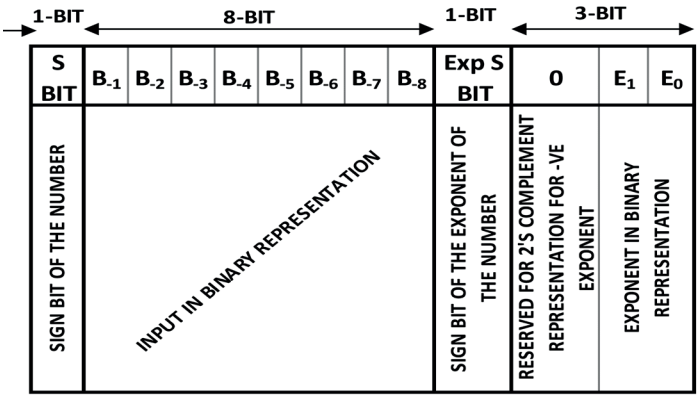

Fig. 2. Input format representation of SFMAC

same inputs are converted to 2's complement form. The final output of the proposed MAC architecture (MAC output) is of 16-bits along with one sign bit and 5-bits of exponent. The inputs to the SFMAC are two 8-bit binary numbers arranged in a format, as shown in Fig. 2. The real value given by the proposed sign-magnitude form is shown in (2).

$$\pm(0.B_{-1}B_{-2}B_{-3}\ldots..B_{-8})_2 \times 2^{\pm(0E_1E_0)_2} \tag{2}$$

The overall size of 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. The sign bit can be 0 or 1 based on positive or negative number representation, respectively. The remaining eleven bits are used for an 8-bit binary representation

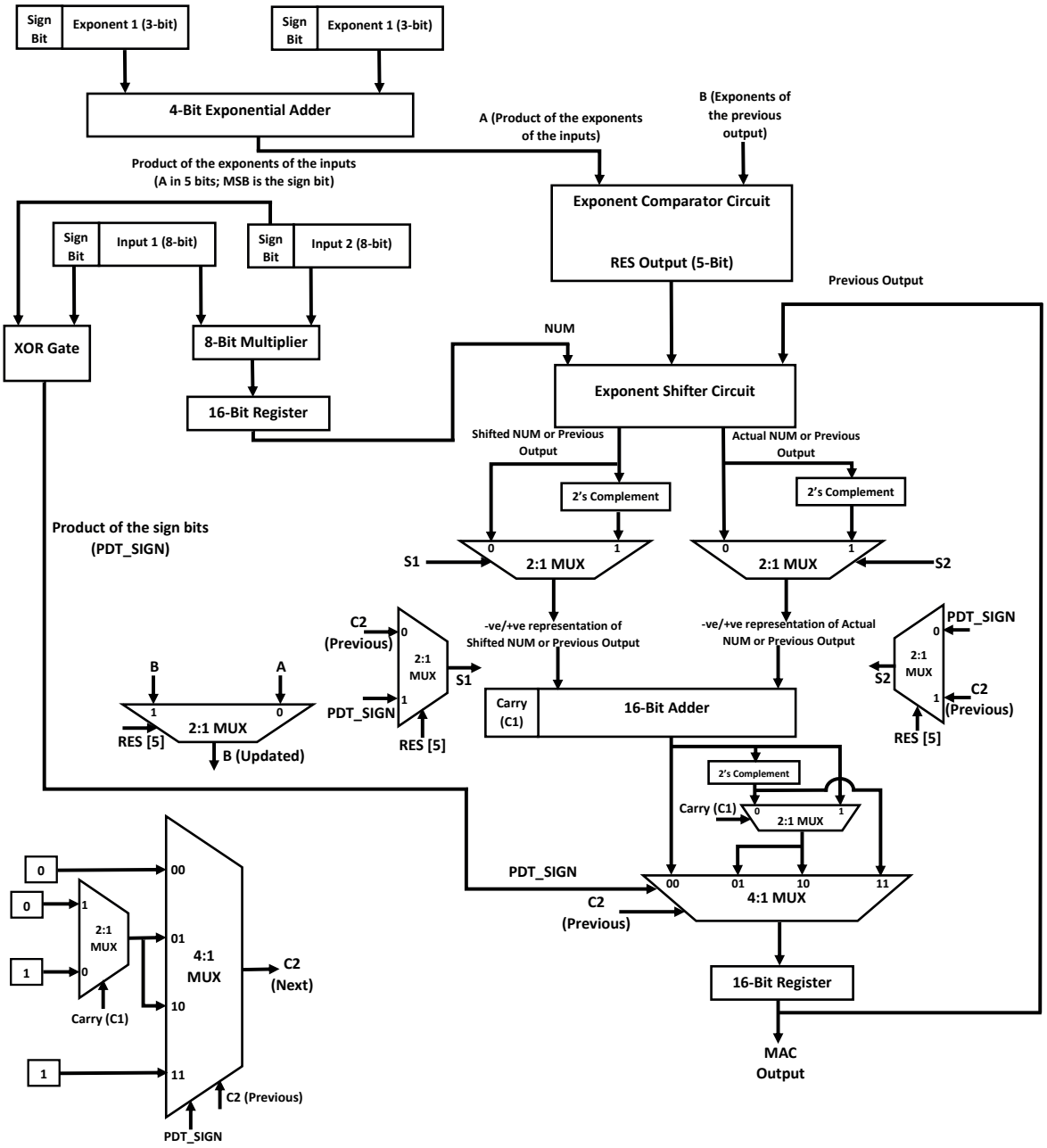

Fig. 3. The novel SFMAC architecture

3-bit exponent representation in binary. One critical 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 shown in (3):

$$-\left(2^{n-1}\right) \rightarrow +\left(2^{n-1}-1\right) \tag{3}$$

where 'n' is the number of bits.

Therefore, in this architecture, the exponent term can range from '–3' to '+3'. The input numbers can 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 decimal number system is from $-(7.96875)_{10}$ to $+(7.96875)_{10}$. Moreover, 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. The major content of the SFMAC architecture are 2's complement block, PED-latch block, 8-bit multiplier, Exponential Adder (EA), 16-bit register, 2:1/4:1 multiplexer of different sizes, Exponent Comparator Circuit (ECC), Exponent Shifter Circuit (ESC) and 16-bit adder. The overall architecture of SFMAC is shown in Fig. 3.

3.2. 2's Complement Block. A 2's complement block is responsible for representing a negative binary number in 2's complement form or vice versa. The architecture of 2's complement block consists of an n-bit inverter unit along with a series of half adders. The 2's complement block is shown in Fig. 4.

3.3. PED-Latch Block. A PED-latch block that uses block enabling technique facilitates the saving of electrical power in digital architecture by reducing the switching activity. 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. Additionally, the PED-latch block uses a "buffered and synchronous pipelined architecture". In the "buffered and synchronous pipelined architectures", "pipeline registers" are introduced between the functional blocks, and are synchronized (using a clock pulse). Each clock signal is delayed in a way that the moment the registers are clocked, the data stored is passed to the next stage. The representation of pipelined architecture with block enabling technique is shown in Fig. 5.

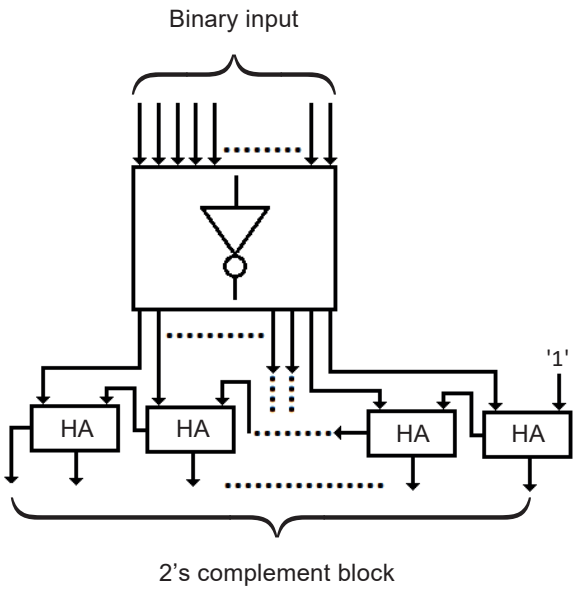

Fig. 4. 2's complement block

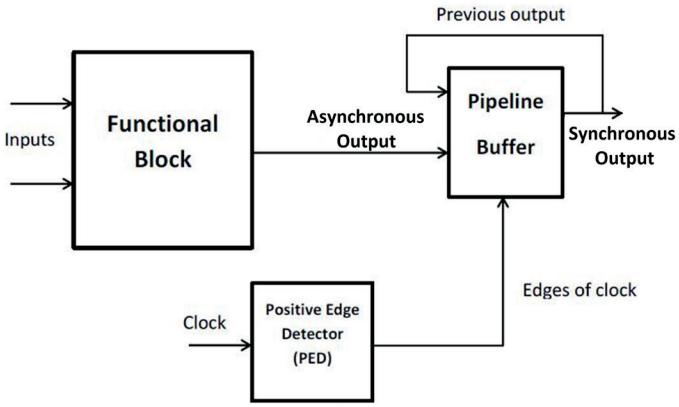

Fig. 5. PED-latch block

3.4. 8-bit Multiplier. The multiplier block used in this case is the UCM architecture, explained in [24]. The additional circuitry that is added to the multiplier is the synchronization.

3.5. Exponential Adder (EA). The EA block performs the addition of the exponents of the inputs $\left(\text{as } 2^n\times 2^m = 2^{(n+m)}\right)$. 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. To make the EA block synchronized, a PED-latch block pair is used at the output of each and every output bits. The MUX-based architecture of the EA block is shown in Fig. 6.

3.6. 16-bit Register. Generally, due to fluctuation in the inputs, the output changes and it is almost impossible to track the output. The main 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.

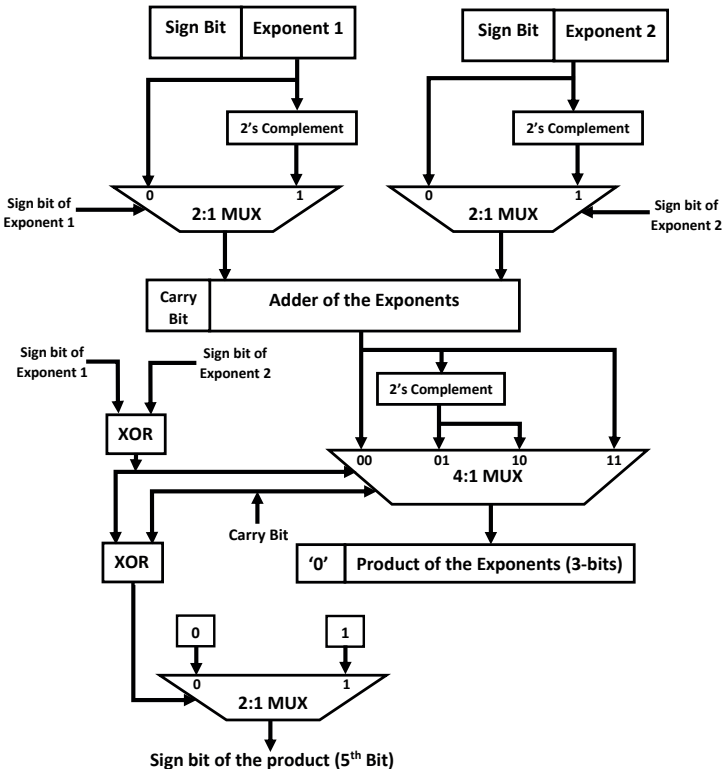

Fig. 6. Operation of EA block

3.7. 2:1/4:1 Multiplexer of Different Sizes. As the algorithm doesn't use any programming approach, for solving the conditions, multiplexers of different sizes with multiple or single bits is considered.

3.8. Exponent Comparator Circuit (ECC). 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 between the two is the arithmetic difference between the numbers. Whereas, if both the inputs carry different signs, then the actual difference between 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)' which is 'b – a' or '(–b) – (–a)' which is 'a – b'. But if the inputs are '+a' and '–b' or '–a' and '+b' then the actual difference is going to be 'a – (–b)' which is 'a + b' or 'b – (–a)' which is 'b + a'. The flowchart in Fig. 7 explains the operation of the ECC block. The architecture uses multiplexers to compare the inputs. The ECC operation produces a 5-bit output, which is to be used for performing the binary shifts.

3.9. 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:

1. 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.
2. The input to the ESC block, which needs no shift, is also identified by the MSB of the ECC block output. If the MSB of the ECC block output is 1, then the previous output is passed as it is (not shifted). On the other hand, if the MSB of the ECC block output is 0, then the product of the inputs is passed as it is (not shifted).

3.10. 16-bit Adder. The adder block is again a synchronized block (i.e., it is clocked and the PED-latch pair is used). 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 its 2's complement value is considered. Similarly, the non-shifted output of the ESC block is negative, then its 2's complement value 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 shifted or non-shifted value of the product of the current inputs and previous output in positive or negative number representation are the inputs to the adder block.

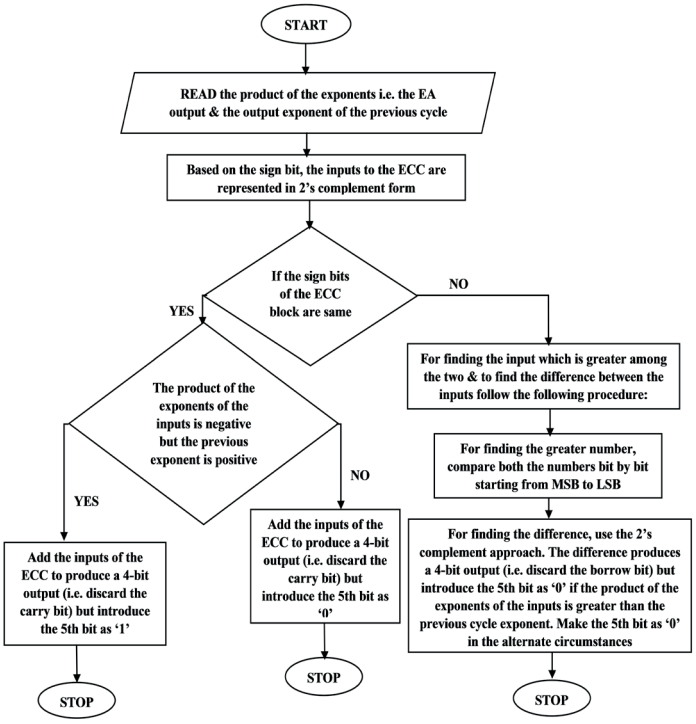

Fig. 7. Operation of ECC block

4. Results and Discussion

The design and implementation of the overall SFMAC architecture is done on Cadence Virtuoso CMOS technologies. The architecture uses a clock gating scheme along with pipelining to reduce power dissipation. The pipeline mechanism using clock pulse is ensured by activating the consecutive blocks after a fixed time period. But the clock for the consecutive blocks are differed by a delay of 1.4 ns. The reason for the delay is to latch the output of the previous block effectively, which acts as the input for the next block. The postponement of clock signals is calculated by the maximum propagation delay of the individual blocks of the SFMAC architecture, as expressed by (4).

$$\tau_{Clock\_delay} = \max\left(\tau_{delay\_EA}, \tau_{delay\_UCM}, \ldots, \tau_{delay\_reg2}\right) \quad (4)$$

where τ_{delay_EA} is the propagation delay of the EA block; τ_{delay_UCM} is the propagation delay of the Universal Compressor based Multiplier (UCM); τ_{delay_reg1} and τ_{delay_reg2} are the propagation delay of the register 1 and register 2 respectively. For finding the delay of the internal blocks, the designs are implemented in 90 nm CMOS technology. The evaluated delay values of the internal blocks of SFMAC is tabulated in Table 1.

As there are 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), which 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 that execution of the last clock and latching on the first clock doesn't get overlapped. Figure 8 shows the output waveform of the proposed SFMAC

Table 1
Propagation delay of the internal blocks of SFMAC architecture

Block	Delay	Inference
UCM	433.7 ps	The maximum delay from A_0 to P_{15}, considering all inputs as high
Register	123.6 ps	Delay from the positive edge of the clock to any of the output
Full Adder	22.4 ps	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 and $Exp2$ as negative
ESC Block (along with ECC block)	1367.5 ps	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 ps	With the critical path from s to y
4:1 MUX	18.9 ps	Maximum delay occurred either in s_0 to y, s_1 to y or s_2 to y

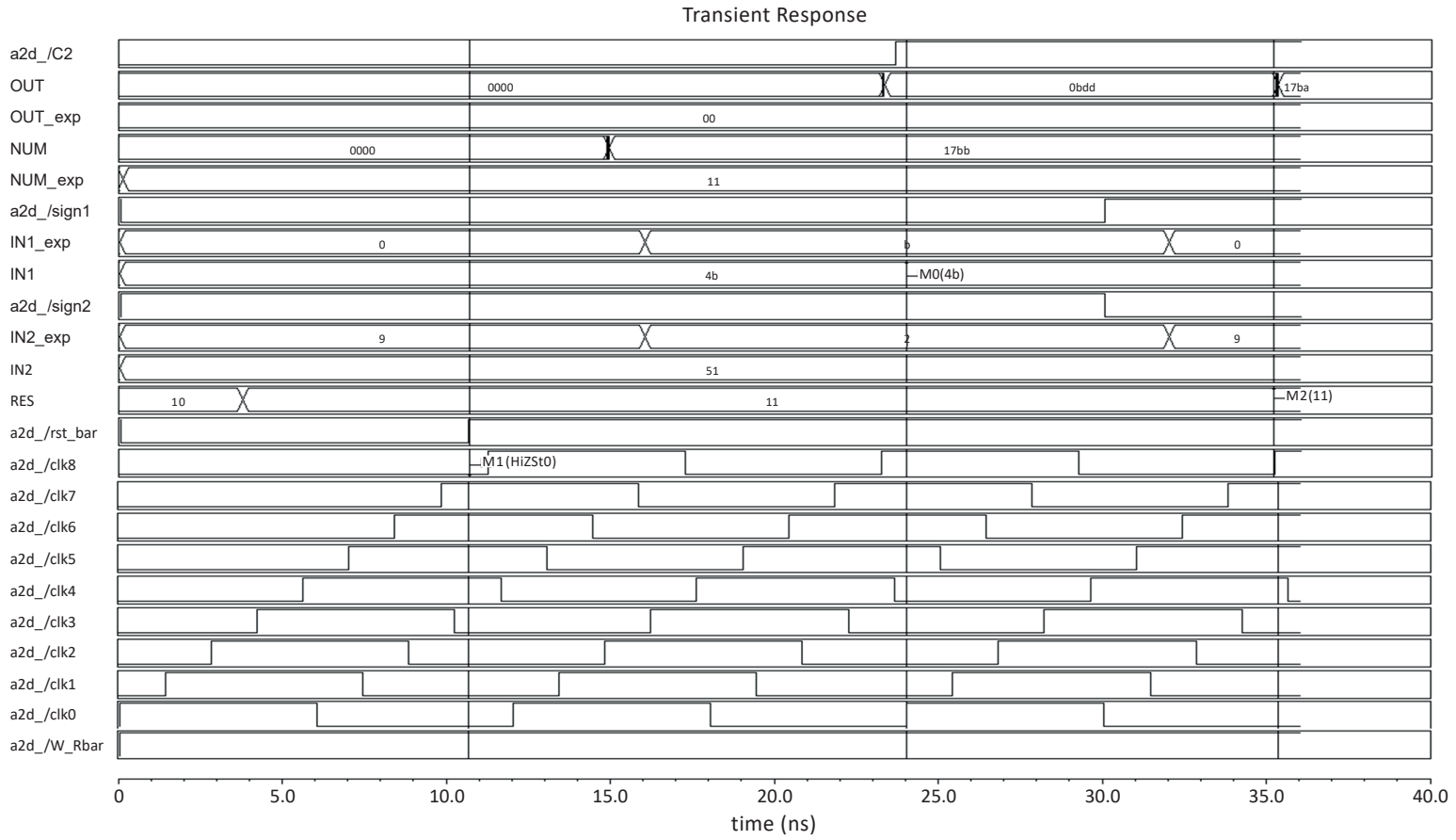

Fig. 8. The simulation waveform of the SFMAC architecture

architecture. The clock in the SFMAC architecture is applied in a pipelined manner, as shown in Fig. 8. The latching of a new set of exponential input is done in clock 0, by activating the EA block which provides the NUM (exponential) output. The *Write*/$\overline{Read}$ signal is used for selecting the write or read operation, which is an active high signal. Whereas the $\overline{Reset}$ signal is used for force resetting the SFMAC system and it is an active low signal. In clock 1, the UCM gets enabled, which provides the 16-bit NUM output based on two 8-bit inputs IN1 and IN2. Clock 2 signal is used as a clock signal for the 16-bit register for the multiplier and there is no clock applied to the ECC block, which produces 5-bit RES output. Clock 3 and RES are applied to the ESC block, which yields the shifted/non-shifted NUM or previous output. Parallelly the same clock is applied to the 2:1 MUXs for updating the select line of the 16-bit 2:1 MUXs to update the true or complemented NUM or previous output.

Clock 4 is applied to the 16-bit 2:1 MUXs to update the true or complemented NUM/previous output. On the other hand, for adding the true or 2's complement form of shifted/non-shifted 16-bit inputs, clock 5 is applied. The 16-bit 2:1 MUX block is activated on the edges of clock 6, which choose between the true value of the full adder output or the 2's complement value of the full adder output. The carry bit of the output of the full adder is applied as the select line. The overall output of the MAC block is based on the selection of XOR of the sign bit of the inputs and the sign bit of the previous output. The input for the 16-bit 4:1 MUX is the output of the 16-bit 2:1 MUX. The inputs for the 16-bit 4:1 MUX is latched at the positive edges of clock 7. The operation of the 16-bit 4:1 MUX is explained in Table 2. Finally, a 16-bit register is used at the output so that the internal glitches don't change the output value. The 16-bit register block is enabled with clock 8, which yields the OUT (SFMAC output), OUT_exp (output exponent) and C2 (sign bit of the output). The SFMAC architecture is not only implemented in GPDK 90 nm but also TSMC 130 nm CMOS technology. Table 3 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 and
2. The clock signals have a time period of 12 ns.

Therefore, until 23.2 ns, the output signal remains at 0. The power consumption of the implemented designs is calculated

Table 2
The operation of the 16-bit 4:1 MUX based on the two select lines

XOR of the sign bit of the inputs	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

Table 3
Comparison of SFMAC at supply voltage 2 V and simulation period 40 ns in GPDK 90 nm and TSMC 130 nm CMOS technology

SFMAC Architecture	Static power in μW (For $V_{DD} = 2$ V)	Average power in μW (For $V_{DD} = 2$ V and simulation period = 40 ns)	Area (total number of transistors)
TSMC 130 nm	2398.76	25990	25783
GPDK 90 nm	476.94	7980	25783

using Cadence Spectre Tool. The static power is evaluated for 2 V 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 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}. 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. The average power of a CMOS circuit is given by (5).

$$P_{avg} = \alpha_T C_{load} V_{DD} f_{clk} \tag{5}$$

where P_{avg} is the average power consumption; α_T is the activity factor; C_{load} is the load capacitance; V_{DD} is the supply voltage and f_{clk} is the clock frequency. On the other hand, the calculation of delay of the overall SFMAC architecture is done by identifying the critical path. While calculating the delay, the maximum possible input is provided so that the worst-case scenario can be obtained. Further, the influence of the clock is also considered while calculating the delay.

4.1. Comparison with existing architectures. The power and delay are the most belied factors in CMOS circuits. Power optimization leads to a significant decrease in the speed of operation on VLSI circuits. Therefore, proper selection of supply voltage should be ensured while designing low-power VLSI circuits. The comparison in terms of power consumption as well as delay of the proposed SFMAC architecture with the existing MAC architectures are shown in Table 4. It is challenging to compare the proposed SFMAC architecture with those who 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 and 130 nm environment. Moreover, almost 99% of the architectures available in the literature have neither implemented for signed operation nor floating-point designs. Though there are some architectures in the literature that 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. Therefore, the practical applicability of such architecture needs to be further tested. Most of the architectures explained in the

Table 4
Proposed SFMAC Vs existing architectures

Sl No.	Author	Existing architecture description	Tool/HDL used	Power Consumption and Delay Analysis		Proposed SFMAC (in 90 nm, 2 V supply for 8×8 bit operation)	
						Power Consumption @ 83.33 MHz	Delay
1	Shanthala, et al., 2009 [26]	Pipelined Multiply Accumulate Unit (fixed-point) in 180 nm technology, 1.8 V at 83.3 MHz and 8×8 bit operation	Cadence Virtuoso	Power: 50.26 mW		Static Power: 0.476 mW Average Power: 7.98 mW	2564.9 ps
				Delay: Not Specified			
2	Jagadees, et al., 2013 [6]	Multiply Accumulate Unit (fixed-point) in 180 nm technology, 1.8V at 217 MHz and 64×64 bit operation	Verilog HDL	Power: 177.732 mW			
				Delay: 4900 ps			
3	Hoang, et al., 2010, [27]	Pipelined Multiply Accumulate Unit (fixed-point) in 65 nm technology, 1.1 V at 591 MHz and 16×16 bit operation	VHDL	Power: 8.2 mW			
				Delay: 1312 ps			
4	Esmaeili, et al., 2012 [28]	Multiply Accumulate Unit (fixed-point) in 90 nm technology, 1 V at 100 MHz and 16×16 bit operation	HDL in Cadence HSPICE simulator	Power: 1.506 mW			
				Delay: Not Specified			
5	Akbarzadeh, et al., 2015 [29]	Pipelined Multiply Accumulate Unit (fixed-point) in 180 nm technology, 1.8 V and 8×8 bit operation	HDL in Synopsys Design Compiler	Dynamic Power	Static Power		
				3.627 mW	2.010 mW		
				Delay:1550 ps			
6	Rahul Narasimhan, et al., 2015 [30]	Multiply Accumulate Unit (fixed-point) in 180 nm technology, 1.8 V at 5 MHz and 16×16 bit operation	Verilog HDL	MAC using Booth	MAC using Vedic		
				Power: 493.648 mW	Power: 1765.241 mW		
				Delay: 11489.4 ps	Delay: 1169.12 ps		
7	Karthikeyan, et al., 2016 [31]	Multiply Accumulate Unit (fixed-point) in 32 nm CMOS and CNTFET technology and 11 bit operation	No details are provided	CMOS Tech	CNTFET Tech		
				0.9902 mW	0.6335 mW		
				Delay: Not Specified			
8	Zhang, et al., 2018 [25]	Fixed/Floating-Point Multiply Accumulate Unit in 90 nm technology for 16-bit half-precision multiplication	VHDL	Power: 14.07 mW			
				Delay: 800 ps			

Table 4, are implemented for unsigned-fixed-point MAC operation and only one architecture (i.e. Zhang, et al., 2018 [25]) is implemented for floating-point signed operation.

The differences are clearly visible from the Table 4 that the performance of [6, 26, 27] have a significantly higher static as well as average power (in mW) than proposed SFMAC architecture. The delay of the unsigned fixed-point architecture in [27] is lesser than the proposed architecture because the operating frequency of the said architecture is much larger than the proposed SFMAC architecture. The performance of [28, 29] are evaluated in 90 nm and 180 nm technologies for 16-bit operations at 1 V and 8-bit operations at 1.8 V respectively. Though, the power consumptions of the existing work mentioned in [28, 29] are lesser than proposed SFMAC architecture (the existing circuit's performance analysis is performed in supply voltage lesser than 2 V whereas for the SFMAC the supply voltage is 2 V). Still, these two existing architectures are capable of performing the MAC operation on a fixed point unsigned number only. Therefore, the existing MAC architectures mentioned in [28, 29] have limited scope. Though the architectures mentioned in [30] are implemented in 180 nm technology and 1.8 V supply voltage for 16-bit MAC operation, the power consumption is way more than the SFMAC architecture. The Vedic multiplier based MAC architecture in [30] has a little advantage in speed in comparison to the proposed SFMAC architecture but at the cost of very high power consumption. For the architecture mentioned in [31], the implementation is done for 1-bit unsigned fixed-point MAC operation in 32 nm CMOS and CNTFET technology and hence, comparison with 8-bit SFMAC is not relevant. Though the architecture mentioned in [25] is the only existing MAC architecture capable of performing on signed floating-point input, the comparison analysis with pro-

posed SFMAC shows that the performance of SFMAC is much better in terms of power consumption. Additionally, the analysis mentioned in [25] doesn't specify any operating frequency of the architecture. Hence, the delay comparison of [25] with the proposed SFMAC architecture is not viable.

3. Conclusions

A suitable power-efficient high-speed MAC unit for signed-floating-point operation is designed using high-speed UCM architecture. The 8×8-bit signed-floating-point MAC (SFMAC) architecture using basic 2:1/4:1 multiplexer is a novel design, which consumes less power at a supply voltage of 2 V and 83.33 MHz operating frequency. The step-wise architectural design is also elaborated in this paper. For design and implementation, the Cadence Spectre tool is used at GPDK 90 nm as well as TSMC 130 nm CMOS technologies. Based on evaluation parameters such as size, supply voltage and operating frequency, proposed SFMAC architecture consumes less power and tolerable amount of worst-case delay. Therefore, it has applicability in low-power high-speed DSP architectures.

References

[1] A. Abdelgawad and M. Bayoumi, "High speed and area efficient multiply accumulate (MAC) unit for digital signal processing applications", *IEEE Int. Symp. Circuits and Systems, New Orleans, LA, USA*, 3199–3202 (2007).

[2] N.J. Babu and R. Sarma, "A novel low power multiplyaccumulate (MAC) unit design for fixed point signed numbers", *Advances in Intelligent Systems and Computing* 394(1): 675–690 (2016).

[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 Trans. Very Large Scale Integr. (VLSI) Syst.* 23(10), 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". *IEEE Int. Symp. Circuits and Systems, Phoenix-Scottsdale, USA*, 81–84 (2002).

[5] S. Deepak and B.J. Kailath, "Optimized MAC unit design",*Int. Conf. on Electron Devices and Solid State Circuit (EDSSC), Bangkok, Thailand*, 1–4 (2012).

[6] P. Jagadees, S. Ravi, and K.H. Mallikarjun, "Design of a high performance 64 bit MAC unit",*Int. Conf. on Circuits, Power and Computing Technologies, Nagercoil, India*, 782–786 (2013).

[7] 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", *4th Int. Conf. on Computing, Communications and Networking Technologies (ICCCNT)*, *Tiruchengode*, *India*, 1–4 (2013).

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

[9] B.J. Xia, P. Liu, and Q.D. Yao, "New method for high performance multiply-accumulator design", *Journal of Zhejiang University Science* 10(7), 1067–1074 (2009).

[10] L. Topor-Kaminski and P. Holajn, "Multiple-input floating-gate MOS transistor in analogue electronics circuit", *Bull. Pol. Ac.: Tech.* 52(3), 251–256 (2004).

[11] P. Kwiatkowski, "Employing FPGA DSP blocks for time-todigital conversion", *Metrol. Meas. Syst.* 26(4), 631–643 (2019).

[12] T. Marciniak, R. Weychan, A. Stankiewicz, and A. Dabrowski, "Biometric speech signal processing in a system with digital signal processor", *Bull. Pol. Ac.: Tech.* 62(3), 589–594 (2014).

[13] K. Wawryn and R. Suszynski, "Low power 9-bit pipelined A/D and 8-bit self-calibrated D/A converters for a DSP system", *Bull. Pol. Ac.: Tech.* 61(4), 979–988 (2013).

[14] E. Jamro, A. Dabrowska-Boruch, P. Russek, M. Wielgosz, and K. Wiatr, "Novel architecture for floating point accumulator with cancelation error detection", *Bull. Pol. Ac.: Tech.* 66(5), 579–587 (2018).

[15] K.B. Jaiswal, N. Kumar, P. Seshadri, and G. Lakshminarayanan, "Low power wallace tree multiplier using modified full adder", *3rd Int. Conf. on Signal Processing*, *Communication and Networking (ICSCN)*, *Chennai*, *India*, 1–4 (2015).

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

[17] 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", *Int. Conf. on Advanced Technologies for Communications (ATC'14)*, *Hanoi*, *Vietnam*, 739–744 (2014).

[18] N. Itoh, Y. Naemura, H. Makino, Y. Nakase, T. Yoshihara, and Y. Horiba, "A 600-MHz 54-bit multiplier with rectangularstyled wallace tree", *IEEE J. Solid-State Circuits* 36(2), 249–257 (2001).

[19] D. Paradhasaradhi, M. Prashanthi, and N. Vivek, "Modified wallace tree multiplier using efficient square root carry select adder", *Int. Conf. on Green Computing Communication and Electrical Engineering (ICGCCEE)*, *Coimbatore*, *India*, 1–5 (2014).

[20] T.Y. Kuo and J.S. Wang, "A low-voltage latch-adder based tree multiplier", *IEEE Int. Symp. Circuits and Systems*, *Seattle*, *WA*, 804–807 (2008).

[21] S. Khan, S. Kakde, and Y. Suryawanshi, "VLSI implementation of reduced complexity wallace multiplier using energy efficient CMOS full adder", *Int. Conf. on Computational Intelligence and Computing Research*, *Enathi*, *India*, 1–4 (2013).

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

[23] B.N.M. Reddy, H.N. Sheshagiri, and S. Shanthala, "Implementation of low power 8-Bit multiplier using gate diffusion input logic", *17th Int. Conf. on Computational Science and Engineering*, *Chengdu*, *China*, 1868–1871 (2014).

[24] R. Sarma, C. Bhargava, and S. Jain, "UCM: A Novel Approach for Delay Optimization", *Int J Performability Eng* 15(4), 1190–1198 (2019).

[25] H. Zhang, H.J. Lee, and S.B. Ko, "Efficient fixed/floatingpoint merged mixed-precision multiply-accumulate unit for deep learning processors", *IEEE Int. Symp. Circuits and Systems*, *Florence*, *Italy*, 1–5 (2018).

[26] S. Shanthala, C.P. Raj, and S.Y. Kulkarni, "Design and VLSI implementation of pipelined multiply accumulate unit", *2nd Int. Conf. on Emerging Trends in Engineering and Technology (ICETET-09)*, *Nagpur*, *India*, 381–386 (2009).

[27] T.T. Hoang, M. Sjlander, and P. Larsson-Edefors, "A highspeed, energy-efficient two-cycle multiply-accumulate (MAC) architecture and its application to a double-throughput MAC unit", *IEEE Trans. Circuits Syst. I*, *Reg. Papers* 57(12), 3073– 3081 (2010).

[28] 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 Trans. Very Large Scale Integr. (VLSI) Syst.*, 20(8), 1547–1551 (2012).

[29] N. Akbarzadeh, S. Timarchi, and A.A. Hamidi, "Efficient multiply-add unit specified for DSPs utilizing low-power pipeline modulo $2n + 1$ multiplier", *9th Iranian Conf. on Machine Vision and Image Processing*, *Tehran*, *Iran*, 120–123 (2015).

[30] A.R. Narasimhan and R.S. Subramanian, "High speed multiply-accumulator coprocessor realized for digital filters", *Int. Conf. on Electrical*, *Computer and Communication Technologies (ICECCT)*, *Coimbatore*, *India*, 1–4 (2015).

[31] K.V. Karthikeyan, R. Babu, N. Mathan, and B. Karthick, "Performance analysis of an efficient MAC unit using CNTFET technology", *Recent Advances In Nano Science And Technology*, *Chennai*, *Tamilnadu*, *India*, 2525–2531 (2016).