LOW POWER HIGH ACCURACY APPROXIMATE MULTIPLER USING HIGH ORDER COMPRESSOR

[1]Dr. SIVA SANKARA PHANI TAMMIREDDY,[2]Gudimetla Harika, [3]Pilla Sairam,[4]Chandana Hari, [5]Kopella Venkata Naga Abhinayak

[1]Professor & HOD, Dept of E.C.E, BVCITS, Batlapalem, Amalapuram, AP
[2,3,4,5]B. Tech, Dept of E.C.E, BVCITS, Batlapalem, Amalapuram, AP

Submitted: 11-02-2026 *Accepted: 18-03-2026* *Published: 24-03-2026*

ABTRACT Consumption of Energy is the major factor, in the various processing application like DSP, ASIC, and FPGA. This project presents power optimized more accurate and approximate 8 x 8 multiplication architecture design. To achieve high accura,4cy, this concept utilizes accurate (i.e., exact) 4:2 compressors in the higher significance weights. To reduce power consumption, high-order approximate compressors are used in the middle significance weights. As an enhancement of this concept, latency and density optimized modified round square architecture is proposed with more accuracy. The approach is to round the operands to the nearest exponent of two. This way the computational intensive part of the multiplication is omitted improving speed and energy consumption at the price of a small error. The proposed approach is applicable to both signed and unsigned multiplication.

Keywords: Digital signal processing, Arithmetic Logic Unit, multiplier and accumulate unit, elliptic curve cryptography, Approximate Multiplier.

INTRODUCTION Energy minimization is major requirements in almost any electronic systems, especially the portable ones such as smart phones, tablets, and different gadgets. It is extremely desired to attain this minimization with minimal performance (speed) penalty [1]. Digital signal processing (DSP) blocks are most wanted in transportable components for realizing various multimedia applications. The computational core of these blocks is the ALU where the multiplications and additions are the major part [6]. The multiplications plays foremost operation in the processing elements which can leads to high consumption of energy and power. Many of the DSP cores implement image and video processing algorithms where final outputs are either images or videos prepared for human consumptions. It facilitates to go for approximations for improving the speed and energy in the arithmetic circuits. This originates from the limited perceptual abilities in observing an image or a video for human beings. In addition to the image and video processing applications, there are other areas where the exactness of the arithmetic operations is not critical to the functionality of the system (see [2],[3]). Approximate computing provides an accuracy, speed and power/energy consumption. The advantage of approximate multiplier reduces the

error rate and gain high speed. For correcting the division error compare operation and a memory look up is required for the each operand is required which increases the time delay for entire multiplication process [4].

LITERATURE SURVEY A traditional method to reduce the aging effects is overdesign which includes techniques like guard-banding ad gate oversizing. This approach can be area and power inefficient [8]. To avoid this problem, an NBTI- aware technology mapping technique was proposed in [7] which guarantee the performance of the circuit during its lifetime. Another technique was an NBTI- aware sleep transistor in [3] which improve the lifetime stability of the power gated circuits under considerations. A joint logic restructuring and pin reordering method in [6] is based on detecting functional symmetries and transistor stacking effects. This approach is an NBTI optimization method that considered path sensitization. Dynamic voltage scaling and bogy-biasing techniques were proposed in [4] and [5] to reduce power or extend circuit life. These techniques require circuit modification or do not provide optimization of specific circuits.

EXISTING TECHNIQUE: The critical path of a multiplier is often related to the maximum height of PPM (partial product matrix). Thus, there is a need to compress the PPM. A n:2 compressor is a slice of a multiplier that reduces n numbers (i.e., product terms) to two numbers when properly replicated. In slice i of the multiplier, the n:2 compressor receives n bits in position i and one or more carry bits from the lower positions (such as i-1), and produces two output bits in positions i and i+1 and one or more carry bits into the higher positions. Conventionally, 4:2 compressors are used in the multiplier design [1,2]. Fig. 1 (a) gives the block diagram of an accurate (i.e., exact) 4:2 compressor. The four input bits are denoted as X0, X1, X2 and X3. The two output bits in positions i and i+1 are denoted to as Sum and Carry respectively. The carry bit from the lower position is denoted as Cin while the carry bit into the higher position is denoted as Cout. Fig. 1 (b) gives the block diagram of an approximate 4:2 compressor. To save the logic of carry chains, the carry bits Cin and Cout are omitted. Moreover, in [1,2], to reduce the error rate, the logics of Sum and Carry are re-designed (i.e., different from the logics of Sum and Carry in an accurate 4:2 compressor). Previous works [1,2] did not consider high-order compression (i.e., did not consider $n \geq 5$). In fact, high-order compression can further reduce the delay and power.

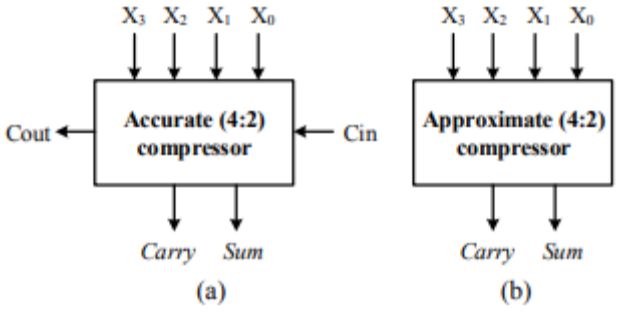

Figure 1. (a) Accurate 4:2 compressor (b) Approximate 4:2 compressor.

THE APPROXIMATION OF CARRY: Here, we study the approximation of the logic of the Carry output. In a conventional half adder, the carry bit Ch is defined as below: Ch (X0,X1) = X0. X1 (1) In a conventional full adder, the carry bit Cf is defined as below: Cf (X0,X1,X2) = X0. X1 + X1. X2 + X0 . X2 (2) As described in [6], we can implement the equation (1) as a modified half adder, and implement the equation (2) as a modified full adder. Fig. 2 (a) and Fig. 2 (b) give the logic of modified half adder and the logic of modified full adder, respectively. Then, based on the modified half adder and the modified full adder, we can construct the approximation logic for the Carry output of a high-order approximate compressor. In the following, we use the Carry output of our approximate 5:2 compressor examples. When the number of input bits is 5 (i.e., n = 5), we can split the 5 input bits into 2 groups: one group includes X0, X1, and X2, and the other group includes X3 and X4. Then, the Carry output of our approximate 5:2 compressor is as below: Cf (X0,X1,X2) +Ch (X3,X4) + Ch (X0+X1+X2,X3+X4). Fig. 3 displays the logic of the Carry output of our approximate 5:2 compressor. When the number of input bits is 8 (i.e., n = 8), we can split the 8 input bits into 3 groups: one group includes X0, X1, and X2, one group includes X3, X4, and X5, and one group includes X6 and X7. Then, the Carry output of our approximate 8:2 compressor is as below: Cf (X0,X1,X2) + Cf (X3,X4,X5) + Ch(X6,X7) + Cf(X0+X1+X2, X3+X4+X5, X6+X7).

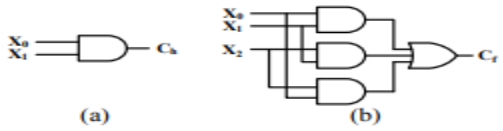

Figure 2. (a) Modified half adder (b) Modified full adder.

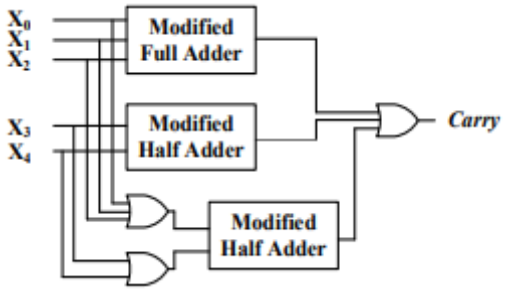

Figure 3. The logic of Carry output of our approximate 5:2 compressor.

THE APPROXIMATION OF SUM Here, we study the approximation of the logic of Sum output. Conventionally, the tree of XOR gates are used to produce the output Sum. However, compared with other logic gates, XOR gate often has larger design overheads. We use the logic gates in SAED 32nm cell

library as an example. Table I tabulates the comparisons among OR gate, NOR gate, XNOR gate, and XOR gate. From Table I, we find that XOR gate has the largest power, the largest area, and the largest delay. Thus, if we can replace XOR gates with other logic gates, all the design overheads (including the power, the area, and the delay) can be reduced.

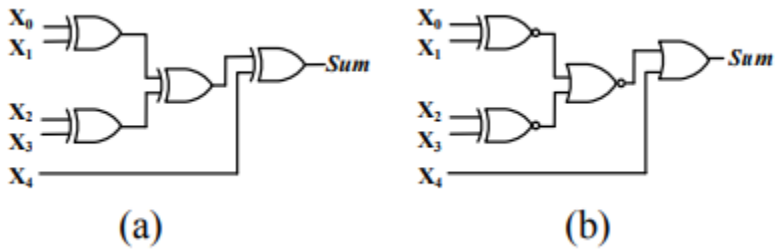

Figure 4. Sum of 5:2 compressor. (a) Accurate (b) Our approximate

We construct the approximation logic (a tree of logic gates) for the Sum output of a high-order approximate compressor as below. At the first level, we use XNOR gate instead of XOR gate. Note that the output of XNOR gate is the inverse of the output of XOR gate. To compensate for the error rate, we use NOR gate at the second level and OR gate at the third level. Since all XOR gates are replaced by other logic gates, the design overheads are greatly saved. we use the Sum output of 5:2 compressor (Fig. 4) as examples. Fig 4 (a) gives the logic of the Sum output of an accurate 5:2 compressor. Fig 4 (b) gives the logic of the Sum output of our approximate 5:2 compressor.

IMPLEMNETATION:

Typically, a multiplier consists of three parts. In the first part, AND gates are utilized to generate partial products. In the second part, the maximum height of PPM (partial product matrix) is reduced by using a carry save adder tree. In the third part, a carry propagation adder is used to produce the final result. The design complexity of a multiplier is primarily related to the PPM reduction circuitry (i.e., the multiplier is primarily related to the PPM reduction circuitry (i.e., the second part). Thus, the study of multiplier design [1-6] focuses on the optimization of the PPM reduction circuitry. In this section, we propose an approximate 8 x 8 multiplier design. Fig. 5 gives the overall structure of our PPM reduction circuitry. According to the significance, the weights are classified into three categories: the higher significance weights, the middle significance weights, and the lower significance weights. Note that the designers are allowed to configure the number of higher significance weights, the number of middle significance weights and the number of lower significance weights for the trade-off between the power consumption and the computational accuracy. To reduce the power consumption with a small error, our PPM reduction circuitry applies the significance driven logic compression technique as below: the higher significance weights use accurate (i.e., exact) 4:2 compressors; the middle significance weights use our approximate high-order compressors the lower significance weights use inaccurate compressors (OR-tree based

approximation). Our PPM reduction circuitry has two stages. The first stage is for all the weights. The second stage is only for the higher significance weights. After the second stage is completed, each weight has at most two product terms.

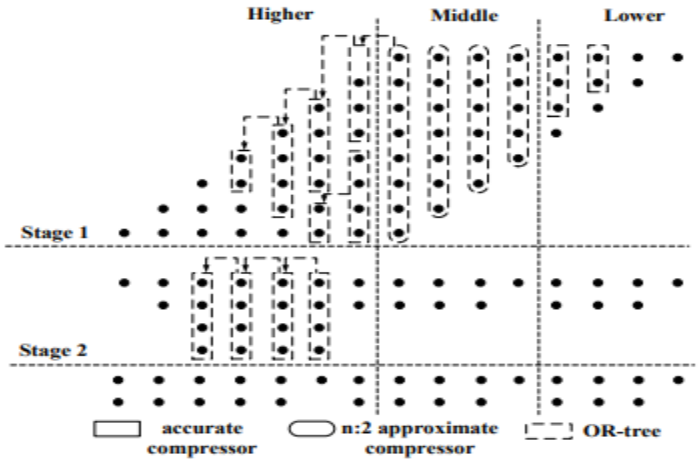

Figure 5. The PPM reduction in the proposed approximate multiplier.

PROPOSED TECHNIQUE:

MODIFIED ROUND SQUARE ARCHITECTURE

The main concept of conventional rounding based approximate multiplier [1] is selecting the rounded values for both the inputs which are in form of 2n and both the inputs should be in the form of 3x2p-1 (p is considered as arbitrary positive integer value which is greater than 1) in this case of the conventional approach the final value obtained by the multiplier would be less or more than the exact result obtained. Depending on the Ar (rounded input value of A) and Br (rounded input value of B) respectively and the result obtained is inaccurate. The motive behind this approximate multiplier is to make use of the ease of operation of power n (2n) .To elaborate on the process of the approximate multiplier, first, let us denote of the input of A and B rounded value by Ar and Br, respectively. The multiplication of A by B can be write as A x B = (Ar - A) x (Br – B) + Ar x B + Br x A – Ar x Br ----1 Key observation is to facilitate the multiplications of Ar * Br, Ar * B, and Br X A may be implemented just by the operation of shifting which is publicized in the eqn (1). The hardware implementation (Ar – A) X (Br – B), however, is rather complex. The weight of this term in the concluding result, depends on differences of the exact numbers from their rounded ones, is typically small. Hence, it is proposed to omit this part from (Ar -A) x (Br – B), helping simplify the multiplication operation shown in the eqn (2). Hence, to perform the multiplication process, the following expression is used A x B = Ar x B + Br x A – Ar x Br ----2 While both values lead to same effect on the accuracy of the multiplier, selecting the larger one (expect for the value p=2) leads to a smaller hardware implementation for determining the nearest rounded value. It

originates from the detail that the number in the composition of 3 × 2 p−2 considered as do not care in the both rounding process up and down manner, and smaller logic expressions may be achieved. With the help of accurate and approximate equation the proposed architecture can be designed. Fig 1 provides the detail block diagram for the MRSA multiplier which is applicable for the two processing such as unsigned multiplication, signed multiplication If the operation is for unsigned multiplication the sign detector and sign set is disabled which can speed up the multiplication process. The two inputs are provided to the detector block which detects MSB of the input and it is provided to the sign set block to denoted signed or unsigned multiplication. Rounding and shifter are worn to reduce the operands value to the nearest power of 2 and it can be shifted with the help of barrel shifter. There are 3 levels of shifter for the following terms obtained in the approximate equation. The kongee stone adder is used to add the two functions from the shifter. The sign can be set with the help detector block. If the output is negative the error value is calculated by inverting the output equation and it is added with binary value of 1. It supposed to be noted that contrary to the previous work where the approximate result is lesser than the exact result, the final result calculated by the MRSA multiplier may be either larger or lesser than the exact result depending on the magnitudes of Ar and Br compared with those of A and B , respectively. Note that if one of the operands (say A) is lesser than its equivalent rounded value while the other operand (say B) is larger than its equivalent rounded value, then the approximate result will be larger than the exact result. Because the term (A r − A) × (B r − B) will be neglected. Since the differentiation between (1) and (2) is precisely this product, the approximate result becomes higher than the exact one. Similarly, if both A and B are larger or both are lesser than Ar and Br, then the approximate result is lesser than the exact result. Hence, before the multiplication operation starts, the values of both input are absolutes and the output sign of the result are based on the inputs signs be determined and then the operation be performed for unsigned numbers and, at the last stage, the proper sign be applied to the unsigned result.

STRUCTURE LEVEL DESIGN OF MRSA MULTIPLIER: From the equation 1 and 2 the structure level implementation of the multiplier were designed. The inputs are represented in the format of two's complement. First, the signs of the inputs are determined, and for each negative value, the unconditional value is generated. Next, the rounding block extracts the nearest value for each unconditional value in the form of 2n. The bit width of the output of this block is n (the most significant bit of the absolute value of an n-bit number is zero for two's complement format). To determine the nearest value of input A, the operands are rounding off to the power of 2 with the help of rounding criteria.

There are four cases for selecting final rounded of value from the original input values there are discussed below 1. Ar is high and Br is low. 2. Ar is low and Br is high. 3. Ar is high and Br is high. 4. Ar is low

and Br is low. By selecting the case one, the approximate result is larger when observed with exact The error rate is the important factors that should be considered while designing the approximate multiplier. The distance between exact and inexact results for the approximate multiplier is calculated before calculating the error rate of the rounding based approximate multiplier. The hardware architectures of the sign detector, rounding, barrel shifter, kongee stone, subtractor and the sign set modules. The RTL architecture for MRSA multiplier is shown in Fig 2 taken by cadence encounter tool 180-nm technology. The sign set block is used to negate the output if the final output is negative valued. To negate values, which have the representation of two's complement, the corresponding circuit based on X+ 1 should be used. To speed up negation operation, one may skip the incrementation process in the negating phase by accepting its associated error. As result. From the case two and three, the approximate result is somewhat larger than the accurate result in contrast with case one. For case four, the approximate result is lower than the exact result. The program should be slightly modified for each one of the cases. The rate or error is extremely low down for case one and four in contrast with other two cases. will be seen later, the impact on the error decreases when an input width increases. If the negation is performed exactly (approximately), the implementation is called signed MRSA (S-MRSA) multiplier [approximate S-MRSA (AS-MRSA) multiplier]. If the inputs are always positive, to speed up and decrease the power consumption, the sign detector and sign set blocks are omitted from the architecture, providing us with the architecture called unsigned MRSA (UMRSA) multiplier

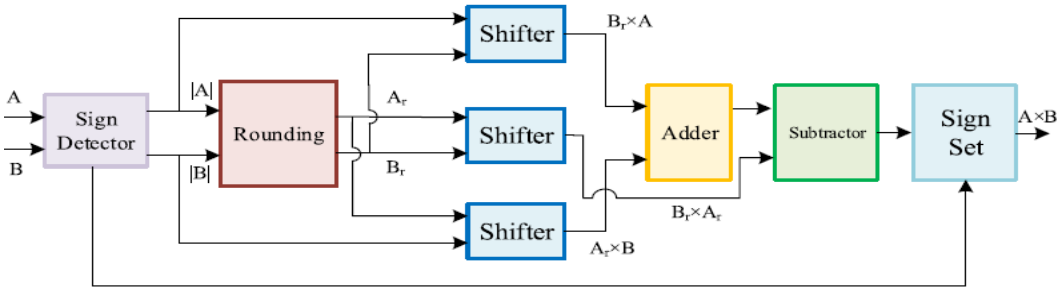

Fig.6 Block diagram for the hardware implementation of the proposed multiplier.

provide the block diagram for the hardware implementation of the proposed multiplier in Fig where the inputs are represented in two's complement format. First, the signs of the inputs are determined, and for each negative value, the absolute value is generated. Next, the rounding block extracts the nearest value for each absolute value in the form of $2n$. It should be noted that the bit width of the output of this block is n (the most significant bit of the absolute value of an n-bit number in the two's complement format is zero). To find the nearest value of input A, we use the following equation to determine each output bit of the rounding block:

$$
\begin{aligned}
A_r[n-1] &= \overline{A[n-1]} \cdot A[n-2] \cdot A[n-3] \\
&\quad + A[n-1] \cdot \overline{A[n-2]} \\
A_r[n-2] &= (\overline{A[n-2]} \cdot A[n-3] \cdot A[n-4] \\
&\quad + A[n-2] \cdot \overline{A[n-3]}) \cdot \overline{A[n-1]} \\
&\vdots \\
A_r[i] &= (\overline{A[i]} \cdot A[i-1] \cdot A[i-2] + A[i] \cdot \overline{A[i-1]}) \cdot \prod_{i=i+1}^{n-1} \overline{A[i]} \\
&\vdots \\
A_r[3] &= (\overline{A[3]} \cdot A[2] \cdot A[1] + A[3] \cdot \overline{A[2]}) \cdot \prod_{i=4}^{n-1} \overline{A[i]} \\
A_r[2] &= A[2] \cdot \overline{A[1]} \cdot \prod_{i=3}^{n-1} \overline{A[i]} \\
A_r[1] &= A[1] \cdot \prod_{i=2}^{n-1} \overline{A[i]} \\
A_r[0] &= A[0] \cdot \prod_{i=1}^{n-1} \overline{A[i]}.
\end{aligned}
$$

In the proposed equation, Ar [i] is one in two cases. In the first case, A[i] is one and all the bits on its left side are zero while A[i − 1] is zero. In the second case, when A[i] and all its left-side bits are zero, A[i − 1] and A[i − 2] are both one. Having determined the rounding values, using three barrel shifter blocks, the products $Ar \times Br$, $Ar \times B$, and $Br \times A$ are calculated. Hence, the amount of shifting is determined based on logAr 2− 1 (or logBr 2 − 1*)* in the case of A (or B) operand. Here, the input bit width of the shifter blocks is n, while their outputs are 2n. A single 2n-bit Kogge-Stone adder is used to calculate the summation of $Ar \times B$ and $Br \times A$. The output of this adder and the result of $Ar \times Br$ are the inputs of the *subtractor* block whose output is the absolute value of the output of the proposed multiplier. Because Ar and Br are in the form of 2n, the inputs of the *subtractor* may take one of the three input patterns. where P is $Ar \times B + Br \times A$ and Z is $Ar \times Br$. The corresponding circuit for implementing this expression is smaller and faster than the conventional subtraction circuit. Finally, if the sign of the final multiplication result should be negative, the output of the subtractor will be negated in the *sign set* block. To negate values, which have the two's complement representation, the corresponding circuit based on $\bar{X}+1$ should be used. To increase the speed of negation operation, one may skip the incrementation process in the negating phase by accepting its associated error. As will be seen later, the significance of the error decreases as the input widths increases. In this paper, if the negation is performed exactly (approximately), the implementation is called signed MRSA (S-MRSA) multiplier [approximate S-MRSA (AS-MRSA) multiplier]. In the case where the inputs are always positive, to increase the speed and reduce the power consumption, the *sign detector* and *sign set* blocks are omitted from the architecture, providing us with the architecture called unsigned MRSA (U-MRSA) multiplier. In this case,

the output width of the *rounding* block is n +1 where this bit is determined based on $Ar\,[n] = A[n-1] \cdot A[n-2]$. This is because in the case of unsigned $11x \ldots x$ (where x denotes do not care) with the bit width of n, its rounding value is 10…0 with the bit width of $n + 1$. Therefore, the input bit width of the shifters is $n + 1$. However, because the maximum amount of shifting is $n - 1$, $2n$ is considered for the output bit width of the shifters.

RESULT:

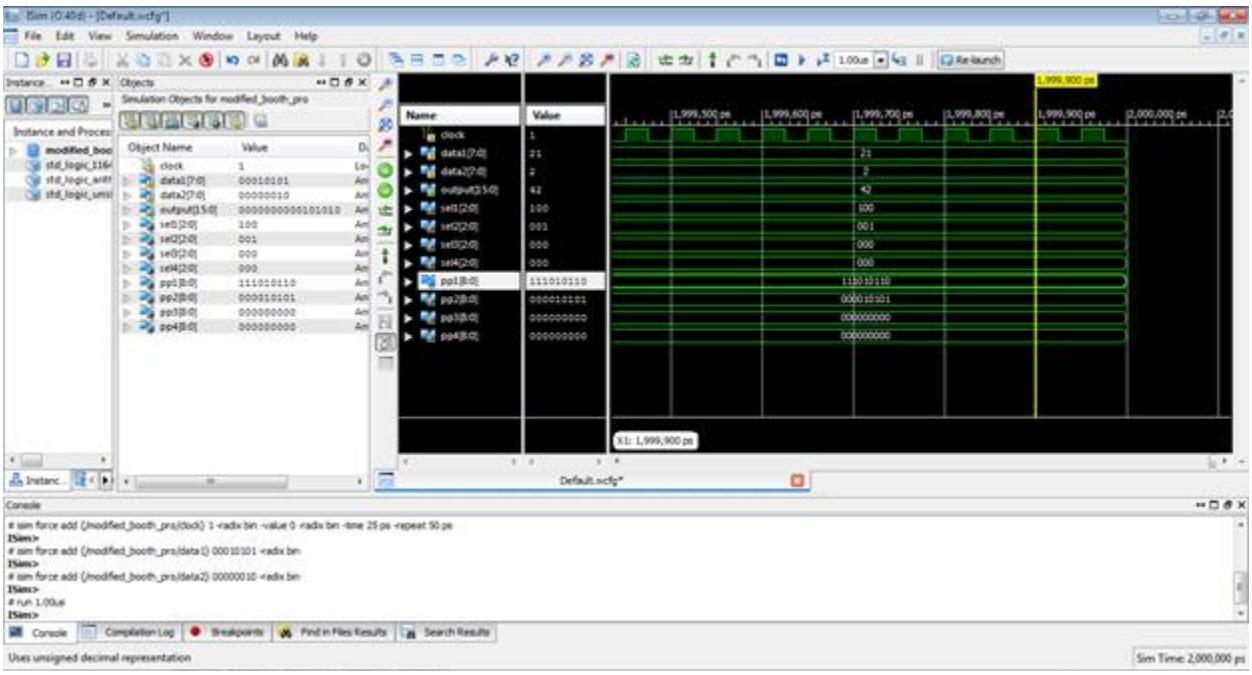

Fig7: Existing simulation result captured in XILINX ISE14.7

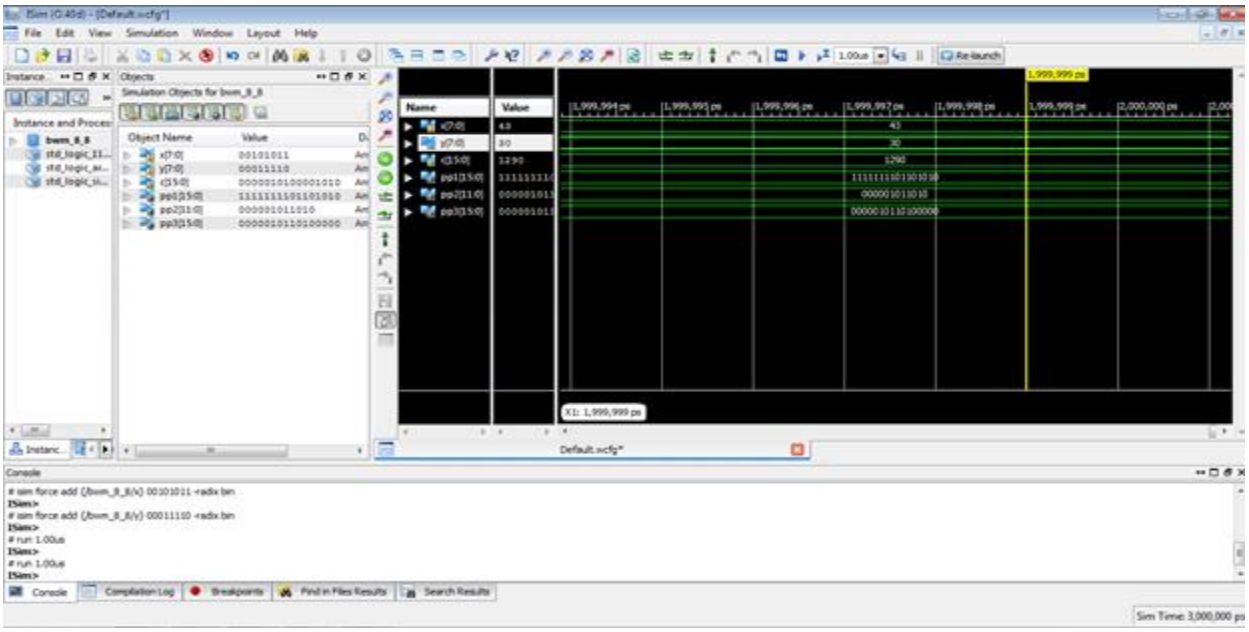

Fig8: Proposed simulation result captured in XILINX ISE14.7

CONCLUSION High-speed and energy efficient approximate multiplier were proposed. The MRSA multiplier had a high accuracy depend upon the 2n input form. The high exhaustive computation part is neglected to provide high performance. So hardware structural design is designed for S-MRSA, MRSA and AS-MRSA multiplier. The efficiencies of the MRSA multiplier were compared with some existing accurate and approximate multipliers with different parameters. With the help of comparison table, MRSA multiplier provides the better area, power, and energy efficient when compared with some already proposed accurate and approximate multiplier.

FUTURE SCOPE: Further, this project is enhanced by modifying partial product addition structures with advanced adders like CSA, CSLA, CLA, to decrease parameters.

REFERENCE:

[1] Wen-Chang Yeh and Chein-Wei Jen, "High-speed Booth encoded parallel multiplier design," IEEE Trans. on Computers, vol. 49, isseu 7, pp. 692-701, July 2000.

[2] Jung-Yup Kang and Jean-Luc Gaudiot, "A simple high-speed multiplier design," IEEE Trans. on Computers, vol. 55, issue 10, Oct. pp. 1253-1258, 2006.

[3] Shiann-Rong Kuang, Jiun-Ping Wang and Cang-Yuan Guo, "Modified Booth multipliers with a regular partial product array," IEEE Trans. onCircuit and Systems, vol.56, Issue 5, pp. 404-408, May 2009.

[4] Li-rong Wang, Shyh-Jye Jou and Chung-Len Lee, "A well-structured modified Booth multiplier design," Proc. of IEEE VLSI-DAT, pp. 85-88, April 2008.

[5] A. A. Khatibzadeh, K. Raahemifar and M. Ahmadi, "A 1.8V 1.1GHz Novel Digital Multiplier," Proc. of IEEE CCECE, pp. 686-689, May 2005.

[6] S. Hus, V. Venkatraman, S. Mathew, H. Kaul, M. Anders, S. Dighe, W. Burleson and R. Krishnamurthy, "A 2GHZ 13.6mW 12x9b mutiplier for energy efficient FFT accelerators," Proc. of IEEE ESSCIRC, pp. 199-202, Sept. 2005.

[7] Hwang-Cherng Chow and I-Chyn Wey, "A 3.3V 1GHz high speed pipelined Booth multiplier," Proc. of IEEE ISCAS, vol. 1, pp. 457-460, May 2002.

[8] M. Aguirre-Hernandez and M. Linarse-Aranda, "Energy-efficient high-speed CMOS pipelined multiplier," Proc. of IEEE CCE, pp. 460-464, Nov. 2008.

[9] Yung-chin Liang, Ching-ji Huang and Wei-bin Yang, "A 320-MHz 8bit x 8bit pipelined multiplier in ultra-low supply voltage," Proc. of IEEE A-SSCC, pp. 73-76, Nov. 2008.

[10] S. B. Tatapudi and J. G. Delgado-Frias, "Designing pipelined systems with a clock period approaching pipeline register delay," Proc. of IEEE MWSCAS, vol. 1, pp. 871-874, Aug. 2005.

[11] A. D. Booth, "A signed binary multiplication technique," Quarterly J. Mechanical and Applied Math, vol. 4, pp.236-240, 1951.

[12] M. D. Ercegovac and T. Lang, *Digital Arithmetic*, Morgan Kaufmann Publishers, Los Altos, CA 94022, USA, 2003.

[13] C. S. Wallace, "A suggestion for a fast multiplier," IEEE Trans. On Computers, vol. BC13, pp. 14-17, Feb. 1964.

[14] M.D. Ercegovac et al., ªFast Multiplication without Carry- Propagate Addition,º IEEE Trans. Computers, vol. 39, no. 11, Nov. 1990.

[15] R.K. Kolagotla et al., ªVLSI Implementation of a 200-Mhz 16 _ 16 Left-to-Right Carry-Free Multiplier in 0.35_m CMOS Technology for Next-Generation DSPs,º Proc. IEEE 1997 Custom Integrated Circuits Conf., pp. 469-472, 1997.

[16] P.F. Stelling and V.G. Oklobdzija, ªOptimal Designs for Multipliers and Multiply-Accumulators,º Proc. 15th IMACS WorldCongress Scientific Computation, Modeling, and Applied Math., A. Sydow, ed., pp. 739-744, Aug. 1997.

[17] Passport 0.35 micron, 3.3 volt, Optimum Silicon SC Library, CB35OS142, Avant! Corporation, Mar. 1998.

[18] G. Goto et al., ªA 4.1ns compact 54 _ 54-b Multiplier UtilizingSign-Select Booth Encoders,º IEEE J. Solid-State Circuits, vol. 32, no. 11, pp. 1,676-1,682, Nov. 1997.

[19] G. Goto et al., ªA 54 _ 54-b Regularly Structured Tree Multiplier,º IEEE J. Solid-State Circuits, vol. 27, no. 9, Sept. 1992.

[20] R. Fried, ªMinimizing Energy Dissipation in High-Speed Multipliers, º Proc. 1997 Int'l Symp. Low Power Electronics and Design, pp. 214-219, 1997.

[21] N.H.E. Weste and K. Eshraghian, Principles of CMOS VLSI Design: A Systems Perspective, second ed., chapter 8, p. 520. Addison Wesley, 1993.

[22] J. Fadavi-Ardekani, ªM_N Booth Encoded Multiplier Generator Using Optimized Wallace Trees,º IEEE Trans. VLSI Systems, vol. 1, no. 2, June 1993.

[23] A.A. Farooqui et al., ªGeneral Data-Path Organization of a MAC Unit for VLSI Implementation of DSP Processors,º Proc. 1998 IEEE Int'l Symp. Circuits and Systems, vol. 2, pp. 260-263, 1998.

[24] K. Hwang, Computer Arithmetic: Principles, Architecture, and Design, chapter 3, p. 81. John Wiley & Sons, 1976.

[25] K.H. Cheng et al., ªThe Improvement of Conditional Sum Adder for Low Power Applications,º Proc. 11th Ann. IEEE Int'l ASICConf., pp. 131-134, 1998.