

Design and evaluation of clock-gating-based approximate multiplier for error-tolerant applications

Chowdam Venkata Sudhakar¹, Suresh Babu Potlady², Prasad Reddy Karipireddy³

¹Department of Electronics and Communication Engineering, School of Engineering, Mohan Babu University (Erstwhile Sree Vidyanyikethan Engineering College), Tirupati, India

²Department of Electronics and Communication Engineering, Sri Venkateswara College of Engineering, Tirupati, India

³Functional Safety Expert, Espoo, Finland

Article Info

Article history:

Received May 23, 2024

Revised Apr 18, 2025

Accepted Jun 10, 2025

Keywords:

Approximate adders
Approximate computing
Clock-gating multiplier
Field-programmable gate array implementation
Power efficiency
Verilog hardware description language

ABSTRACT

The multiplier is an essential component in real-time applications. Even though approximation arithmetic affects output accuracy in multipliers, it offers a realistic avenue to constructing power area and speed-efficient digital circuits. The approximation computing technique is commonly used in error-tolerant applications such as signal, image, and video processing. In this paper, approximate multipliers (AMs) are designed using both conventional and approximate half adders (A-HA) and full adders (A-FA), which are strategically placed to add partial products at the most significant bit (MSB) positions, and OR gates are used to add partial products at the lower significant bit (LSB). In addition, this research article demonstrates unsigned and signed multipliers using the ripple carry adder (RCA), carry save adder (CSA), conditional sum adder (COSA), carry select adder (CSLA), and clock gating technique. The proposed multipliers are implemented in Verilog hardware description language (HDL) and simulated on the Xilinx VIVADO 2021.2 design tool with target platform Artix-7 AC701 FPGA. The simulation results found that unsigned and signed approximate multiplier power consumption was reduced by 13% and 18.18% respectively and enhanced accuracy.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Venkata Sudhakar Chowdam

Department of Electronics and Communication Engineering, School of Engineering

Mohan Babu University (Erstwhile Sree Vidyanyikethan Engineering College)

Tirupati-517102, India

Email: adrianat@itmorelia.edu.mx

1. INTRODUCTION

An effective multiplier ought to have a small area and high concert [1]. A well-known strategy for improving circuit concert without increasing hardware complication is to use approximations rather than exact computations for the hardware implementation of multipliers [2]. When developing a multiplier using the approximate method, abate design characteristics such as delay and power needs while sacrificing precision. This technique may diminish results accuracy, but it is appropriate for error-resilience applications such as signal, image, and video processing [3]. The approximate multiplication is potentially significant in hardware complexity reduction, power consumption, and processing time reduction [4].

There are various techniques and strategies employed in designing approximate multipliers for reducing hardware complexity, power consumption, and delay while accepting a certain level of error in the result [5], [6]. Some common techniques and strategies employed in designing approximate multipliers include approximate arithmetic techniques, truncated multiplication, parallelism and pipelining, approximate

carry-save and sum (CSA) trees, approximate booth encoding, approximate multiplication algorithms, dynamic voltage and frequency scaling, error estimation and compensation, application-specific customization, hardware-software co-design [7]. These methods are used with a variety of multiplier architectures, such as array multipliers, booth multipliers, and Wallace tree multiplier (WTA) [8]. The technique chosen for the multiplier design will be determined by the application's specific requirements, such as the level of precision required, power consumption constraints, and available hardware resources.

The objective of the research is to design and evaluation of approximation multipliers for achieving low power consumption, low area, and delay by using clock-gating, approximate arithmetic circuits. In this paper, AMs are designed using both conventional and A-HAs and A-FAs, which are strategically placed to add partial products at the most significant bit (MSB) positions, and OR gates are used to add partial products at the lower significant bit (LSB).

Salehi [4] designed and synthesized 4-to-128-bit multipliers using system Verilog and Synopsys design compiler. According to post-synthesis studies, a 128-bit multiplier hoarded energy, 65% less critical delay, and around 45% chip area than an accurate equivalent. In evaluating the efficiency of the method, a real-world image analysis application revealed up to 68.3% energy reduction with minimal losses. A 16-bit approximation multiplier constructed in a 28 nm CMOS technology exhibits a 20% and up to 69% reduction in delay and power, respectively, when compared to the WTA [9]. Venkatachalam and Ko [10] suggested two approximate multipliers with power saving by 72% and 38%, respectively. When compared to current approximation multipliers, they exhibit greater precision. The suggested approximation multipliers have mean relative error estimates as low as 7.6% and 0.02%, respectively, which is better than the prior works. In recent years, approximate computing methods significantly increase energy efficiency by removing the requirement for totally exact or completely deterministic computations [11]. Immareddy and Sundaramoorthy [12] proposed a tuneable error characteristics multiplier with an average error of 1.39%-3.32%, these inaccurate multipliers save an average of 31.78%-45.4% in power when compared to similar accurate multiplier designs. Rajo and Rao [13] discussed the history and advancements of approximation multiplier architectural design, as well as a prospective analysis for future advancements. Ramasamy and Nagarajan [14] developed an 8-bit hybrid segment approximate multiplier (HS-AM) and an extended HS-AM for image compression with accuracy 99.85% and 99.999%, respectively, for varied inputs. Osta *et al.* [15] constructed an approximate multiplier based on inexact adder circuits and achieved a power savings of up to 17.39% while improving time delay by 13.49% at a cost of less than 5% accuracy loss [16].

This research assists VLSI engineers in developing power, area, and delay efficient circuits at the cost of accuracy in error tolerance applications such as audio and video processing employing machine learning methods, computer graphics, wireless communication, robotics, and internet of thing (IoT) devices. The rest of the article is organized as follows: sections 2 and 3 elaborated on materials and methods used for designing approximation multipliers (respectively). Section 4 illustrates results and discussions followed by conclusions and references in section 5.

2. RESEARCH MATERIALS

A typical approximate multiplier is implemented in three stages. The 1st stage includes partial product generation using AND gates. In the 2nd stage, partial products (PPs) are added using approximate arithmetic circuits to diminish the number of additions along with speeding up the multiplication, in the 3rd stage final product is obtained using four different adders namely ripple carry adder (RCA), conditional sum adder (COSA), carry save adder (CSA), and carry select adder (CSLA), and computed multiplier's parameters power, area, and delay. The results are related to find which is efficient among them.

2.1. Approximate adders

Approximation adders are designed and implemented to minimize the multiplier complexity and power ingesting in error-tolerant applications. Approximate adder circuits are used in multipliers to perform partial product addition operations with some level of approximation, often sacrificing accuracy for gains in performance, power efficiency, latency, and area reduction [16].

2.1.1. Half adders

Approximate half adder (A-HA) is a component used in digital circuits to execute an addition of two single binary bits with some level of approximation [17]. Figure 1(a) shows a conventional half adder (C-HA) in which output sum (S) is implemented using an exclusively-OR (XOR) gate and its Boolean expression represented in (1) and carry (C) output implemented with logic AND gate and its Boolean expression is given in (2):

$$S = A \oplus B \quad (1)$$

$$C = A \cdot B \quad (2)$$

where \oplus signifies the XOR operation and period ‘.’ signifies the AND operation.

Figure 1(b) shows an A-HA in which output SUM(S) is implemented using a logic OR gate and CARRY (C) output implemented with logic AND gate. The output logic expressions of A-HA are given in (3) and (4). In the A-HA, the XOR gate is replaced by an OR gate to obtain the sum out [18]:

$$\text{SUM} = A + B \quad (3)$$

$$\text{CARRY} = A \cdot B \quad (4)$$

where ‘+’ represents the OR operation.

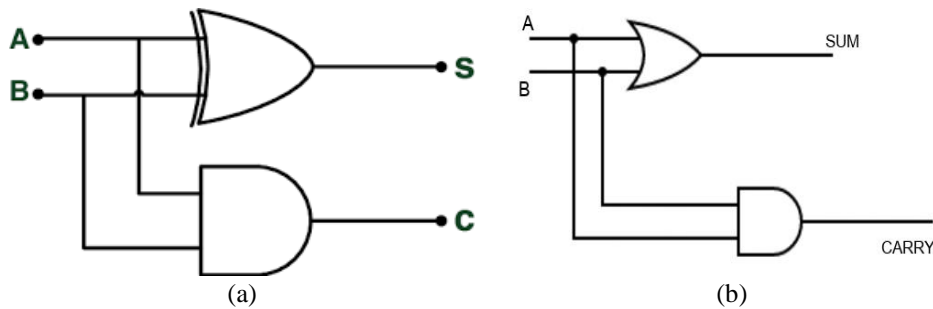


Figure 1. Half adder logic circuits: (a) C-HA and (b) A-HA

Table 1 presents the truth table for a C-HA and A-HA, with one absolute difference highlighted in red. In C-HA the S output is calculated with the XOR gate, which returns high (or 1) only when the number of high inputs is odd. The AND gate is used to calculate the C output, which only returns high (or 1) if both inputs are high.

Table 1. Half adder truth table with an absolute difference

Inputs				Outputs		Absolute difference
A	B	C-HA		A-HA		
		S	C	SUM	CARRY	
0	0	0	0	0	0	0
0	1	1	0	1	0	0
1	0	1	0	1	0	0
1	1	0	1	1	1	1

2.1.2. Full adders

Figure 2(a) shows a conventional full adder (C-FA), It adds three binary bits, usually denoted as A, B, and C, and produces a Sum(S) bit and a Carry bit as output. In C-FA output Sum(S) is implemented using XOR gate and output Carry (C) implemented with a AND logic followed by OR logic gates. The logic expressions of Sum(S) and Carry (C) are given in (5) and (6) respectively.

$$\text{Sum} = A \oplus B \oplus C \quad (5)$$

$$C_{\text{out}} = A \cdot B + (C \cdot (A \oplus B)) \quad (6)$$

Figure 2(b) shows an approximate full adder (A-FA) in which ‘SUM’ is a complement of ‘CARRY’ and ‘CARRY’ is implemented AND gates. The logic expressions of adders are given in (7) and (8). In the AFA, the XOR gate of the sum is replaced by and-or-inverter (AOI) gates [19].

$$\text{CARRY} = A \cdot B + B \cdot C + C \cdot A \quad (7)$$

$$\text{SUM} = \text{CARRY}' \quad (8)$$

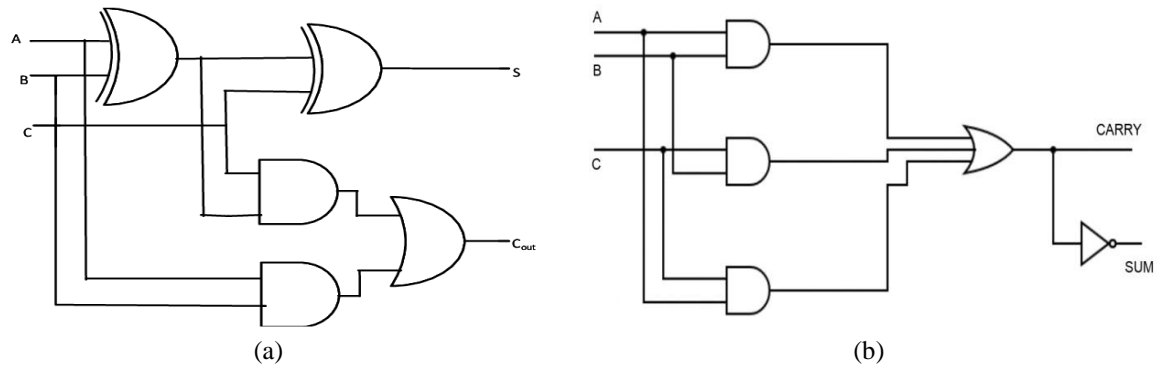


Figure 2. Full adder logic circuits: (a) C-FA and (b) A-FA

Table 2 illustrates the C-FA and A-FA truth table with absolute differences indicated with red color. In C-FA the output S is calculated with the two XOR gates connected in series, which returns high (or 1) only when the number of high (or 1) inputs is odd. The AND logic followed by OR gate calculate the C.

Table 2. Full adder truth table with an absolute difference

Inputs			Outputs				Absolute difference
A	B	C	Conventional FA S	Conventional FA C _{out}	Approximate FA SUM	Approximate FA CARRY	
0	0	0	0	0	0	0	0
0	0	1	1	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	1	0	1	0
1	0	0	1	0	1	0	0
1	0	1	0	1	0	1	0
1	1	0	0	1	0	1	0
1	1	1	1	1	0	1	1

2.2. 4-bit ripple carry adder

Figure 3 depicts a 4-bit RCA, which consists of four complete adders. The first bits of operands, A₀ and B₀, are transferred to the first full-adder (FA₀), with Cin equal to zero (Cin=0). The initial bit of sum (S₀) is formed, and the output carry (Co) is propagated to the second adder that comes before it. Similarly, the second full adder receives the operand's second bit, the third full adder receives the third bit, and the fourth adder receives the fourth bit. Each FA generates a corresponding sum, and a carry, both of which are rippled to the following FA as input carry. The RCA circuit offers a relatively short design time, but it becomes rather slow as the number of stages increases [20].

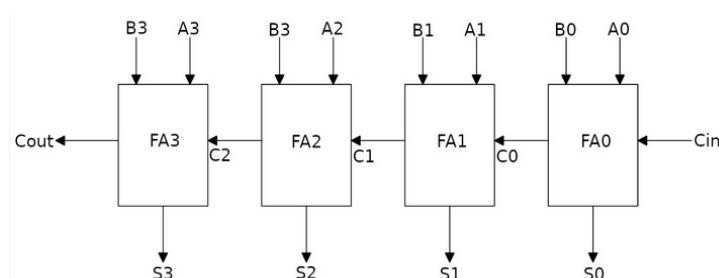


Figure 3. 4-bit RCA

2.3. 4-bit carry select adder

An RCA has a smaller area in design while it has low speed. A carry look ahead adder (CLA) is faster in operation but it occupies a high area. CSA lies in between the spectrum. Generally, CSLA consists of RCA and multiplexers. CSLA is a multi-operand addition circuit that picks the 'Sum' and 'Carry' output from stage-1 RCA when the carry input is '0' and the 'Sum' and 'Carry' output from stage-2 RCA when the

carry input is '1'. An $N+1$ multiplexer controls the 'Sum' and 'Carry' outputs of N -bit addition operations. Figure 4 shows a 4-bit CSLA using two 4-bit RCA and 5 numbers of 2 to 1 multiplexers [20].

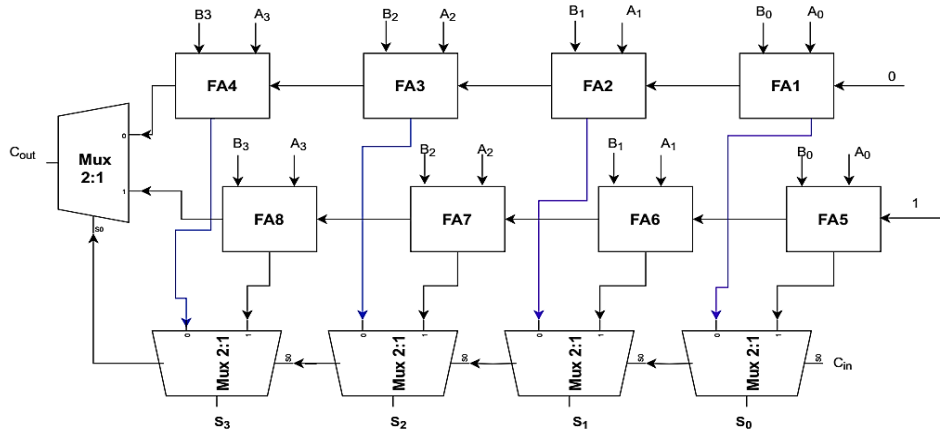


Figure 4. 4-bit CSLA

2.4. 4-bit carry save adder

A 4-bit CSA helps in summation of three 4-bit operands in a way that reduces the propagation delay compared to traditional ripple-carry adders [20]. CSAs are used in the calculation of the accumulation of partial products in integer multiplication [21]. A 4-bit CSA adds three 4-bit binary values, A [3:0], B [3:0], and C [3:0], to produce a 4-bit partial sum1, M [3:0], and a 4-bit carry, N [4:1], without immediately propagating the carry result. The carry bits from each bit location are preserved and applied to the next significant position in next step. Figure 5(a) illustrates how it works.

Figure 5(b) depicts the two steps of implementation for the 4-bit CSA. In the first step, four full adders are placed in parallel. Each full adder processes one bit of operands A [3:0], B [3:0], and C [3:0], resulting in partial sum1 M [3:0] and carry N [4:1]. Stage 2 is identical to RCA in that the stored carry and sum1 bits are combined to produce the final sum S [4:0] and carry-out.

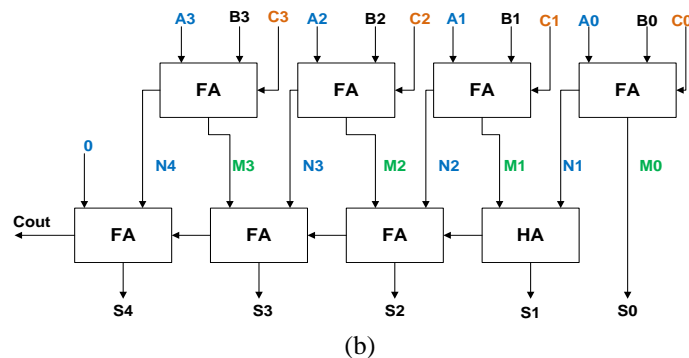
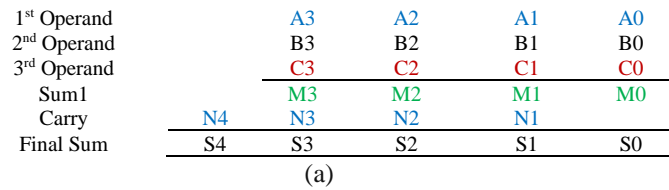


Figure 5. 4-bit (a) carry save adder logic and (b) carry save adder

2.5. 4-bit conditional sum adder

In this 4-bit COSA, the first FA takes inputs X_0 , Y_0 , and C_{in} and generates C_{out} . For inputs X_1 and Y_1 , there are two adders with carry-ins of 0 and 1, respectively. The sum S_1 is the outcome of a multiplexing

(MUX) operation between the two sums generated using the select lines as carry-out from the previous FA. The second addition is also multiplexed and provided as a select signal to select greater bit additions, as illustrated in Figure 6. The interconnector essentially aggregates the input bit lines into a larger sum bus. The addition is substantially quicker since higher bit addition operations do not rely on carry propagation (CP) [20]. However, selecting the proper result requires the preceding carry-out to be established. This is a somewhat affluent design because $N=2^n$ -bit summation requires $2N-1$ FAs and $(2^n - n - 2)$ MUXs, which are not encountered in RCAs [22].

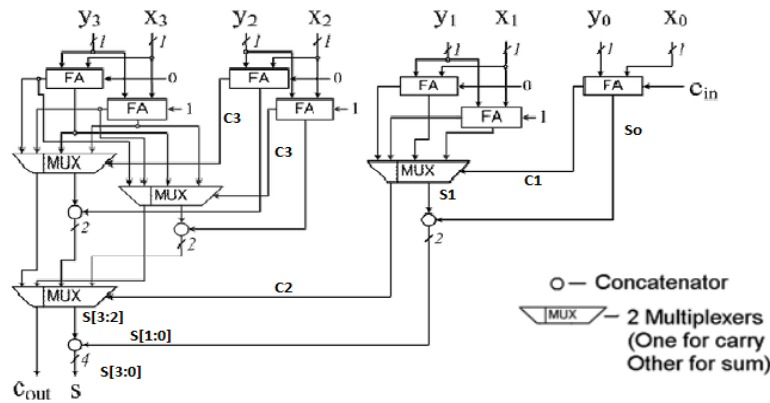


Figure 6. 4-bit COSA

3. RESEARCH METHOD

Multipliers come in two varieties: unsigned and signed. Unsigned multiplication uses all N -bits to express the magnitude of the operand. In signed multiplication, the operand's MSB bit (N th bit) represents the sign bit, while the remaining $N-1$ bits reflect the operand's magnitude. The recital of the multiplier in terms of speed, area, and power utilization is further influenced by the number of bits utilized [23]. Fixed-point signed numbers are represented widely using Sign-magnitude representation or complement representation [20], [24].

3.1. 4-bit approximate multiplier

Unsigned and signed 4-bit approximate multipliers are crafted with AND gates for producing partial products, OR gates utilized in the part of the products accumulation stage to minimize area and energy consumption. Inexact half adders (inHA) and inexact full adders (inFA) [6] are used in the multiplier's upper half part (MSB positions), OR gates [25] are applied in the lower part (LSB positions), where partial products are segregated as lower and upper parts using the critical column as shown in Figure 7. A traditional RCA is utilized as the basic adder block for the final Sum [26].

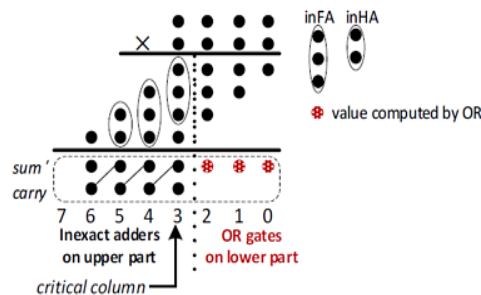


Figure 7. Dot diagram of approximate 4×4 multiplier

An approximate 4-bit multiplier shown in Figure 8 is implemented using an approximate arithmetic logic of Gates and circuits to accomplish the multiplication operation between two 4-bit numbers that is $A[3:0]$ and $B[3:0]$. At first, the partial products are produced by AND Gates between all the input bits. In the second step, 16 partial products $P[15:0]$ are reduced using full adders, half adders, and OR gates to get

the final product. Figure 8 shows the architecture of the approximate 4-bit multiplier. Two FAs, one half-adder, and three OR gates are used to minimize the partial products. After two stages of reduction, we get the three LSBs of the output, $Y[2:0]$, and two 4-bit values that are accumulated using RCA/COSA/C/CSLA/CSA to produce the MSB part of the output, $Y[7:3]$ [27]. An approximation multiplier with inexact HA, FA, OR compressors is a simple and efficient approach to approximate the product of two binary values. This estimated design provides lower circuit complexity but has the poorest error performance.

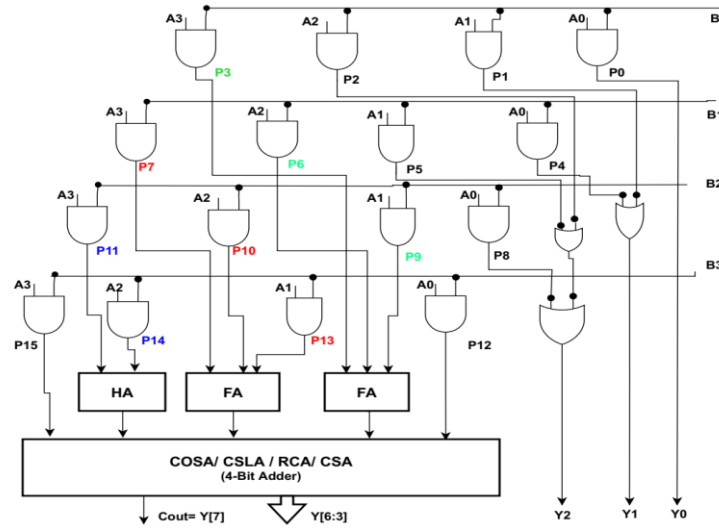


Figure 8. Architecture of 4-bit approximate multiplier

3.2. 8-bit approximate multiplier

Figure 9 represents an approximate 8-bit multiplier architecture constructed by using four approximate 4×4 multiplier blocks are shown in Figure 8 [10]. The LSB part of product $Z[6:0]$ is achieved through OR gates, while CSA provides the MSB portion of product $Z[15:7]$. The final product ($Z[15:0]$) is obtained by combining the LSB part ($Z[6:0]$) and the MSB part ($Z[15:7]$). The architecture is designed to provide an approximate result while reducing computational complexity and power consumption [28].

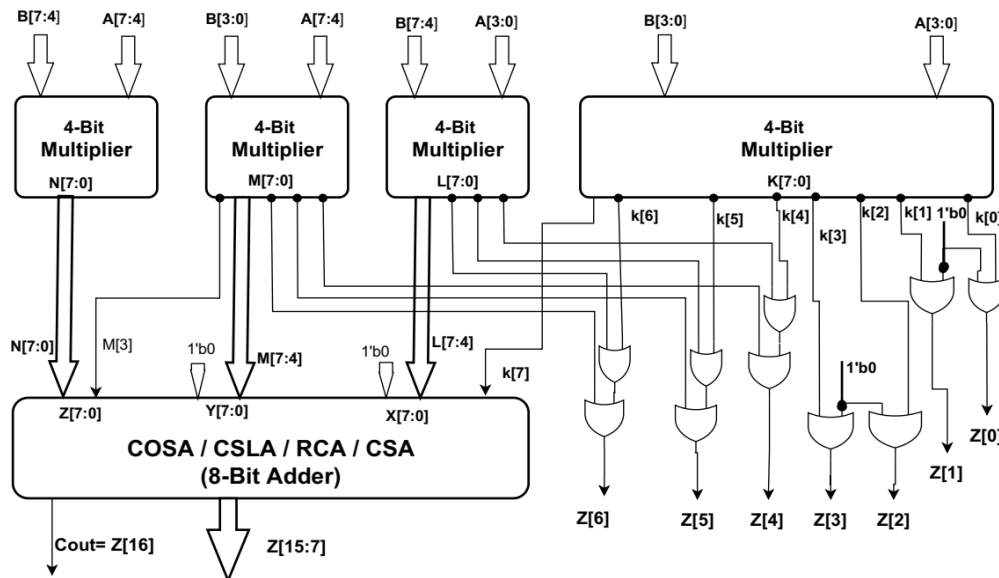


Figure 9. Architecture of 8-bit approximate multiplier using 4-bit multipliers

3.3. 16-bit approximate multipliers with clock-gating

The clock-gating approach used in multipliers reduces both power and circuit area. Applying a clock-gating logic prevents unnecessary switching of adders during clock cycles while stored data is intact [29]. The suggested architecture reduces dynamic power dissipation by suppressing signal activities with clock-gating when they are not required. Figure 10 shows a block diagram of a multiplier with a flip-flop-based clock gating multiplier [30].

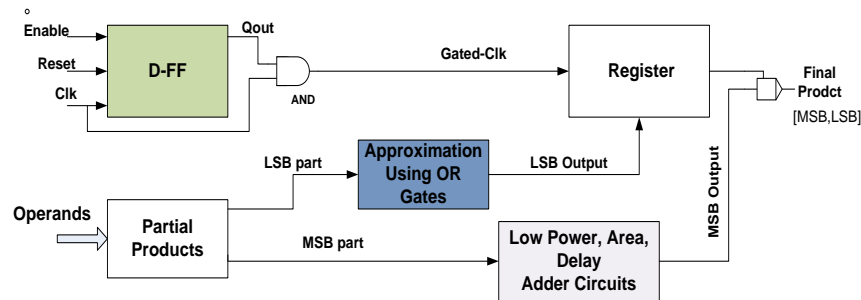


Figure 10. Generalized flip-flop based clock gating approximate multiplier

In Figure 10, the gated-clk goes too high during D-FF output and the clock input is in a high state otherwise gated clock is at zero state. In the existing approximate multiplier, the approximation method is applied on either side of the critical column, which is named the LSB and MSB. In the suggested new design, the approximation over the LSB side remains the same, and the approximation at the MSB side, the approximate HAs and FAs have been swapped out with exact half-full adders at the appropriate places [16] to enhance the precision of the estimated multiplier. The outcome of the LSB part is enabled by the gated clock thus, the power utilization of the multiplier is minimized. Figure 11 depicts the approximate 4-bit multiplier, in which the MSB partial products are aggregated using exact full adder (exact FA) and half adders (exact HA), resulting in increased accuracy in the final product [31].

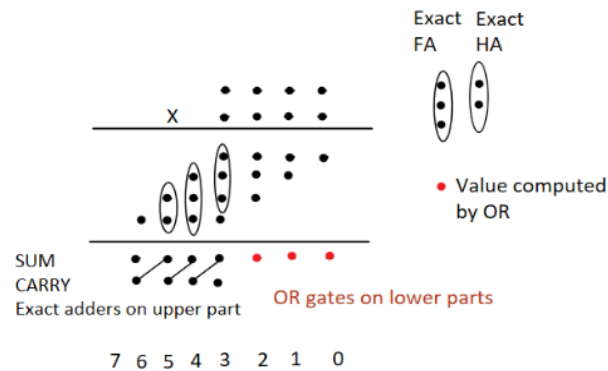


Figure 11. Dot diagram of approximate 4-bit multiplier with exact HA and FA

The approximate 16-bit multiplier architecture illustrated in Figure 12 is crafted using the four approximate 8-bit multiplier blocks from Figure 9. In Figure 12 the clock-gating circuit controls the LAB part of the final product which is $Z[14:0]$. 16-bit adder generates the MSB part $Z[32:15]$. The final product is the combination of $Z[14:0]$ and $Z[32:15]$. The CSA efficiently combines the partial products from the approximate 8-bit multiplier blocks to produce the higher bits of the result without introducing significant overhead.

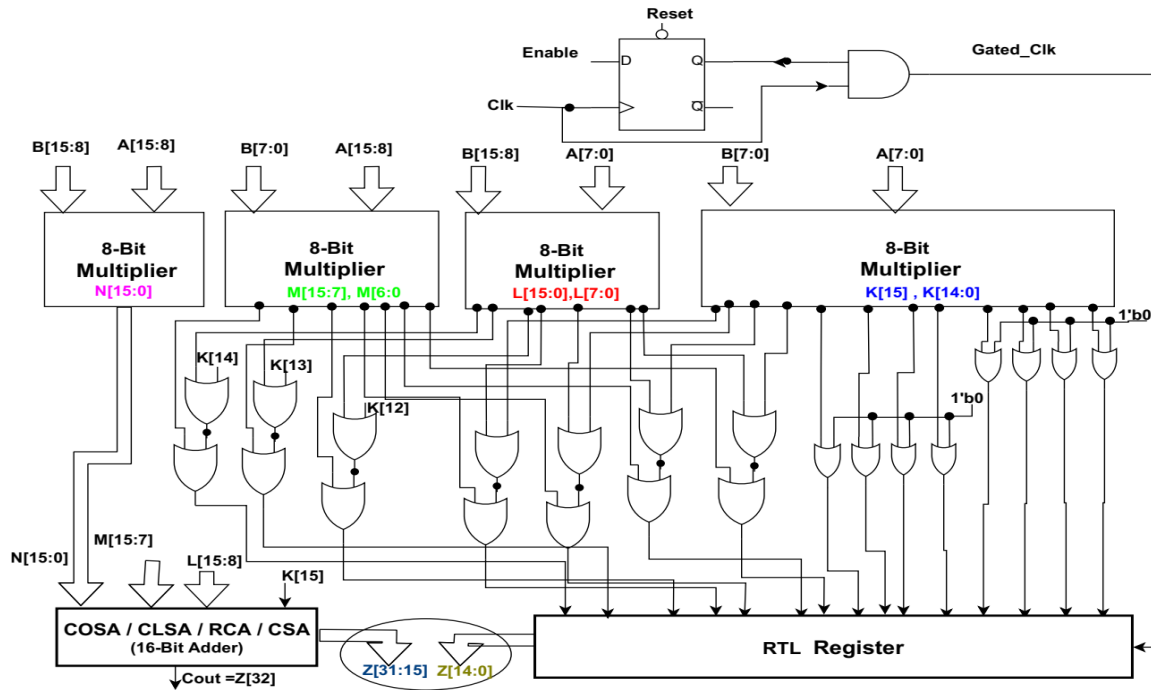


Figure 12. Architecture of 16-bit approximate multiplier using 8-bit multiplier and clock gating

4. RESULTS AND DISCUSSION

The simulation result of unsigned as well as signed 16-bit multipliers using inexact adders is shown in Figures 13 and 14. Figure 13 illustrates the simulation results of 16-bit unsigned approximate multiplier with a multiplicand $a[15:0]=16'd\ 2987$ and a multiplier $b[15:0]=16'd\ 50763$ and the final product is $(C_{out}, z[31:0])=32'd\ 2151383029$. Figure 14 illustrates the simulation results of 16-bit signed approximate multiplier with a multiplicand $a[15:0]=-16'd\ 6543$ and a multiplier $b[15:0]=-16'd\ 9776$ and the final product is $(C_{out}, z[31:0])=32'd\ 325384696$.

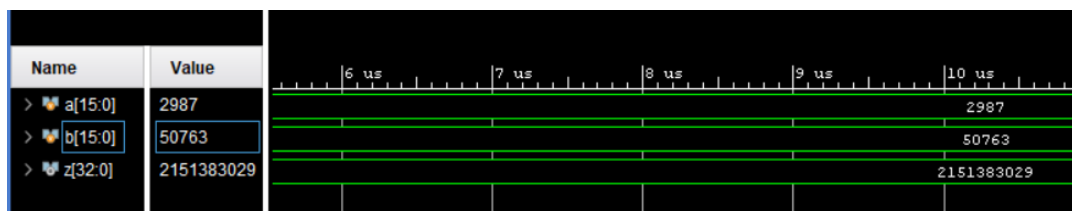


Figure 13. 16-bit unsigned approximate multiplier output with approximate adders

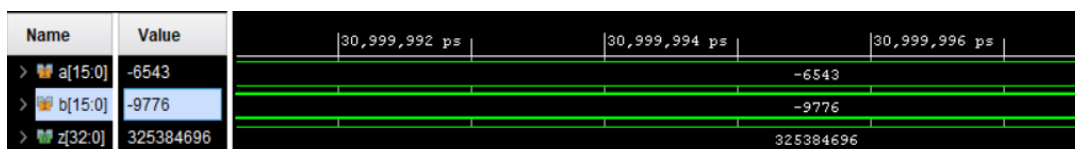


Figure 14. 16-bit signed approximate multiplier output with approximate adders

The simulation results of unsigned and signed 16-bit approximate multipliers using clock gating and exact adders for MSB part addition are illustrated in Figures 15 and 16. Figure 15 illustrates the simulation results of 16-bit clk-gating unsigned approximate multiplier with a multiplicand $a[15:0]=16'd\ 2987$ and a multiplier $b[15:0]=16'd\ 50763$ and the final product is $(C_{out}, z[31:0])=32'd\ 210141055$. Figure 16 illustrates

the simulation results of 16-bit signed approximate multiplier with a multiplicand $a[15:0] = -16'd\ 6543$ and a multiplier $b[15:0] = -16'd\ 9776$ and the final product is $(C_{out}, z[31:0]) = 32'd\ 315850576$.

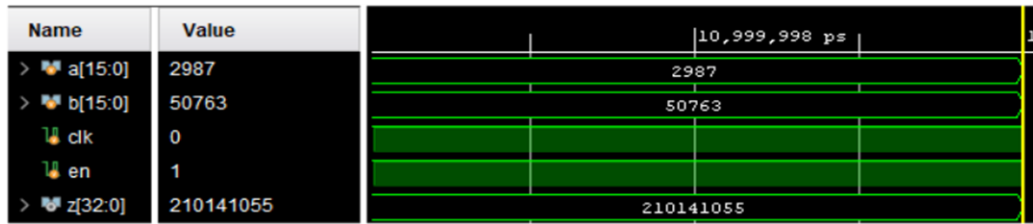


Figure 15. 16-bit unsigned approximate multiplier output with clk-gating and inexact adders

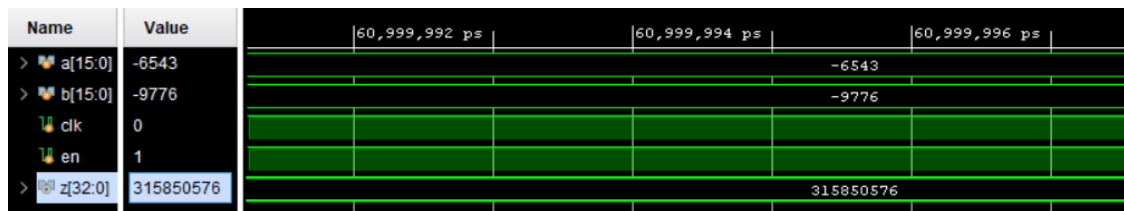


Figure 16. 16-bit signed approximate multiplier output with clk-gating and exact adders

4.1. Quantitative comparison among multipliers

In this section, we compared approximate multipliers (AM) described in previous sections in terms of power, area, and latency. Thus, the next subsections compare the designs in two categories: fixed-point unsigned and signed multipliers, it should be emphasized that all of the designs were assessed using the same environment. We applied approximation at the partial product accumulation stage to achieve less delay, lower power consumption, and increased area efficiency, as well as to improve the precision of the final product obtained with COSA/CSLA/RCA/CAS. Tables 3 and 4 show the delay/power/area comparison of 4-bit, 8-bit, and 16-bit signed and unsigned multipliers among four types of adders. The results found that the carry selects adder gave better results in some cases that are highlighted in green color. In general, at the same approximation level, CLSA has a low delay, RCA has a low area, and COSA has low power consumption than another adder.

Table 3. Unsigned approximate multiplier parameters

Adders	Power (w)			Area (LUT)			Critical path delay (ns)		
	4-bit	8-bit	16-bit	4-bit	8-bit	16-bit	4-bit	8-bit	16-bit
COSA	0.141	0.17	0.237	11	67	305	5.343	7.894	10.693
CSLA	0.141	0.17	0.258	11	61	275	5.336	6.911	9.816
RCA	0.141	0.174	0.277	11	60	260	5.343	7.302	10.776
CSA	0.142	0.178	0.316	11	66	293	5.343	7.565	11.67

*LUTs: lookup tables

Table 4. Signed approximate multiplier parameters

Adders	Power (w)			Area (LUT)			Critical path delay (ns)		
	4-bit	8-bit	16-bit	4-bit	8-bit	16-bit	4-bit	8-bit	16-bit
COSA	0.141	0.182	0.27	10	71	321	5.343	7.792	11.39
CSLA	0.141	0.186	0.289	10	74	310	5.343	7.588	10.453
RCA	0.141	0.187	0.315	12	65	315	5.795	7.35	11.128
CSA	0.141	0.181	0.313	11	68	294	5.343	7.575	11.579

Figures 17 to 19 compare the multiplier circuit parameters of unsigned and signed multipliers designed using different adders at the final stage. This comparison found that the sensitivity of the approximation approaches to the synthesis mode, has not been extensively demonstrated in previous studies. Figure 17 depicts the power consumption of signed and unsigned four-bit, eight-bit, and sixteen-bit

multipliers. A multiplier with a COSA adder at the final stage utilized less power than one with a RCA/CSLA/CSA adder at the final stage.

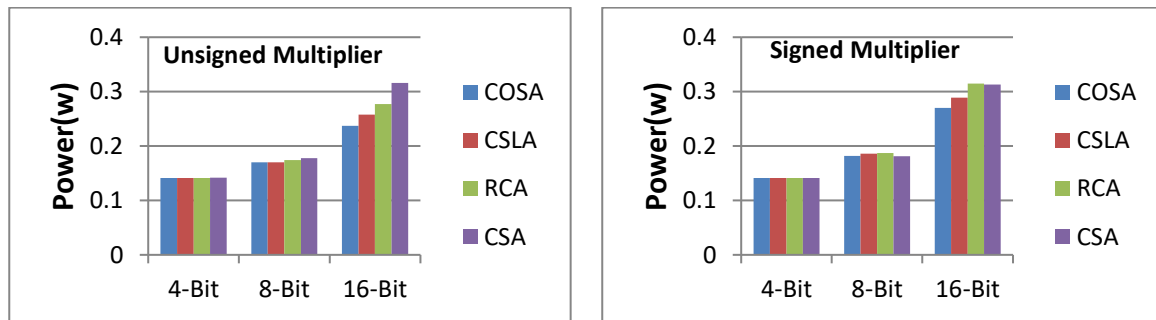


Figure 17. Approximate multipliers power characteristics comparison

Figure 18 depicts the area occupied by the signed and unsigned 4-bit, 8-bit, and 16-bit multipliers. The multiplier with RCA at the last consumed less power than the COSA/CSA/CSLA adders at the end stage. Figure 19 shows the delay induced by the signed and unsigned 4-bit, 8-bit, and 16-bit multipliers. A multiplier with CLSA at the last stage produced less delay than COSA/CSA/CSLA at the end stage.

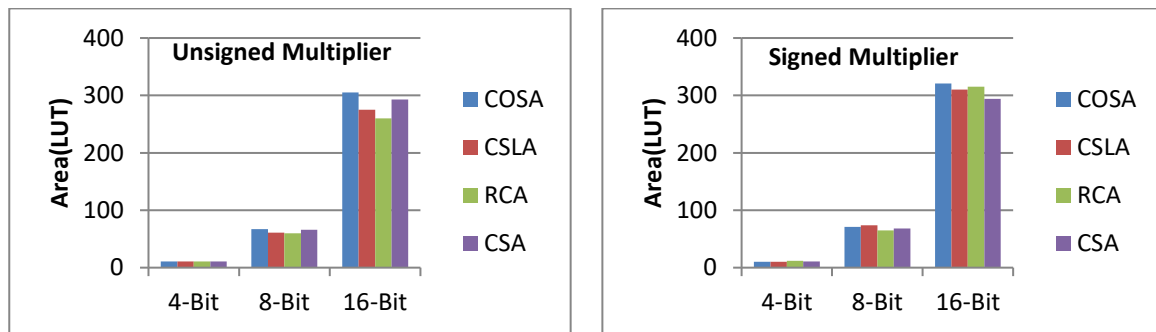


Figure 18. Approximate multipliers area characteristics comparison

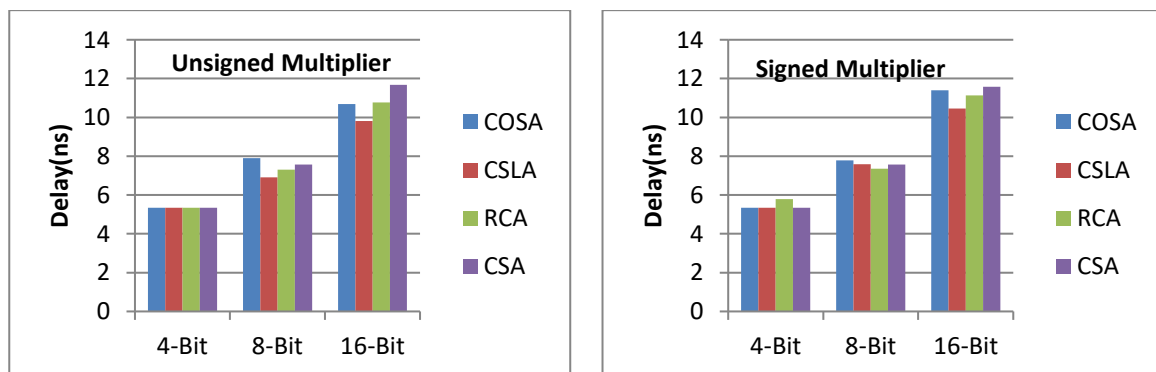


Figure 19. A approximate multipliers delay characteristics comparison

Table 5 shows the delay/power/area comparison of 16-bit signed and unsigned multipliers. The results found that the carry selected adder gave better results in some cases that are highlighted in green color. In general, at the same approximation level, CLSA has a low delay, RCA has low area, and COSA has a lower power consumption than the other adder.

From Tables 5 to 7, the proposed clock gating technique design proves to be better than the existing inexact adder approximate multiplier design as it reduces power dissipation by 13% and 18.18% for unsigned and signed 16-bit clock gating multiplier, the only slight increases in area and delay dissipation which remains as a trade-off.

Table 5. 16-bit approximate multiplier using clock gating and exact adders

Adder type	Unsigned 16-bit multiplier			Signed 16-bit multiplier		
	Power (W)	Area (LUT)	Delay (ns)	Power (W)	Area (LUT)	Delay (ns)
COSA	0.237	305	10.693	0.271	321	11.390
CSLA	0.258	275	09.816	0.289	310	10.453
RCA	0.277	260	10.776	0.315	315	11.128
CSA	0.277	260	10.773	0.303	287	11.307

Table 6. Unsigned 16-bit approximate multiplier parameter comparison for CSA

Multiplier characteristics	In-exact adders	Exact adder and clk-gating	% change
Power (w)	0.313	0.277	13.00
Delay (ns)	10.267	10.773	-4.70
Area (LUTs)	221	260	-15.00

Table 7. Signed 16-bit approximate multiplier parameter comparison for CSA

Multiplier characteristics	In-exact adders	Exact adder and clk-gating	% change
Power (w)	0.360	0.303	18.81
Delay (ns)	10.903	11.307	-3.57
Area (LUTs)	246	287	-14.29

Calculating the proportion of power dissipation using (7) allows us to easily understand the power efficiency of the suggested multiplier.

$$\text{Power}(\%) = \frac{\text{Power utilization}_{\text{inexact}} - \text{Power utilization}_{\text{exact}}}{\text{Power utilization}_{\text{inexact}}} \quad (7)$$

Where inexact is inexact adders approximate multiplier and exact is exact adder approximate multiplier.

5. CONCLUSION

The paper outlines the design of both unsigned and signed approximate 16-bit multipliers using the clock-gating technique. The results found that the multiplier accuracy improved and power dissipation decreased along with the trade-off between the parameters power, area, and delay it includes the design of clock-gating, which is crafted using Verilog hardware description language (HDL) and simulated on Artix-7 AC701 FPGA in Xilinx Vivado Tool. The proposed design proves to be better than the existing design as it produces more accurate results and reduces power dissipation by 13% and 18.18% for unsigned and signed approximate multipliers respectively along with small changes in in area and delay. In the future research, the proposed multipliers will be used in image processing applications such as image multiplication, smoothing, and sharpening to produce high-quality images with error tolerance.

ACKNOWLEDGMENTS

The authors would like to thank the management of Mohan Babu University for providing the necessary infrastructure to conduct this research. We would want to express our gratitude to the staff and students of the Department of Electronics and Communication Engineering.




REFERENCES

- [1] Z. Wang, G. Zhang, J. Ye, J. Jiang, F. Li, and Y. Wang, "Accurate reliability analysis methods for approximate computing circuits," *Tsinghua Science and Technology*, vol. 27, no. 4, pp. 729–740, Aug. 2022, doi: 10.26599/TST.2020.9010032.
- [2] F. Ferdaus, B. M. S. B. Talukder, and M. T. Rahman, "Approximate MRAM: High-performance and Power-efficient Computing with MRAM Chips for Error-tolerant Applications," *IEEE Transactions on Computers*, pp. 1–1, 2022, doi: 10.1109/TC.2022.3174584.
- [3] T. Krishnan, P. Anguraj, S. S. V. A., and S. K., "Design of Area Efficient Unified Binary/Decimal Adder/Subtractor Using Triple Carry Based Prefix Adder," in *2022 8th International Conference on Advanced Computing and Communication Systems*




Design and evaluation of clock-gating-based approximate multiplier for ... (Chowdam Venkata Sudhakar)

- (ICACCS), Mar. 2022, pp. 1720–1725, doi: 10.1109/ICACCS54159.2022.9785045.
- [4] S. A. Salehi, “Low-Cost Stochastic Number Generators for Stochastic Computing,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 4, pp. 992–1001, Apr. 2020, doi: 10.1109/TVLSI.2019.2963678.
 - [5] V. A and R. Dhavse, “Design of High Accuracy, Power Efficient and Area Efficient 16x16 Approximate Multiplier,” in *2020 IEEE 17th India Council International Conference (INDICON)*, Dec. 2020, pp. 1–6, doi: 10.1109/INDICON49873.2020.9342223.
 - [6] N. Mehmood, M. Hansson, and A. Alvandpour, “An Energy-Efficient 32-bit Multiplier Architecture in 90-nm CMOS,” in *2006 NORCHIP*, Nov. 2006, pp. 35–38, doi: 10.1109/NORCHIP.2006.329239.
 - [7] S. Ullah and A. Kumar, *Approximate Arithmetic Circuit Architectures for FPGA-based Systems*. Cham: Springer International Publishing, 2023, doi: 10.1007/978-3-031-21294-9.
 - [8] Kyung-Ju Cho, Kwang-Chul Lee, Jin-Gyun Chung, and K. K. Parhi, “Design of low-error fixed-width modified booth multiplier,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 5, pp. 522–531, May 2004, doi: 10.1109/TVLSI.2004.825853.
 - [9] H. Waris, C. Wang, W. Liu, J. Han, and F. Lombardi, “Hybrid Partial Product-Based High-Performance Approximate Recursive Multipliers,” *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 1, pp. 507–513, Jan. 2022, doi: 10.1109/TETC.2020.3013977.
 - [10] S. Venkatachalam and S.-B. Ko, “Design of Power and Area Efficient Approximate Multipliers,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 5, pp. 1782–1786, May 2017, doi: 10.1109/TVLSI.2016.2643639.
 - [11] J. Han and M. Orshansky, “Approximate computing: An emerging paradigm for energy-efficient design,” in *2013 18TH IEEE EUROPEAN TEST SYMPOSIUM (ETS)*, May 2013, pp. 1–6, doi: 10.1109/ETS.2013.6569370.
 - [12] S. Immareddy and A. Sundaramoorthy, “A survey paper on design and implementation of multipliers for digital system applications,” *Artificial Intelligence Review*, vol. 55, no. 6, pp. 4575–4603, Aug. 2022, doi: 10.1007/s10462-021-10113-0.
 - [13] D. T. Raju and Y. S. Rao, “Investigation of Error-Tolerant Approximate Multipliers for Image Processing Applications,” 2022, pp. 357–370, doi: 10.1007/978-981-19-2130-8_29.
 - [14] J. Ramasamy and S. Nagarajan, “Hybrid Segment Approximate Multiplication for Image Processing Applications,” *Circuits and Systems*, vol. 07, no. 08, pp. 1701–1708, 2016, doi: 10.4236/cs.2016.78147.
 - [15] M. Osta, A. Ibrahim, H. Chible, and M. Valle, “Approximate Multipliers Based on Inexact Adders for Energy Efficient Data Processing,” in *2017 New Generation of CAS (NGCAS)*, Sep. 2017, pp. 125–128, doi: 10.1109/NGCAS.2017.41.
 - [16] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, “Low-Power Digital Signal Processing Using Approximate Adders,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 1, pp. 124–137, Jan. 2013, doi: 10.1109/TCAD.2012.2217962.
 - [17] P. Puneeth, A. S. Raghuvanshi, and S. Yadav, “Design and Implementation of High Frequency 16-bit full adder on FPGA Families,” in *2023 4th International Conference for Emerging Technology (INCET)*, May 2023, pp. 1–7, doi: 10.1109/INCET57972.2023.10170506.
 - [18] V. J. Arulkarthick and A. Rathinaswamy, “Delay and area efficient approximate multiplier using reverse carry propagate full adder,” *Microprocessors and Microsystems*, vol. 74, p. 103009, Apr. 2020, doi: 10.1016/j.micpro.2020.103009.
 - [19] M. Priyadharshni and S. Kumaravel, “A Comparative Exploration About Approximate Full Adders for Error Tolerant Applications,” 2019, pp. 61–74, doi: 10.1007/978-981-13-5950-7_6.
 - [20] B. Koyada, N. Meghana, M. O. Jaleel, and P. R. Jeripotula, “A comparative study on adders,” in *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, Mar. 2017, pp. 2226–2230, doi: 10.1109/WiSPNET.2017.8300155.
 - [21] B. S. Naik and C. V. Sudhakar, “Study, Implementation and Comparison of Different Multipliers Based on Array, Vedic and KCM using Squarer Mathematics using EDA Tools,” *International Journal of VLSI System Design and Communication Systems*, vol. 2, no. 4, pp. 0250–0255, 2014.
 - [22] K. N. B. Gowthami and C. V. Sudhakar, “Design and Simulation of an Efficient Vedic Booth Multiplier,” *International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)*, vol. 4, no. 4, pp. 914–919, 2015.
 - [23] P. Anguraj and T. Krishnan, “Design and realization of area-efficient approximate multiplier structures for image processing applications,” *Microprocessors and Microsystems*, vol. 102, p. 104925, Oct. 2023, doi: 10.1016/j.micpro.2023.104925.
 - [24] Z. Li *et al.*, “Adaptable Approximate Multiplier Design Based on Input Distribution and Polarity,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 12, pp. 1813–1826, Dec. 2022, doi: 10.1109/TVLSI.2022.3197229.
 - [25] Y. Guo, H. Sun, and S. Kimura, “Design of Power and Area Efficient Lower-Part-OR Approximate Multiplier,” in *TENCON 2018 - 2018 IEEE Region 10 Conference*, Oct. 2018, pp. 2110–2115, doi: 10.1109/TENCON.2018.8650108.
 - [26] G. G. Kumar, C. V. Sudhakar, and M. N. Babu, “Design of high-speed Vedic square by using Vedic multiplication techniques,” *International Journal of Scientific & Engineering Research*, vol. 4, no. 1, pp. 1–4, 2013.
 - [27] E. Zacharelos, I. Nunziata, G. Saggese, A. G. M. Strollo, and E. Napoli, “Approximate Recursive Multipliers Using Low Power Building Blocks,” *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 3, pp. 1315–1330, Jul. 2022, doi: 10.1109/TETC.2022.3186240.
 - [28] G. Thakur, H. Sohal, and S. Jain, “Power–Area-Optimized Approximate Multiplier Design for Image Fusion,” *Circuits, Systems, and Signal Processing*, vol. 43, no. 4, pp. 2288–2319, Apr. 2024, doi: 10.1007/s00034-023-02559-0.
 - [29] T. Lang, E. Musoll, and J. Cortadella, “Individual flip-flops with gated clocks for low power datapaths,” *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 44, no. 6, pp. 507–516, Jun. 1997, doi: 10.1109/82.592586.
 - [30] Y. Wu *et al.*, “A Survey on Approximate Multiplier Designs for Energy Efficiency: From Algorithms to Circuits,” *ACM Transactions on Design Automation of Electronic Systems*, vol. 29, no. 1, pp. 1–37, Jan. 2024, doi: 10.1145/3610291.
 - [31] S. Chandaka and B. Narayanam, “Hardware Efficient Approximate Multiplier Architecture for Image Processing Applications,” *Journal of Electronic Testing*, vol. 38, no. 2, pp. 217–230, Apr. 2022, doi: 10.1007/s10836-022-06000-3.




BIOGRAPHIES OF AUTHORS

Dr. Chowdam Venkata Sudhakar    received B.Tech., degree in Electronic Instrumentation and Control Engineering from Sri Venkateswara University, Tirupati, Andhra Pradesh, India in 2006. He received M.Tech., degree in Digital Systems and Computer Electronics from J.N.T.U.H. Kukatpally, Hyderabad, Telangana India in 2010 and received Ph.D. in the Department of Electronics and Communication Engineering from Sri Venkateswara University in 2024. Currently working as an Assistant Professor in the Department of Electronics and Communication Engineering, at Mohan Babu University (Erstwhile Sree Vidyanikethan Engineering College). His areas of interests include VLSI architectures for image processing, sensors, and signal processing, and remote sensing imagery data analysis. He can be contacted at email: sudhakar.chowdam@gmail.com.



Dr. Suresh Babu Potlady    received B.Tech. degree in Electronic and Communication Engineering from Sri Venkateswara University, Tirupati, Andhra Pradesh, India in 2005. He received M.Tech. degree in Electronics Instrumentation and Communication Systems from Sri Venkateswara University, Tirupati, Andhra Pradesh, India in 2008 and received Ph.D. in the Department of Electronics and Communication Engineering from Sri Venkateswara University in 2024. He published more than 15 papers in various reputed Journals and Conferences. Currently working as an Associate Professor in the Department of Electronics and Communication Engineering, at Sri Venkateswara College of Engineering (Autonomous), Tirupati. His areas of interests include radar signal processing, VLSI design, digital image processing, embedded systems, and IoT. He can be contacted at email: sureshbabu.413@gmail.com.



Prasad Reddy Karipireddy    received B.Tech. degree in Electronic Instrumentation and Control Engineering from Sri Venkateswara University, Tirupati, Andhra Pradesh, India in 2006. He obtained his Master of Science (M.S.) degree in Machine Automation from Tampere University of Technology, Tampere, Finland in 2010. He is currently working as a Functional Safety and Cyber security expert at Hult Oy, Espoo, Finland. He is a certified expert in Functional safety and Cyber security by TÜV SÜD. His areas of interests include functional safety and cyber security in oil and gas, chemical, pulp and paper, automotive, machinery, and rail industries. He can be contacted at email: karipireddyprasadreddy@gmail.com.