ISSN: 2278-4632
Vol-10 Issue-8 No. 1 August 2020

High Speed Approximate Multiplier with TOSAM Architecture

SALINA SUNIL YADAV (PG Scholar) [1],
Dr. YANAMANDRA. RAVI SEKHAR (Assistant Professor) [2]
SRI VENKATESWARA COLLEGE OF ENGINEERING AND TECHNOLOGY, SRIKAKULAM

ABSTRACT

The modern world is running forward in achieving the challenge of maximum speed. So many efficient structures are employed in the traditional designs so that the max speed can be included. Most of the structures will have multiplier as the basic blocks, which will run with less speed because of its huge structure. In practical, not all applications need accurate results such as in image processing and digital signal processing etc. so approximate multipliers are employed. By considering these two points called truncation- and rounding-based scalable approximate multiplier (TOSAM) is presented, which reduces the number of partial products by truncating each of the input operands based on their leading one-bit position. In the proposed design, multiplication is performed by shift, add, and small fixed-width multiplication operations resulting in large improvements in the speed compared to those of the exact multiplier. To improve the total accuracy, input operands of the multiplication part are rounded to the nearest odd number. Because input operands are truncated based on their leading one-bit positions, the accuracy becomes weakly dependent on the width of the input operands and the multiplier becomes scalable. Higher improvements in design parameter (speed) are observed.

Index Terms— Accuracy configurable, approximate multiplier, high speed, scalable, truncating.

I. INTRODUCTION

Multipliers play an essential position these day's in digital signal processing and numerous other programs. With advances in technology, many researchers have kept in implementation and are looking to design multipliers which offer both of the following layout targets – high speed, low power intake, regularity of format and subsequently less place or even aggregate of them in one multiplier hence making them appropriate for diverse high speed, low power and compact VLSI implementation. Delay in output is one of the essential layout constraints in designing digital structures. Approximate computing (AC) is one of the procedures which may be used to reduce delay and/or growth the velocity. Since the computing end result might not be correct, AC can be exploited in errors-resilient applications. Examples of these programs include audio and photo processing, machine learning, and records mining. More specially, in many sign processing packages, a huge part of the strength intake is resulting from arithmetic operations (e.g., as much as almost seventy 5% of the total strength consumption of a quick Fourier transforms structure). Among these operations, multiplication, this is used time and again, is a high latency and strength ingesting operation. This makes approximate multipliers good candidates for being employed in mistakes-tolerant sign processing units. Generally, a multiplication operation consists of 3 steps. In step one, the partial products are generated based totally at the enter operands. In the second step, the partial merchandise is gathered until simplest

ISSN: 2278-4632
Vol-10 Issue-8 No. 1 August 2020

rows continue to be. In the final step, the remained two rows are summed with the aid of using a (fast) adder. One can also apply the approximation to each of these steps. Approximation can be invoked inside the first step to lower the number of partial products, or to lower the complexity of their technology. Approximation can be carried out in the second step of the multiplication procedure to lower the latency or electricity intake of the discount levels. One of those tactics is to make use of approximate compressors.

The latency and power intake of the multiplication operation are exceedingly tormented by the architecture of the adder used within the final step of the multiplication method. Hence, one might also hire an approximate adder in the very last step to enhance the energy intake of the multiplier. In this paper, proposing of an approximation approach for decreasing the number of partial products. In the proposed approximate set of rules, enter operands are truncated to h and t bits in line with the position of their leading one bit, where those truncated values are employed for the multiplication and addition operations. In addition, to lessen the error as a result of the truncation operation, we discover the approximate amount of the truncated values by rounding them. These simplifications bring about better accuracy and overall performance in comparison to those of the proposed approximate multipliers. Moreover, the proposed approximate multiplier has a nearly everyday blunders distribution with close to zero mean cost. The calculation core of the proposed multiplier performs multiplication and addition operations on truncated and rounded numbers and the results shifted to the left to generate the final output.

Because the mathematics operations are finished on the truncated values, the calculation middle of the proposed multiplier is small and consumes less power compared to that of the precise multiplier. Also, the accuracy of the proposed method is mainly depending on t and h parameter values and is not considerably suffering from the width of the enter operands. This gives a scalability feature for the proposed multiplier. Key contributions of this paper can be summarized as follows.

1)A new scheme for the scalable approximate multiplier, which finds the position of the leading one bit and exploits both truncation and rounding operations to improve the accuracy of the multiplication operation.

2) Exploration of t (truncation) and h (rounding) para- meters to find a tradeoff between accuracy, delay, and energy consumption.

3) Presenting hardware implementation of truncation- and rounding-based scalable approximate multiplier (TOSAM) for both signed and unsigned operations

4) Investigating design parameters of the proposed multiplier for image processing and classification applications.

II. LITERATURE SURVEY

In this section, we review some of the research efforts on designing approximate multipliers. In the dynamic segment method (DSM) structure [1], the input operands were truncated to *m* bits based on the position of their leading one bit where a fixed-width multiplication was performed on the truncated values. This way of truncation made the generated output always less than the exact one, making the mean relative error (MRE) negative.

This is an undesired feature due to the fact that for the approximate arithmetic units with the Gaussian error distribution, it is better to have the mean error close to zero for having a higher signal-to-noise ratio (SNR) when dealing with digital signal processing

ISSN: 2278-4632
Vol-10 Issue-8 No. 1 August 2020

applications [15]. In the dynamic range unbiased multiplier (DRUM) structure [6], to mitigate the error resulted from the truncation operation for pushing the MRE toward zero, the least significant bit of the truncated input was set to "1." In LETAM structure [5], the input operands were truncated and in the multiplication step, half of the partial products were omitted. Hence, the delay and power consumption were improved compared to those of the DSM and DRUM structures due to omitting the partial products. In RoBA multiplier [7], the input operands were rounded to the nearest power of two where the output was produced by some shift, add, and subtraction operations. In this structure, the number of elements that should be summed to generate final result was reduced compared to the exact multiplier leading to better energy and speed. As another approach, to improve the speed and area of the multiplier, the least significant bits of the partial products were eliminated [4].

A straightforward way to generate the partial products is to multiply each bit of the multiplier by the multiplicand, which can be performed simply by performing logical AND operation. Another approach is to encode the multiplier in higher radixes and multiply the encoded multiplier by the multiplicand. As the radix increases, encoding the multiplier becomes more complex. Hence, to decrease this complexity, one may use approximate encoders to generate partial products [6]. In [7], the partial products of an approximate radix-4 Booth multiplier were generated and accumulated approximately. Also, in [8], an approximate radix-8 Booth multiplier was proposed which used approximate adders to produce the least significant bits of the triple multiplicand. In [8], the most significant bits of the multiplier were encoded using exact radix-4 encoding and the least significant bits were encoded using an approximate higher radix encoding which rounded the least significant bits to the nearest power of two. In [9], four approximate 4:2 compressors were proposed and exploited in the reduction levels of the multiplier. In [10], an approximate 4:2 compressor was proposed and employed in the accumulation step and an error recovery module was added to improve the accuracy of multiplication.

In [10], several approximate 5:3 compressors were used in an approximate 15:4 compressor utilized in the main approximate multiplier structure. It should be mentioned that to increase the accuracy, accurate compressors were exploited to produce the most significant bits of the result. In, several approximate compressors have been proposed. Also, an algorithm was suggested to design efficient approximate multipliers composed of these compressors. In, several approximate adders were considered as the building blocks of the approximate multiplier and the design space was explored to find the optimum design. To improve the speed of multiplication, one approach is to change the numbering system to the logarithmic one to perform addition instead of multiplication. In this method, the logarithm of the input operands is generated, their sum is calculated, and an antilogarithm operation is performed on their sum to generate the final result. The complexity of this method originates from generating the logarithm and antilogarithm steps. The accuracy of the multiplier depends on the accuracy of these steps. Several studies have been conducted on how to find the logarithm and antilogarithm of a number. Mitchell proposed a simple approximate method to calculate the logarithm and antilog- arithm of a number and used it to generate the multiplication results (Mitchell multiplier). Since then, some studies have been conducted on improving the Mitchell-based logarithmic multipliers .In this paper, we propose an approximate multiplier that finds the position of the leading one bits of the input operands,

ISSN: 2278-4632
Vol-10 Issue-8 No. 1 August 2020

truncates and rounds them with different widths, and performs some shift, add, and small fixed-width multiplication operations to generate the multiplication result.

III. PROPOSED APPROXIMATE MULTIPLIER

A) TOSAM

Each positive integer number (N) can be represented as

$$N = \sum_{i=0}^{k} 2^i x_i \qquad (1)$$

Where k denotes the position of its leading one bit and xi is the i th bit of N. By factoring 2k from (1), it is rewritten as

$$N = 2^k \left(\sum_{i=0}^{k} 2^{i-k} x_i \right) = 2^k \times X \qquad (2)$$

where X is a fractional number between 1.0 and 2.0. Based on (2), the result of multiplying A by B may be calculated as

$$A \times B = 2^{k_A + k_B} \times X_A \times X_B. \qquad (3)$$

Widths of XA and XB are the same as A and B making the calculation of the exact value of XA × XB time and power consuming. We propose calculating the approximate amount of this term based on the fractional parts of XA and XB. In the remainder of this paper, we represent the fractional part of X as Y obtained from

$$Y = X - 1. \qquad (4)$$

For example, assume that X = (1.1101)2. In this case, Y = (0.1101)2. To generate the approximate value of Y, we divide this range (0.0–1.0) into S equal segments where S is a power of two represented by

$$S = 2^h \qquad (5)$$

Where h denotes an arbitrary positive integer which is one of our design parameters. It is obvious that the length of each segment is equal to 1/S. We propose to generate the approximate value of Y as

$$Y_{\mathrm{APX}} = \frac{2m-1}{2S} \quad \text{if } \frac{m-1}{S} \le Y < \frac{m}{S}, \quad m = 1, 2, \ldots, S. \qquad (6)$$

For a better illustration, the approximate amounts of Y for the case where S is equal to 4, is depicted in Fig. 1. To find YAPX, it is required to consider only h most significant bits of Y . For example, when S = 4(h =2), if two most significant bits of Y are zero, it means that 0 ≤ Y < 1/4.

Hence, we choose 1/8 = (0.001)2 as YAPX. When two most significant bits of Y are "10," which implies that 2/4 ≤ Y < 3/4, hence, YAPX is approximated as 5/8 = (0.101)2. In other words, the value of YAPX is obtained simply by truncating Y to h bits and inserting a "1" bit to the right side of the truncated Y . As a result, the width of YAPX will be equal to h + 1 bits. Making use of (4), (3) is rewritten as

ISSN: 2278-4632
Vol-10 Issue-8 No. 1 August 2020

$$A \times B = 2^{k_A+k_B} \times (1 + Y_A + Y_B + Y_A \times Y_B). \quad (7)$$

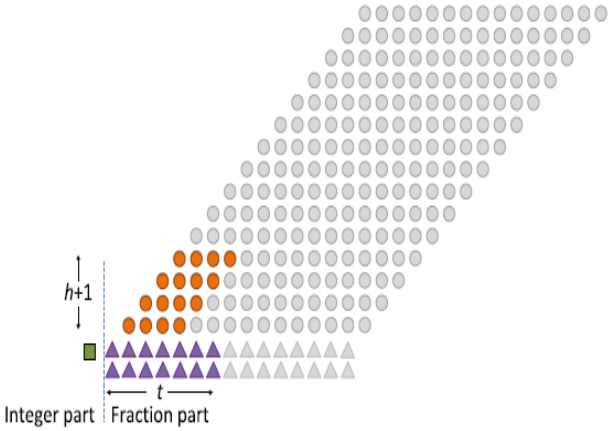

Fig.1.Dot diagram of term

1+ (YA) t + (YB) t + (YA) APX× (YB) APX where
t = 7 and h = 3.
Now, the approximate of (7) may be expressed as

$$A \times B \approx 2^{k_A+k_B} \times (1 + Y_A + Y_B + (Y_A)_{\text{APX}} \times (Y_B)_{\text{APX}}). \quad (8)$$

To improve the speed of calculation, we truncate YAand YB to t bits, where in the rest of this paper, we denote by (YA) tand (YB) t. Hence, we modify (8) as

$$A \times B \approx (A \times B)_{\text{APX}} = 2^{k_A+k_B} \times \begin{pmatrix} 1 + (Y_A)_t + (Y_B)_t + \\ (Y_A)_{\text{APX}} \times (Y_B)_{\text{APX}} \end{pmatrix} \quad (9)$$

Where the width of (YA) APX ((YB)APX) is h +1 bits. To have a better understanding, the dot diagram of the proposed algorithm for the case where t = 7 and h = 3 compared to that of an exact 16-bit multiplier is depicted in Fig. 1. The green square shows the "1" bit in the term 1+ (YA)t +(YB)t + (YA)APX ×(YB)APX. Orange circles denote partial products of (YA) APX× (YB) APX, whereas purple triangles show the bits of (YA) t and (YB) t. Gray circles and triangles are omitted and are not considered in the calculations. As shown in Fig. 1, in the exact 16-bit multiplier, the number of partial products is equal to 256, which must be summed to generate the final result while in the proposed method, only 31 of the partial products are kept This reduction rate will rise as the bit length of the multiplier input operands increases. As an example, the steps of multiplying A by B for the case of t = 7 and h = 3 are depicted in Fig. 2. In the rest of this paper, we denote our proposed structures by TOSAM (X, Y) where X and Y correspond to h and t.

The accuracy of the proposed approach depends on the values of the parameters t and h. Therefore, in the error analysis section (Section V), we will find a relationship between t and h parameters to achieve an almost high accuracy while having an acceptable speed and energy consumption. Finally, the proposed multiplication approach is feasible for the case of unsigned operands. To use it for signed multipliers, cone may find the absolute value of the input operands, multiply them by the proposed algorithm, and fix the sign of the final result according

ISSN: 2278-4632
Vol-10 Issue-8 No. 1 August 2020

to the sign of the input operands. Finding the exact absolute value of the input operands may degrade the speed of calculation and, hence, we produce it according to the method presented in.

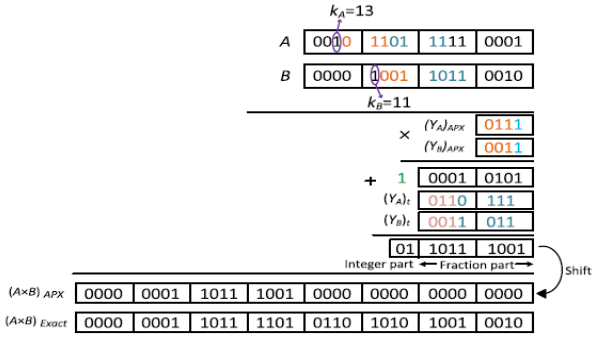

Fig. 2. Numeric example of 16-bit TOSAM (3, 7) with A = 11761 and B = 2482.

The approximate result [(A × B) APX] is equal to 28 901 376 while the exact result [(A × B) Exact] is equal to 29 190 802. In this case, the absolute error is 289 426 which is about 0.99% of the exact output (the error is less than 1% in this case).

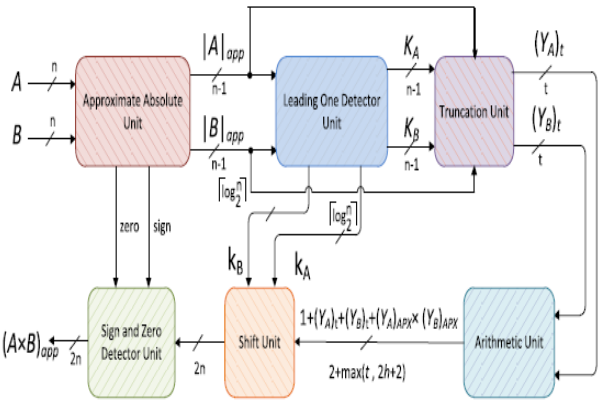

Fig. 3. Block diagram of the proposed approximate signed multiplier.

B) HARDWARE IMPLEMENTATION

The block diagram of the proposed signed approximate multiplier is depicted in Fig. 3. First, the approximate absolute value of the input operands (|A| app, |B| app) is determined using the Approximate Absolute Unit, similar to the one exploited in . In this unit, the bits of the input are inverted if the input is negative and they are not changed if the input is positive. |A| app and |B |app are injected to the Leading-One Detector Unit and the positions of their leading one bits are found using

$$K[i] = \left(\bigwedge_{j=i+1}^{n-2} \overline{I[j]}\right) \wedge I[i] \text{ for } 0 \le i \le n-2 \qquad (10)$$

Where I can be either |A| app or |B| app. Only one bit of the signal K is "1" revealing the position of the input leading one bit. By using the KA and KB signals in a lookup table, kA and kB signals needed for (7) can be generated. The schematic of the Leading-One Detector Unit for 8-bit input operands is depicted in Fig. 4. For example, assume that |A |app = (011001)2, in this case KA = (010000)2 and kA = (100)2 = 4. Signals |A| app, |B| app, KA, and KB are then applied to the Truncation Unit to produce (YA)t and (YB)t . Assume that the input and output of this unit are I and (Y) t.

ISSN: 2278-4632
Vol-10 Issue-8 No. 1 August 2020

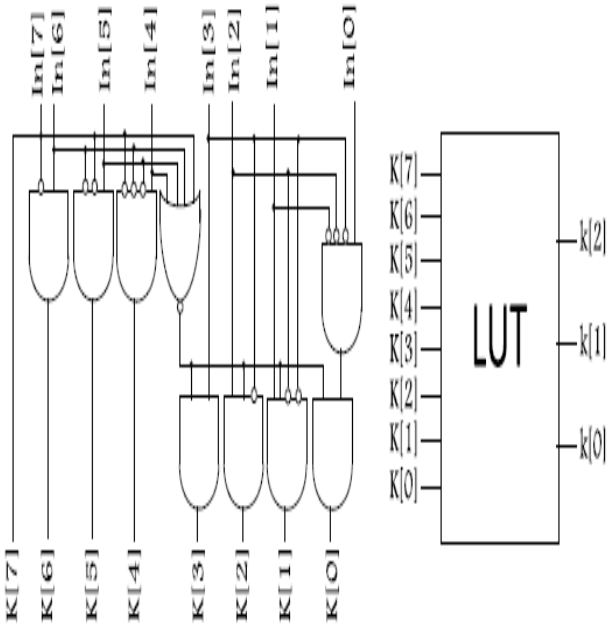

Fig. 4. Schematic of the Leading-One Detector Unit for 8-bit input operands.

In this case, the i th bit of the output can be generated using

$$(Y)_t[i] = \bigvee_{j=0}^{n-2} (K[j] \wedge I[j+i-t]) \text{ for } i < t. \qquad (11)$$

Signals (YA)t and (YB)t are then exerted to the Arithmetic Unit to calculate the term 1 + (YA)t + (YB)t + (YA)APX × (YB)APX. It should be noted that the h most significant bits of (YA) APX and (YB) APX are the same as the h most significant bits of (YA)tand (YB)t whose rightmost bits are always "1." Hence, there is no need to add extra hardware to generate (YA) APX and (YB) APX signals which are produced by simple wiring.

In the Shift Unit, the output of the Arithmetic Unit is shifted to left by kA +kB to produce the term 2kA+kB ×(1+(YA)t + (YB)t + (YA)APX × (YB)APX) [see (9)]. In the Sign and Zero Detector Unit, the sign of the output is set according to the sign of the multiplier input operands and also the output is set to zero if at least one of the inputs is zero. In the case of the unsigned multipliers, the Approximate Absolute Unit should be omitted and the Sign and Zero Detector Unit should be replaced by a Zero Detector Unit.

TOSAM can be implemented in an accuracy configurable structure. In order to implement an accuracy configurable structure of TOSAM, all of its units should be designed for the largest desired h and t values such that the design can work in all operation modes. We suggest a configurable TOSAM structure with three different operating modes of T2, T6, and T9 corresponding to TOSAM (0, 2), TOSAM (2, 6), and TOSAM (5, 9), respectively. The Truncation and the Shift Units of the configurable TOSAM should be designed for the largest t and h values (h = 5 and t = 9 in this case). In the Arithmetic Unit, some of the adders and logical AND gates should be power gated based on the operating mode to make the design more power efficient. The reduction levels of the partial products based on the operating modes are depicted in Fig. 5. In the last level, a fast 9-bit adder is employed. To decrease its switching activity, some of its inputs are set to "0" with a transmission gate (TG), based on the operating mode.

ISSN: 2278-4632
Vol-10 Issue-8 No. 1 August 2020

In the T2 mode, only the purple partial products are accumulated, only the purple adders are active (not power gated), and all of the inputs of the 9-bit adder are set to "0." The 10 least significant bits of the result are set to "0" using the

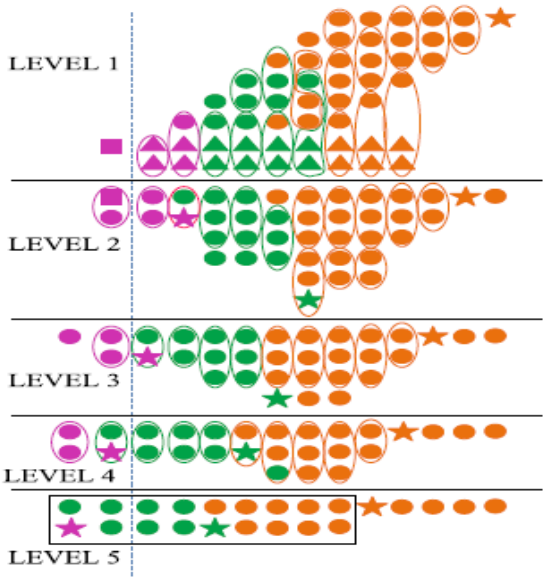

Fig. 5. Reduction levels of accuracy configurable TOSAM with three different operating modes.

TGs and purple stars are passed through the TGs to generate four most significant bits of the output. In the T6 mode, only the green and purple partial products are generated and summed to compose the final output and the orange adders are power gated. In addition, the orange inputs of the 9-bit adder are set to "0." Also, in the eighth column of LEVEL1, there are two orange circles that should be set to "0" by TGs when operating in the T6 mode. In this mode, the six least significant bits of the result are set to "0," green stars are passed through TGs to generate four intermediate bits of the output, and the four most significant bits of the result are produced by the 9-bit adder.

In the T9 mode, all parts are active. The orange stars are passed through the TGs to generate the five least significant bits of the result and the other bits are produced by the 9-bit adder. Note that the least significant bits of (Y A) APX and (Y B) APX, which depend on the operating mode, should be rounded (set to "1"). It is simply done by performing a logical OR operation on the corresponding bit and the operating mode. For example, when T6 signal is "1," the logical OR operation sets the corresponding bit of (Y A) APX and (Y B) APX to "1."

IV. RESULTS

ISSN: 2278-4632
Vol-10 Issue-8 No. 1 August 2020

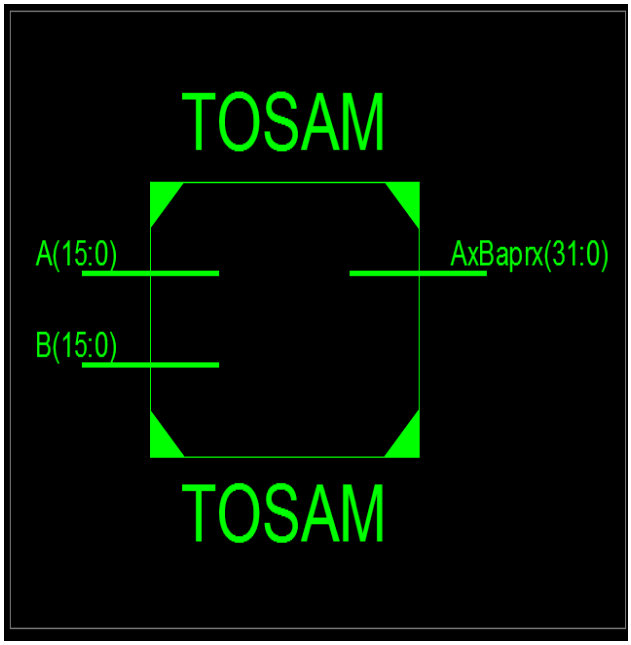

Fig 6: RTL schematic of TOSAM

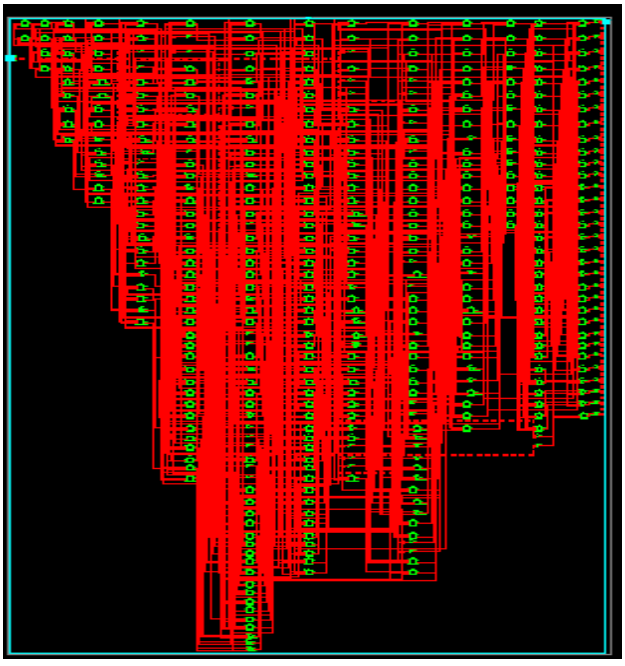

Fig 7: view technology schematic of TOSAM

ISSN: 2278-4632
Vol-10 Issue-8 No. 1 August 2020

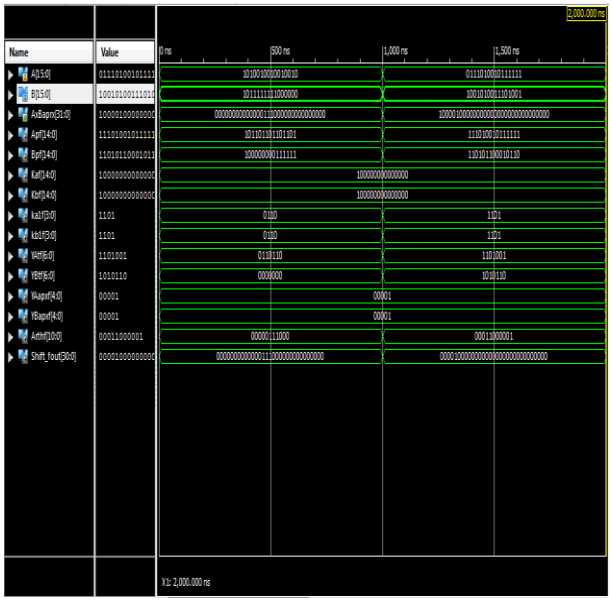

Fig 8: simulated waveforms of TOSAM

Table 1: parameter comparison table

Parameter	Existed design	Proposed design
Delay (ns)	45.141	29.189

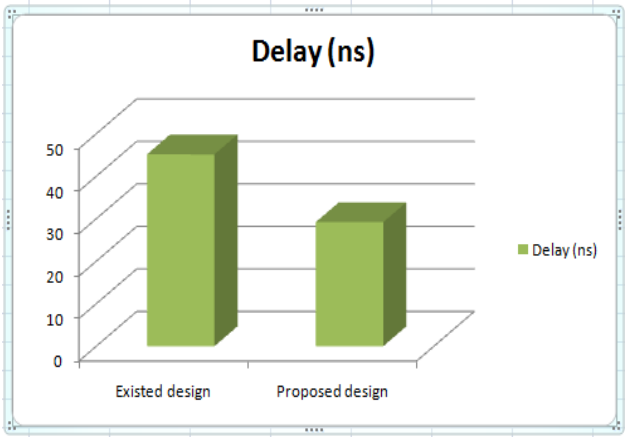

Fig9: delay comparison bar graph

V. CONCLUSION

ISSN: 2278-4632
Vol-10 Issue-8 No. 1 August 2020

In this paper, we suggested a high speed approximate multiplier in which the input operands were truncated with two different lengths, *t* and *h*,. The proposed multiplier was scalable and outperformed other approximate multipliers in terms of speed,. The proposed 32-bit multiplier has meet the requirement of speed .the delay is reduced from 45.141 to 29.189 when implementation of the proposed design is considered. Almost 39% of delay is reduced to previous method. Hence the proposed design is developed and simulated using XILINX ISE, with Verilog HDL language. Also, the high accuracy of the proposed multiplier made is a good choice to be exploited in image processing and classification applications.

REFERENCES

[1] S. Narayanamoorthy, H. A. Moghaddam, Z. Liu, T. Park, and N. S. Kim, "Energy-efficient approximate multiplication for digital signal processing and classification applications," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 6, pp. 1180–1184, Jun. 2015.
[2] S. Xu and B. C. Schafer, "Exposing approximate computing optimizations at different levels: From behavioral to gate-level," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 11, pp. 3077–3088, Nov. 2017.
[3] A. Raghu Nathan and K. Roy, "Approximate computing: Energy efficient computing with good-enough results," in *Proc. IEEE 19th Int. On-Line Test. Symp. (IOLTS)*, Chania, Greece, Jul. 2013, p. 258.
[4] D. Jeon, "Energy-efficient digital signal processing hardware design," Ph.D. dissertation, Dept. Elect. Eng., Michigan Univ., Ann Arbor, MI, USA, 2014.
[5] S. Vahdat, M. Kamal, A. Afzali-Kusha, and M. Pedram, "LETAM: A low energy truncation-based approximate multiplier," *Comput. Elect. Eng.*, vol. 63, pp. 1–17, Oct. 2017.
[6] S. Hashemi, R. I. Bahar, and S. Reda, "DRUM: A dynamic range unbiased multiplier for approximate applications," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Austin, TX, USA, Nov. 2015, pp. 418–425.
[7] R. Zendegani, M. Kamal, M. Bahadori, A. Afzali-Kusha, and M. Pedram, "RoBa multiplier: A rounding-based approximate multiplier for high-speed yet energy-efficient digital signal processing," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 2, pp. 393–401, Feb. 2017.
[8] V. Leon, G. Zervakis, D. Soudris, and K. Pekmestzi, "Approximate hybrid high radix encoding for energy-efficient inexact multipliers," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 3, pp. 421–430, Mar. 2018.
[9] O. Akbari, M. Kamal, A. Afzali-Kusha, and M. Pedram, "Dual-quality 4:2 compressors for utilizing in dynamic accuracy configurable multipliers," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 4, pp. 1352–1361, Apr. 2017.
[10] M. Ha and S. Lee, "Multipliers with approximate 4–2 compressors and error recovery modules," *IEEE Embedded Syst. Lett.*, vol. 10, no. 1, pp. 6–9, Mar. 2018.