International Journal of Scientific Research in Science and Technology
Print ISSN: 2395-6011 | Online ISSN: 2395-602X (www.ijsrst.com)
doi : https://doi.org/10.32628/IJSRST

Implementation of Segmentation Based Efficient Imprecise Multipliers with Mux Based Full Adders

Sunita Reddy[1], Mahesh R. K[2].

[1]M. Tech Scholar, Department of ECE (VLSI& ES), Sharanbasva University, Kalaburagi, India

[2]Assistant Professor, Department of ECE, Sharanbasva University, Kalaburagi, India

Article Info
Volume 9, Issue 4
Page Number : 173-181

Publication Issue
July-August 2022

Article History
Accepted : 05 July 2022
Published : 14 July 2022

ABSTRACT

We offer a new method for multiplying two unsigned binary values using a Segmentation-based Approximate Multiplier in this paper. The primary problem for approximate multiplier systems is to decrease the area and latency while avoiding substantial errors. Almost every approximation multiplier on the market divides the input operands in order to take advantage of flow parallelism. The suggested design shrinks a Partial Products Matrix of the order of nx(2n-1) to a Reduced Partial Product Matrixof the order of 4x2n. It also removes the need for additional hardware for partial product compression and rearrangement. Along with this effort, we also present -SAM, an optimized version of our fundamental concept. SAM reduces the basic design's on-chip space and power use even more. Xilinx Vivado is used to simulate and synthesize the efficiency of the suggested technique.

Keywords: - Approximate multiplier, Partial product matrix, Reduced partial product matrix, SAM, Xilinx Vivado.

I. INTRODUCTION

Partially generated products, partially reduced products, and propagation of carry are the basic process involved in multiplier. Hence, approximations can be used in any one of these blocks. In the design of computing systems, energy efficiency has become a top priority. The major goal of this presentation is to evaluate recent advances in approximation computing (AC). Fortunately, most of these programs include an inherent error-resilience feature. Furthermore, most of these applications do not necessitate the computation of a single or golden numerical value.

Approximate computing purposely incorporates acceptable faults in the process and are energy-efficient. The Dennard's scaling yields declining returns as technology advances, taking use of the new source of energy-efficiency afforded by approximation computing is becoming increasingly vital.

Because of the huge volumes of data and intricate computations necessary in these applications, a new issue has emerged as big data processing and artificial intelligence become more important. To accelerate the development of these new technologies, energy-efficient and high-performance general-purpose

compute engines, as well as application-specific integrated circuits, are in great demand. It is not necessary to always have exact output sometimes less exact output can compensate for each other or have no substantial impact on the computed results. As a result, approximation computing (AC) has evolved as a novel method to energy-efficient design as well as boosting the performance of a computing system with little accuracy loss.

Approximate computing is a new trend in digital design that foregoes the need for accurate calculation in favor of increased speed and power. To test the performance of the suggested compressors, this study provides new approximate compressors with 8 bit and 16 bit multiplier designs. The suggested circuits give greater power or speed for a target precision when compared to previously describe approximated multipliers.

Approximate computing has developed as a new paradigm for circuit and system design that is both high-performance and energy-efficient. With so many approximate arithmetic circuits being presented, it's become vital to comprehend a design or approximation approach for a given application in order to maximize performance and energy economy while minimizing accuracy loss. This paper attempts to offer a detailed overview and comparison of newly developed approximation arithmetic circuits under various design restrictions. Approximate adders, multipliers, and dividers are synthesized and described under performance and area optimizations. After that, the error and circuit characteristics are generalized for various design classes. The circuits with lower error rates or error biases perform better in simple computations, such as the sum of products, whereas more complex accumulative computations that involve multiple matrix multiplications and convolutions are vulnerable to single-sided errors that result in a large error bias in the computed result. Because addition mistakes are more sensitive than multiplication errors in such complicated calculations, multipliers may tolerate a larger approximation than adders. Approximation is used to reduce the overhead on calculation units of a processor, resulting in improved performance and efficiency. The speed of operation, which is inversely related to the system delay, necessitates massive parallel processes, which consume a lot of hardware and energy. By lowering the accuracy and dependability of the system, energy and space efficient solutions may be realized. Approximate computing has emerged as a potential approach for maintaining the right mix of latency, area, and power.

Faster systems with reduced design complexity and power consumption emerge from arithmetic processes that are approximated. The trade-off would be a loss of accuracy, which would not necessarily impede machine learning and multimedia applications in their usual operation. These kind of applications make good use of the technology. To improve the efficiency of approximation arithmetic units, much research has been carried out. Partial product summation has unquestionably been the most significant source of power consumption and system delay in multiplication operations. Compressors have been shown to minimize the time required for partial product summing, according to research. Half adders and/or complete adders are used by compressors to estimate the number of logic 1 in the input. Also it is available with a significant rise in power consumption, lowering device lifetime and reliability

Due to restricted human perception there's a chance that full precision computation blocks will be replaced with approximation ones.

II. EARLIER WORK

They created an array multiplier with accurate adders in the previous technique. There are three inputs and two outputs in these precise adders. Due to the demand for faster computer arithmetic essential for the rapidly expanding processor architectures, research on the design of high-speed arithmetic circuits exploded in the middle of the 1960s.

International research concentrated in particular on the creation of parallel digital multiplier circuits, which were significantly less optimised than adder circuits. Luigi Dadda's 1965 paper, along with C. S. Wallace's, is one of the two most important contributions to the construction of optimal parallel digital multipliers for fixed-point binary values. Fixed-point multiplication is the most fundamental and common type of multiplication, and it's used everywhere, either directly or as part of floating-point multiplication.

The essential scientific notion behind L. Dadda's work is that the summation of the carry bits in the partial product matrix of the multiplication may be effectively deferred and distributed in order to decrease the number of additions and the propagation delay. In comparison to prior methods, such as the Wallace one, which comes just after it, this parallel multiplier approach dramatically reduces the number of logic gates utilised by the circuit (1964).

Not only that, but the theory behind the Dadda multiplier technique, namely postponing and dispersing carry propagation, is relatively novel in comparison to the methods in use at the time, and it has been heavily used in future computer arithmetic advancements. Along with the Wallace scheme, the Dadda scheme for parallel multipliers has become one of the two well recognised topologies for such a fundamental arithmetic circuit type.

Since then, the so-called Dadda trees, together with the Wallace trees, have been one of the two basic parallel multiplier systems depicted in every computer arithmetic textbook and taught in university computer arithmetic courses.

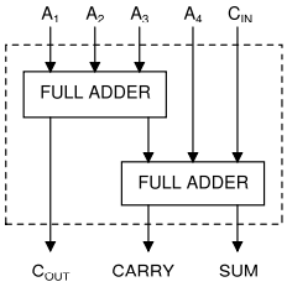

Fig 1: Exact compressor

From the Figure 1, A1, A2, A3, A4, and CIN are its inputs. The letters COUT, CARRY, and SUM are all capitalized.

$$C_{OUT} = A_3(A_1 \oplus A_2) + A_1(\overline{A_1 \oplus A_2}) \quad (1)$$

$$CARRY = C_{IN}(A_1 \oplus A_2 \oplus A_3 \oplus A_4) + A_4(\overline{A_1 \oplus A_2 \oplus A_3 \oplus A_4}) \quad (2)$$

$$SUM = C_{IN} \oplus A_1 \oplus A_2 \oplus A_3 \oplus A_4 \quad (3)$$

Figure 2 shows a compressor chain. The input carry from the previous 4: 2 compressor, which handled the lower significant bits, is represented by CIN. CARRY and COUT are order '1' outputs with more importance than the input CIN.

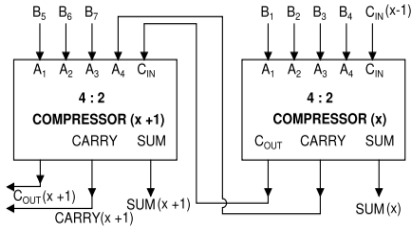

Fig 2: Compressor chain

Table 1: Truth table of Exact Compressor

A_1	A_2	A_3	A_4	C_{IN}	C_{OUT}	$CARRY$	SUM
0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	1
0	0	0	1	0	0	0	1
0	0	0	1	1	0	1	0
0	0	1	0	0	0	0	1
0	0	1	0	1	0	1	0
0	0	1	1	0	0	1	0
0	0	1	1	1	0	1	1
0	1	0	0	0	0	0	1
0	1	0	0	1	0	1	0
0	1	0	1	0	0	1	0
0	1	0	1	1	0	1	1
0	1	1	0	0	1	0	0
0	1	1	0	1	1	0	1
0	1	1	1	0	1	0	1
0	1	1	1	1	1	1	0
1	0	0	0	0	0	0	1
1	0	0	0	1	0	1	0
1	0	0	1	0	0	1	0
1	0	0	1	1	0	1	1
1	0	1	0	0	1	0	0
1	0	1	0	1	1	0	1
1	0	1	1	0	1	0	1
1	0	1	1	1	1	1	0
1	1	0	0	0	1	0	0
1	1	0	0	1	1	0	1
1	1	0	1	0	1	0	1
1	1	0	1	1	1	1	0
1	1	1	0	0	1	0	1
1	1	1	0	1	1	1	0
1	1	1	1	0	1	1	0
1	1	1	1	1	1	1	1

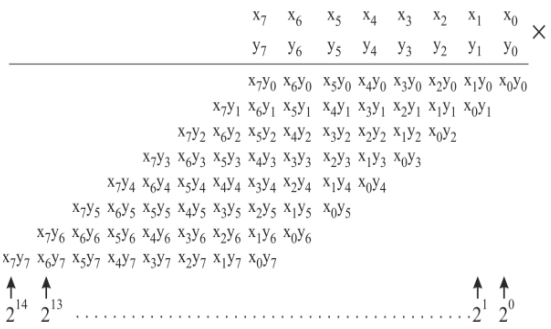

Fig 3 : Partial product matrix of a 8 × 8 bit Multiplier.

$$S = \sum \{p_0, p_1, p_2, \ldots p_{j-1}\}$$

In both Digital Signal Processors and Microprocessors, the multiplier is critical for executing arithmetic operations. The efficient and effective Multiplication Algorithm must be used to improve the performance characteristics of either DSPs or Microprocessors. The multiplier is one of the most fundamental components of digital systems.

Multipliers also have a role in the digital system's computation speed and power consumption. As a result, for a digital system, it is critical to build a high-speed multiplier with low power dissipation. As a result, it can improve the digital systems' efficiency.

Compressing the columns Because of their high processing speed, multipliers have gained favor. The well-known Column Compression Multipliers are Wallace and Dadda. In 1964, Chris Wallace, a computer scientist from Australia, proposed the Wallace Multiplier.

Luigi Dadda, an Italian computer engineer, proposed modifying the Wallace Multiplier to create the Dadda Multiplier. Wallace and Dadda Multipliers are both based on reduction. A [3,2] counter (Full adder) and a [2,2] counter are used to compress the columns, resulting in a decrease (Half adder). Wallace and Dadda Multipliers have three steps in common.

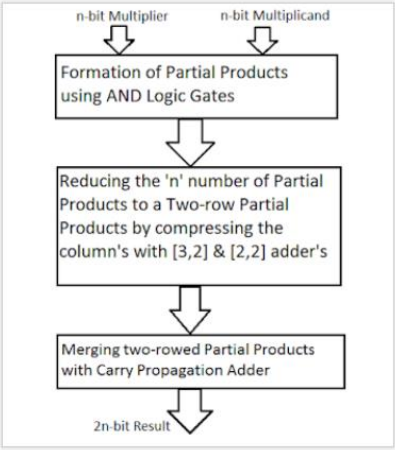

Fig 4: Wallace and Dadda Multipliers have three steps

Dadda devised a reduction approach that produces reduced two-row Partial products with the fewest possible reduction phases. Dadda was able to do this by strategically positioning the [3,2] and [2,2] counters in the maximum Critical route.

A N by N partial product is produced by an N-bit multiplier and multiplicand. A Matrix is made up of these incomplete products. Through a series of reduction processes, Dadda decreased the height of these Matrix to a two-row matrix.

Algorithm:

1. Assume that the last two-row matrix height is d1 = 2, and that the subsequent matrix heights are calculated using dj+1 = 1.5 * dj, where j = 1,2,3,4,..... In this matrix height, rounding down to the smallest fraction should be done. 13.5 = 13 is one example of this (rounded). The heights of the matrices will be as follows: 2, 3, 4, 6, 9, 13, 19, 28,..... Finally, the greatest dj should be acquired so that the height of the resulting matrix does not exceed the total height of the matrix.
2. During the first reduction step, use the [3, 2] and [2, 2] counters to compress the columns so that the resulting reduced matrix height does not exceed dj.
3. The total should be transferred to the same column in the next reduction step, and the

carry should be given to the next column, during compression.

4. Repeat steps 2–4 until you have a two-row reduced matrix.

After applying approximation to the traditional PPM, a Reduced Partial Products Matrix (R-PPM) is created. The 4 2n R-PPM is then split into two parts. The n most significant bits are on the left half, while the n least significant bits are on the right. Parallel compression is applied to these two parts. The design of a traditional n-bit multiplier consists of three steps:

(i) Generation of partial products through AND gates and formulation of PPM of dimension n × (2n – 1),
(ii) Compression, and rearrangement of PPM and,
(iii) Generation of the final result via accumulation.

The divide-and-conquer approach is used to create the proposed unsigned approximation multiplier SAM. Approximation is used in this design from the very beginning. The R-PPM of dimension 4 2n is created after the first stage has been finished, unlike the nx (2n -1) PPM.

General array multipliers were used to construct the multiplication of segmented bits, and full adders and half adders were used to reduce the partial products of the array multiplier.

Full adder:

Only two integers are added with the half adder. The complete adder was created in order to solve this issue. Three 1-bit binary values A, B, and carry C are added using the complete adder. For different sizes of array multipliers, the hardware requirements in terms of full adder (FA) and length of final adder (FAL) are calculated as shown below.

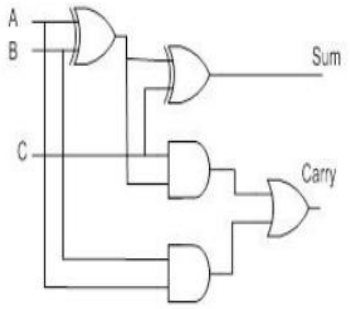

Fig 5: Full adder

The adder with three inputs and two outputs is known as a full adder. A and B are the initial two inputs, whereas C is the third. CARRY denotes the carry output, while SUM denotes the typical output.

Table 2 : Full adder

Inputs			Outputs	
A	B	C_{in}	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Array multiplier using FA and HA:

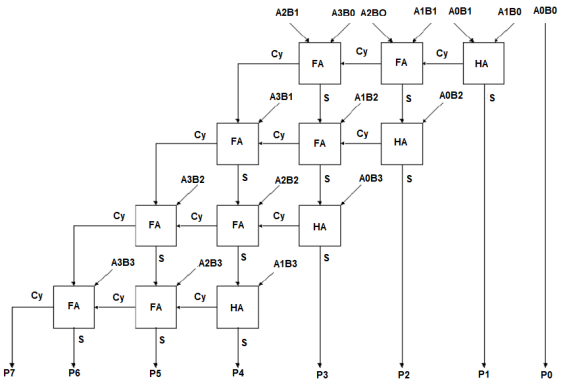

Fig 6: existing 4 bit array multiplier

III. PROPOSED METHOD

An approach for approximate multiplication of two unsigned binary values is provided in this project:

Segmentation based Approximate Multiplier (SAM). The suggested architecture takes use of parallelism in the compression step by segmenting partial products.

After applying approximation to the traditional PPM, we produce a Reduced Partial Products Matrix (R-PPM) in our suggested work. To make the design more power efficient, full adders that reduce PPM are replaced with modified full adders.

The divide-and-conquer approach is used to create the proposed unsigned approximation multiplier SAM. Approximation is used in this design from the very beginning. The R-PPM of dimension 4 2n is created after the first stage has been finished, unlike the n (2n 1) PPM.

The two n-bit input operands A and B are used in our n-bit approximation multiplier. These input operands are separated into two halves, AH, BH (most significant bits) and AL, BL (least significant bits), so that A and B may be expressed as a linear combination of AH, AL and BH, BL, as shown in the equation.

$$A = A_H \times 2^{n/2} + A_L, B = B_H \times 2^{n/2} + B_L \quad (1)$$

From Equation 1, it is clear that AH, AL, BH and BL are n/2 bit binary numbers. Now A × B can be written as,

$$A \times B = A_H \times B_H \times 2^n + (A_H \times B_L + A_L \times B_H) \times 2^{n/2} + A_L \times B_L \quad (2)$$

In Equation 2, since the term AL×BL contributes significantly less to the final result, it is approximated as AL|BL. So, Equation 2 can be modified as,

$$A \times B = A_H \times B_H \times 2^n + (A_H \times B_L + A_L \times B_H) \times 2^{n/2} + (A_L | B_L) \quad (3)$$

The four rows of the R-PPM generated using Equation 3 are shown in Figure 7.

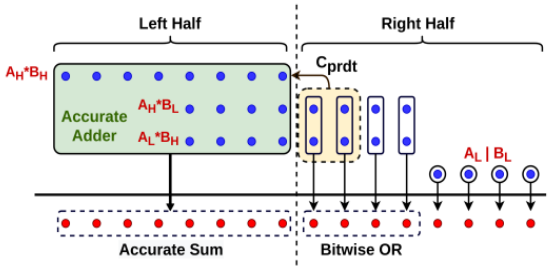

Fig 7: Compression of R-PPM

2n bits of AHxBH 2n make up the first row of R-PPM. This may be translated as n-most significant bits received from AHxBH, followed by n-zeros on the least significant side owing to n-bit shifting. R-second PPM's and third rows each have 3n/2 bits of AHxBL 2n/2 and ALxBH 2n/2. This may be decoded as an n-bit product of AHxBL and Ax BH, shifted by n/2 bits, and then examined. The n-bit logical OR of AL and BL is found in R-fourth PPM's and final row. Unlike standard PPMs of order nx(2n-1), our suggested approach compresses to a 4x2n matrix for every bit configuration.

Compression and Accumulation: Divide and Conquer Approach

The detailed compression mechanisms are discussed below:

1) Compression in right half: While the right half of the first row of the R-PPM comprises zero entries, the second and third rows of the right half have n/2 product entries followed by n/2 zero entries, as seen in Figure 1. Similarly, the fourth row has just n/2 items on the least significant side, as seen in Figure 1. The n/2 least significant bits of the final result are aggregated from these n/2 bits (in Figure 1, last row) on the rightmost side. The logical OR operation is applied to the n/2 significant entries of row 2 and row 3 to acquire the other most significant n/2 bits of the right half, and these n/2 bits are aggregated as the [n:n/2] bits of the final response.

2) Compression in Left Half: The R-most PPM's significant entries are seen in the left side of Figure 1. This half's compression strategy is precise, and it's implemented using an exact adder. Row 2 and 3's two most significant bits are utilised to approximate forecast the adder's carry input bit. The logic of carry prediction is presented in the next paragraphs.

Carry Prediction Logic: Because the carry input for the left half is independent of the preceding carry-ins, if the most significant bits of row 2 and row 3 are in the combination of (0, 0) or (1, 1), the carry input for the left half is always 0 or 1. However, because of the reliance on prior carry-ins, the carry input cannot be

reliably calculated if the most important bits are in the combination of (1, 0) or (0, 1). In these circumstances, the next most important bits are taken into account, and the same analysis is performed. Carry prediction logic may be expressed as follows using Boolean simplification:

Cprdt = A[n – 1] · B[n – 1]+ A[n – 2] · B[n – 2](A[n – 1] + B[n – 1]) (4)

In Equation 4, A[n – 1] and B[n – 1] are the most significant entries of the right half of row 2 and row 3 of Figure 3.1.

Similarly, A[n–2] and B[n–2] are the second-most significant entries of the right half of row 2 and row 3. The obtained n-bits are accumulated as [2n:n] bits of the final result.

SAM enhancements -SAM is a program that allows you to communicate with other people. The two halves of the input operands AH, AL, BH, and BL are further separated in our improved version of our fundamental design SAM. AHL and AHH are two subgroups of AH. The most major component of AH is denoted by AHH, whereas the least significant part is denoted by AHL. BHH and BHL are the two subcategories of BH. The abbreviations BHH and BHL have the same meaning.

This is done to replace the second term AH ×BL ×2 $^{n/2}$ and third term AL ×BH ×2 $^{n/2}$ of Equation 2 through new subparts. The AH × BL product of the second term can be re-written as

$$A_H \times B_L = A_{HH} \times B_{LH} \times 2^{n/2} + (A_{HH} \times B_{LL} + A_{HL} \times B_{LH} + A_{HL} \times B_{LL}) \times 2^{3n/4} \quad (5)$$

A similar expression can be written for the third term of Equation 2. In Equation 5, the terms other than the first term do not contribute significantly to the product. Therefore, these can be approximated and Equation 5 reduces to

$$A_H \times B_L = A_{HH} \times B_{LH} \times 2^{n/2} | ((A_{HH} \& B_{LL}) | (A_{HL} \& B_{LH}) | (A_{HL} \& B_{LL})) \times 2^{3n/4} \quad (6)$$

By incorporating approximation in the second and third terms of Equation 2, the optimized design -SAM focuses on decreasing the on-chip space and power. This version's R-PPM differs somewhat from that of the original SAM design. Instead of n significant bits, Row 2 and Row 3 of R-PPM will now contain just 3n/4 significant bits followed by zeroes (due to shifting), as in the SAM architecture. This version follows the same compression strategy as the SAM design.

Modified full adder:

The modified full adder is having two 4:1 multiplexer as shown in below figure 8. Using this the modified full adder decreases with low power consumption.

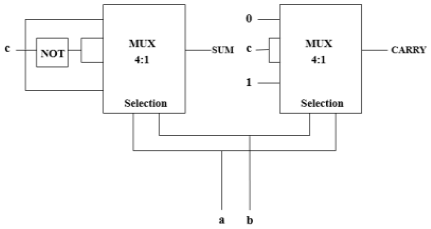

Fig 8: Proposed full adder multiplexers

A multiplexer is a circuit with 2n data inputs, 'n' selection lines, and a single output line. Using the values of the selection lines, one of these data inputs will be linked to the output. There will be 2n different combinations of zeros and ones because there are 'n' selection lines. As a result, just one data input will be chosen for each combination. Mux stands for multiplexer. Four data inputs (I3, I2, I1, and I0), two selection lines (s1 and s0), and one output (Y) make up the 4x1 Multiplexer.

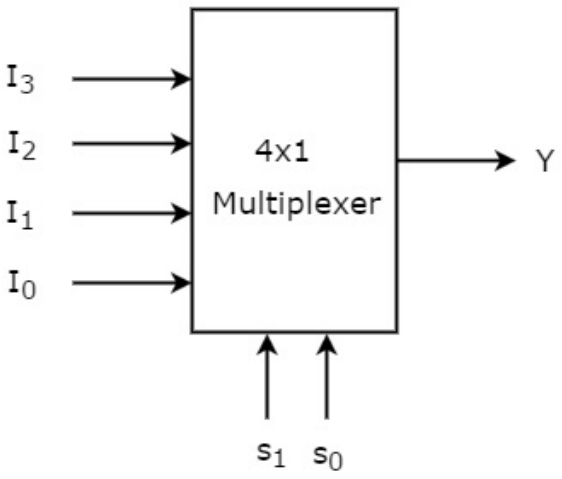

Fig 9: 4:1 multiplexer

IV. SIMULATED OUTPUTS:

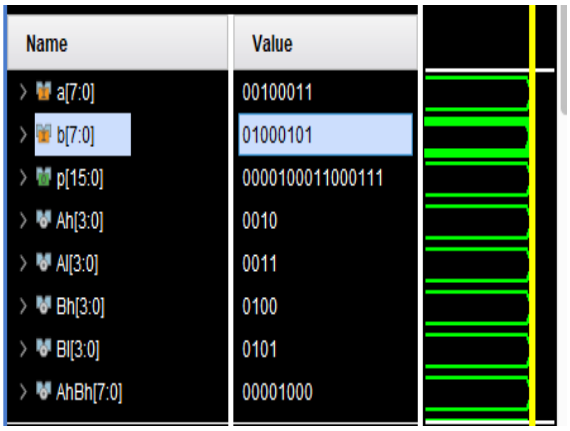

Fig 10: Simulation Result

Table 3 : Evaluation table:

	Area (LUT's)	Delay (ns)	Power (W)
RPPM	84	10.014	10.708
uSAM	53	10.029	9.181
Extension RPPM	64	10.368	10.559
Extension uSAM	52	9.935	9.114

V. CONCLUSION

In this paper, a modified full adder is provided as extension which enhances the parameters for the proposed implementation of various designs. Here we have proposed a novel method to multiply two unsigned binary numbers through SAM and μ-SAM. In this work, for n-bit multiplication, a reduced partial products matrix of dimension 4 × 2n is formulated rather than the conventional partial products matrix of dimension n×(2n–1). The R-PPM is equally divided into two segments and these segments are compressed in a non-blocking parallel manner, ensuring faster accumulation of result.

VI. REFERENCES

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

[2]. S. Vahdat, M. Kamal, A. Afzali-Kusha, and M. Pedram, TOSAM: An Energy-Efficient Truncation- and Rounding-Based Scalable Approximate Multiplier, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2019.

[3]. S. Vahdat, LETAM: A low energy truncation-based approximate multiplier, Computers Electrical Engineering, 2017.

[4]. O. Akbari, M. Kamal, A. Afzali-Kusha, and M. Pedram, Dual-Quality 4:2 Compressors for Utilizing in Dynamic Accuracy Configurable Multipliers, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2017.

[5]. A. Raha and V. Raghunathan, Towards full-system energy-accuracy tradeoffs: A case study of an approximate smart camera system*, in 2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC), 2017, pp. 1–6.

[6]. M. S. Ansari, B. F. Cockburn, and J. Han, A hardware-efficient logarithmic multiplier with improved accuracy, in Design, Automation Test in Europe Conference Exhibition (DATE), 2019.

[7]. L. Dadda, Some schemes for parallel multipliers, Alta Frequenza, vol. 34, pp. 349–356, 1965.

[8]. S. Hashemi, R. I. Bahar, and S. Reda, Drum: A dynamic range unbiased multiplier for approximate applications, in IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2015.

[9]. M. Osta, A. Ibrahim, H. Chible, and M. Valle, Approximate multipliers based on inexact adders for energy efficient data processing, in New Generation of CAS (NGCAS), 2017.

[10]. T. Yang, T. Ukezono, and T. Sato, A low-power high-speed accuracycontrollable approximate

multiplier design, in 23rd Asia and South Pacific Design Automation Conference (ASP-DAC), 2018.

[11]. A. Pandey, M. Reddy Karri, P. Yadav, N. K. Y.B., and V. M.H., Design and analysis of approximate multipliers for error-tolerant applications, in IEEE International Symposium on Smart Electronic Systems (iSES), 2018.

[12]. D. Pandey, S. Singh, V. Mishra, S. Satapathy and D. S. Banerjee, "SAM: A Segmentation Based Approximate Multiplier for Error Tolerant Applications," 2021 IEEE International Symposium on Circuits and Systems (ISCAS), 2021, pp. 1-5, doi: 10.1109/ISCAS51556.2021.9401266.

Cite this article as :

Sunita Reddy, Mahesh R. K, "Implementation of Segmentation Based Efficient Imprecise Multipliers with Mux Based Full Adders", International Journal of Scientific Research in Science and Technology (IJSRST), Online ISSN : 2395-602X, Print ISSN : 2395-6011, Volume 9 Issue 4, pp. 173-181, July-August 2022.
Journal URL : https://ijsrst.com/IJSRST229430