Scopus

DOI

Google Scholar

IMPLEMENTATION OF LOW POWER AND HIGH ACCURACY 2D FIR FILTER USING APPROXIMATE MULTIPLIER FOR IMAGE PROCESSING APPLICATIONS

Merfeena[1], Shanubha Mari Jero[2], Dyana Christilda[3]

[1]Student, Department of Electronics and Communication Engineering, Vins Christian College of Engineering, Tamil Nadu, India.
[2]Student, Department of Electronics and Communication Engineering, Vins Christian College of Engineering, Tamil Nadu, India.
[3]Professor, Department of Electronics and Communication Engineering, Vins Christian College of Engineering, Tamil Nadu, India.

ABSTRACT

The multiplier is the most important arithmetic functional unit in many applications, and since these applications frequently call for numerous multiplications, they consume a lot of power. The multipliers in these gadgets are the biggest consumers of electricity. Speed, area, and power are the three key factors that influence the multiplier's performance. If the speed increases, a lot of space is consumed, and the opposite is also true. In this study, we provide a roughly multiplier with excellent accuracy and minimal power. The accuracy of the approximation multiplier is a design parameter for an error-tolerant multiplier (ETM). Compared to existing multipliers, it utilises less space and electricity. When compared to Vedic multiplier, Wallace tree multiplier, and booth multiplier, experimental results demonstrate that the suggested approximation multiplier uses less electricity on average. In this paper, we provide an accurate, error-tolerant multiplier. The suggested approximate multiplier is simulated, synthesised, and targeted on a virtex 6 FPGA device in order to study its performance.
Keywords: Approximate multiplier, Real time application, Error tolerant.

INTRODUCTION

One of the most potent DSP tools for executing various tasks like filter construction, convolution, FFT, and circular convolution is the multiplier. FIR filter design is one of the most often utilised applications in DSP. In communication, signal and image processing, and embedded ASICs, multipliers are crucial components. In many applications, it serves as the most crucial and basic arithmetical function unit. The process of developing multipliers involves a number of processes, and as a result, they take up more space in the hardware, require more energy, and have an impact on performance. Artificial intelligence, image recognition, and digital signal processing are the applications. These applications require a lot of multiplications, which uses a lot of electricity. Implementing this type of application is difficult due to its high-power consumption, especially on mobile devices. Numerous studies have suggested methods for lowering a multiplier circuit's power usage. The approximation multiplier is one method used to lower the power

consumption of a multiplier. The approximation multiplier trades off accuracy in favour of reduced cell size, timing lag, and increased power consumption.

Two categories can be used to describe the approximate multiplier. The first form involves managing the multiplier's temporal path. By utilising dynamic voltage scaling, it is accomplished. When a low voltage is provided to the multiplier, the critical route will take longer to complete. As a result, when the time path is violated, the error occurs and produces approximations of the intended results. The second type is utilised to redesign the precise multiplier circuits by altering the multipliers' operational characteristics. The Wallace tree multiplier, Vedic multiplier, and booth multiplier are a few examples of multipliers. The majority of the previously suggested designs for rebuilding multipliers are inaccurate compressors with m inputs and n outputs.

One of the digital filters that is frequently used in digital signal processing applications in numerous industries, including imaging, instrumentation, communications, etc. is the finite impulse response (FIR) filter. The FIR filter may be implemented using programmable digital signal processors (PDSPs). However, in order to implement a large-order filter, numerous intricate calculations are required, which has an impact on the speed, cost, flexibility, etc. of standard digital signal processors. Any processor's performance will only be influenced by its power and latency. To get a CPU that works well, the power and delay in any processor should be reduced. The multiplier architecture is the one that is utilised the most in CPUs. Only the efficient processor is produced if the multiplier's power consumption and latency are decreased.

LITERATURE REVIEW

The approximate compressor-based multiplier for images proposed by Swathi Krishna T U *et al*. is 15-4. He employed a 15-4 approximation compressor that was created using the Xilinx ISE design suite. In applications for image processing, multiplying operations can be carried out with less latency, less power, and less space. Compared to accurate multipliers, approximate compressor-based multipliers offer better circuit performance with the drawbacks of error rate and average normalised error distance values. But in applications like image processing, it is reasonable.

A brand-new approximate multiplier design for digital signal processing is put out by Yue Zhao1 *et al.* He presented a new OR and AND gate-based approximate 4-to-2 compressor in this manner. The space and power consumption can be significantly reduced with the proposed approximative multipliers. Further testing of the proposed approximate multipliers is done in edge detection and image sharpening applications. The PSNR and SSIM findings demonstrate the applicability of our suggested approximate multipliers for image processing.

In this paper, Ihsen Alouani1 *et al.* offers a new architecture that uses accuracy as a design criterion and implements a rough parallel multiplier utilising heterogeneous blocks. While improving performance and power trade-offs, the proposed heterogeneous multiplier outperforms the tested circuits in terms of output precision. To increase power efficiency and performance, approximate computing relies on the range of accepted inaccuracy in the calculation process.

RoBA multiplier, an approximation-based multiplier proposed by Reza Zendegani *et al.* is intended for high-speed yet energy-efficient digital signal processing. Three hardware implementations of the approximation multiplier—one for unsigned operations and two for signed operations—are the technique used. Increase efficiency and speed at the expense of a slight inaccuracy.

Sana Mazahir *et al.* provides a probabilistic error analysis of a recursive approximate multiplier. The error probability analysis for recursive approximate multipliers with approximate partial products is employed. We obtain precise error performance evaluation.

Kenta Shirane *et al*. recommends designing approximate multipliers with maximal error-awareness. The method employed is a methodology for designing a sequence of approximate array multipliers with varying accuracy, area, power, and delay. The results reveal that our developed multipliers outperform existing approximation multipliers in terms of accuracy-area efficiency.

Mukesh Kumar Sukla *et al*. proposes a low-power, low-area approximate multiplier with decreased partial products. This work's method proposes a power and area optimised architecture for an error-tolerant multiplier. By applying approximation at the lower level of partial product, a trade-off with the error characteristic is made. The increase in area and power is accomplished with a probability error (PE) of 50%. Up to 957. 854MHz, the crucial frequency is reached.

Sunghyun Kim *et al.* introduces an approximate multiplier with good performance and low energy consumption for error-tolerant applications. It is proposed that an approximate binary multiplier be employed for high-performance and energy-efficient architecture with acceptable error characteristics. The simulation results show that the suggested approximate multiplier can produce considerable improvements in all performance measures while maintaining low error metrics.

Weiqiang Liu *et al.* suggests designing and testing approximation logarithmic multipliers for low power error-tolerant applications. This suggested iterative ALMs (IALMs) method employs a setone adder in both mantissa adders during an iteration, as well as lower-part-or adders and approximate mirror adders for the final addition. The normalised mean error distance of 16-bit approximation LMs is reduced by up to 18% when compared to traditional LMs with exact units, and the power-delay product is reduced by up to 37%.

Pabithra.S *et al.* offer an analysis of an approximation multiplier utilising a 15-4 compressor for error tolerant applications. The 15-4 compressor was built using an approximate 5-3 compressor approach. Because it consumes less power, about 5-3 compressors are employed in four distinct ways in 15-4 compressor. When compared to the actual multiplier with a tolerable error rate, approximate multipliers produced better outcomes in terms of power, area, and speed. Because the projected multiplier's latency was inversely tied to its speed, it also provided lower latency than the actual one.

EXISTING MULTIPLIERS

MULTIPLIERS

The execution time of multiplication operations is crucial in arithmetic operations, and it is a major factor in a design's performance evaluation. As a result, efficient multipliers for speed, power consumption, and area are required. These multipliers are appropriate for a wide range of high-speed, low-power, and compact VLSI applications. A multiplier is a key basic building component in the design of systems that use digital signal processing and other applications. Many academics are always attempting to build multipliers with low power consumption, high speed, and regular structure, so that they take up less space for compact VLSI implementation. Many algorithms have been presented in the past to do multiplication. Each algorithm has its own set of advantages and disadvantages, as measured by their speed, power consumption, and circuit complexity.

VEDIC MULTIPLIER

Multipliers are essential components of digital systems and play a key role in digital design. Vedic mathematics is a system of mathematical rules that allows for more efficient speed application. Urdhva Tiryakbhyam Sutras and Nikhilam Sutras are utilised in Vedic multipliers. In our project, the Urdhva Tiryagbhyam Sutra is taken into account in the Vedic multiplication method. Urdhva Tiryagbhyam serves as the fundamental sutra for quick and easy multiplication procedures. The input data is divided into two equal pieces, followed by the cross and vertical product.

VEDIC SERIES OF MULTIPLIERS

Vedic mathematics is an ancient mathematical system that existed in India. Basic arithmetic methods are powerful, straightforward, and logical in this approach. Another advantage is its consistency. These benefits make Vedic mathematics an essential research area. The rules of Vedic mathematics are primarily based on sixteen sutras. Urdhva Triyakbhyam sutras and Nikhilam sutras are employed for multiplication among these sixteen sutras.

When compared to conventional multipliers, Vedic multipliers are considered to be among the best, and Urdhva Triyakbhyam sutra-based multiplication is more efficient than Nikhilam sutra-based multiplication. Because of its regularity and simplicity, Vedic mathematics is simple to implement on FPGA. All partial products required for multiplication are calculated far in advance of the actual multiplication. This is a significant advantage of this multiplication. Based on a Vedic mathematics technique, these partial products are joined to generate the final product, resulting in an extremely fast approach. Multiplier designs based on Vedic mathematics are fast and require little power.

Multipliers are fundamental and essential components of a digital signal processor. Multiplication is a critical step in increasing the processing speed of digital signal processors. Multiplier blocks are used in convolution, Fast Fourier

transforms, and other transforms. Urdhva Tiryagbhyam is one of the most efficient multiplication procedures in Vedic mathematics.

MULTIPLICATION USING VEDIC MULTIPLIER

Urdhva Tiryagbhyami is a universal multiplication formula that can be applied to every circumstance of multiplication. When the Vedic multiplication method is used, the input data is divided into two equal pieces, followed by the cross and vertical product. The graphic below depicts the 4multiplication process utilising Vedic method. Each square block represents a multiplier of two. The first 22 multiplier's inputs are A1A0 and B1B0, as indicated. The leftmost block is a 22 multiplier with A3A2 and B3B2 inputs. The middle two 22multipliers' inputs are A3A2 and B1B0, and A1A0 and B3B2.
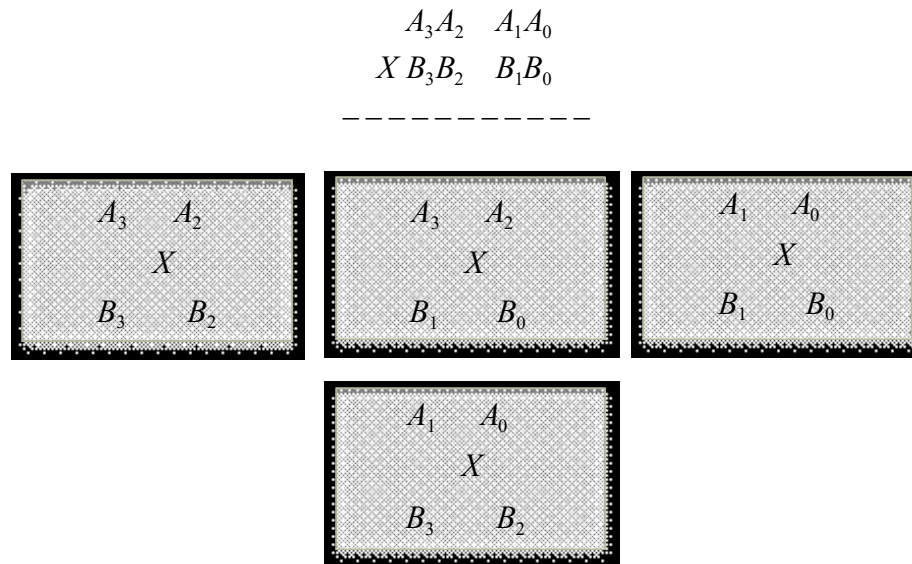
$$A_3A_2 \quad A_1A_0$$
$$X\,B_3B_2 \quad B_1B_0$$
$$- - - - - - - - - -$$

Fig 1. 4×4 multiplication operation using Vedic technique

S33 S32 S31 S30 is produced by multiplying A3A2 and B3B2.
S23 S22 S21 S20 is produced by multiplying A3A2 and B1B0.
S13 S12 S11 S10 is obtained by multiplying A1A0 and B3B2.
S03 S02 S01 S00 is obtained by multiplying A1A0 and B1B0.
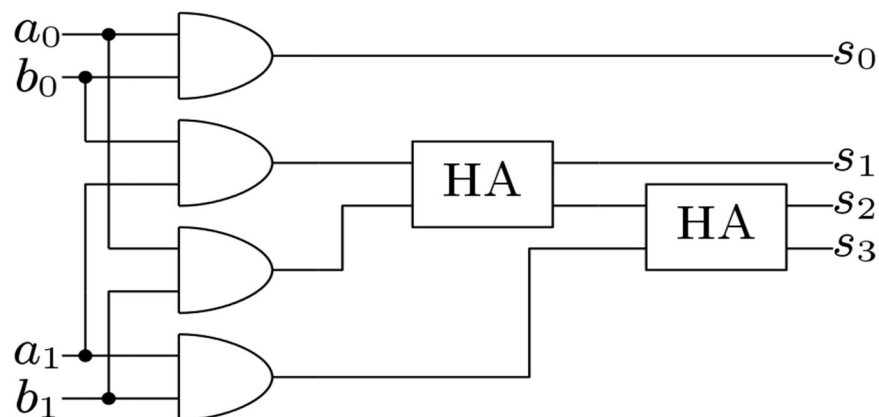All partial products computed by these four blocks are added together.

Fig 2. 2 bit Vedic multiplier

BOOTH MULTIPLIER

The Booth multiplication method is important in developing signed multipliers that use multiplier encoders and reduce the number of intermediate products. Multiplier is the fundamental and essential component of an arithmetic unit in high-performance tasks such as digital signal processing (DSP), digital image processing (DIP), and high-performance central processing unit (CPU).

Using a booth encoder to limit the number of partial products created during the multiplication process is a practical and efficient technique of enhancing the multiplier's speed. When using the booth encoding approach, less additions are required than when using the traditional multiplication rule. The Booth Encoder (BE), the Booth Selector (BS), and the adder tree summation are the three essential components of a conventional Booth Multiplier. The carry look ahead adder is employed in this case. It has been discovered that the net architecture generated by the adder and multiplier optimises speed and area by reducing the number of partial products necessary and lowering the required power consumption.

MULTIPLICATION USING BOOTH MULTIPLIER

We consider the bits of '1' in the multiplier, then perform the multiplication operation for those subsequent multiplicand bits, and then display these bits. If the bit is '0,' we obtain zero for the multiplicand bits, which are then displayed in the iteration steps. As a result, multiplication is a lengthy procedure because each bit is multiplied, and the multiplication action takes longer. As a result, regular multiplication requires more time for the multiplication operation where each and every bit is multiplied, resulting in a greater number of iteration steps. The latency increases as the number of iteration steps increases. So, we use a booth multiplier to reduce the amount of iteration steps and downsides. In multiplication, the basic booth multiplier is used for both signed and unsigned bits. In the instance of this multiplier shifting procedure, it is done directly in certain cases and indirectly in others. So it is a sophisticated procedure in which each bit is examined and then shifting occurs in multiplication.
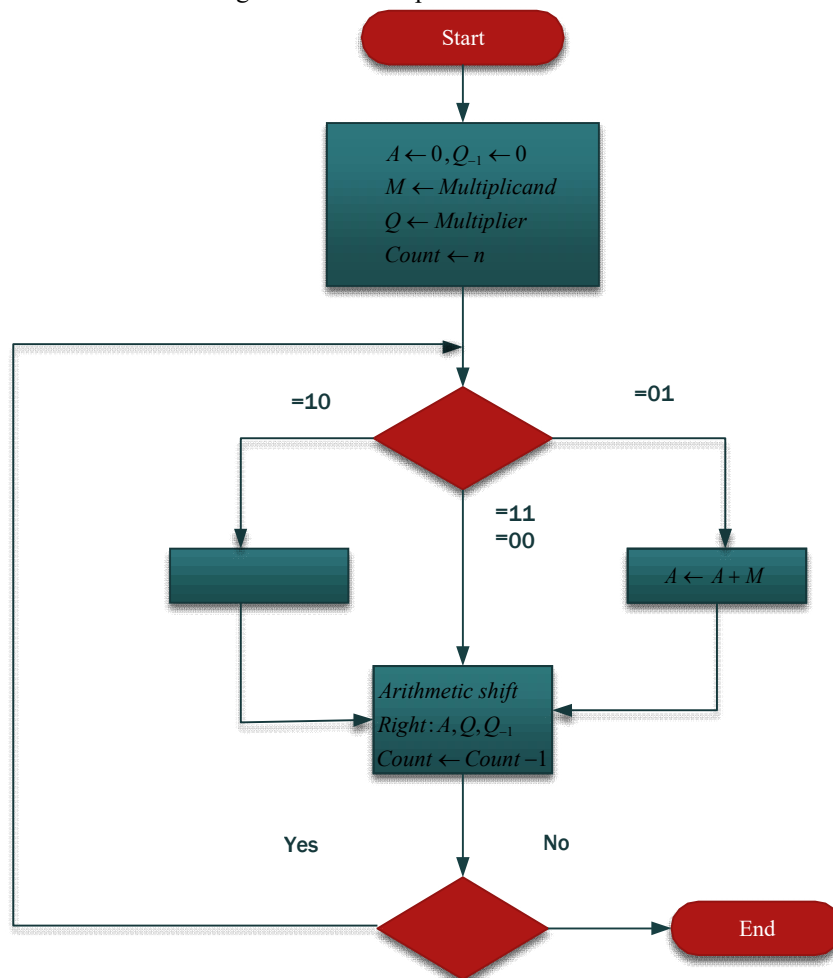

Fig 3. Block diagram of Booth multiplier

The final result comes when the sequence counter approaches zero. As a result, it is considered one of the slowest processes to continue and takes a long time to process; as a result, the delay also increases and gradually leads to the increment of the delay, which then reduces the speed of the multiplier. As a result, electricity consumption rises. We choose the adjusted booth multiplier after evaluating all of the disadvantages. While performing the multiplication with the booth multiplier, the number of iteration steps will be lowered.

ARCHITECTURE OF BOOTH MULTIPLIER
Complement Generator, Booth Encoder, Partial Product, and Carry Save adder are the four components of the architecture.
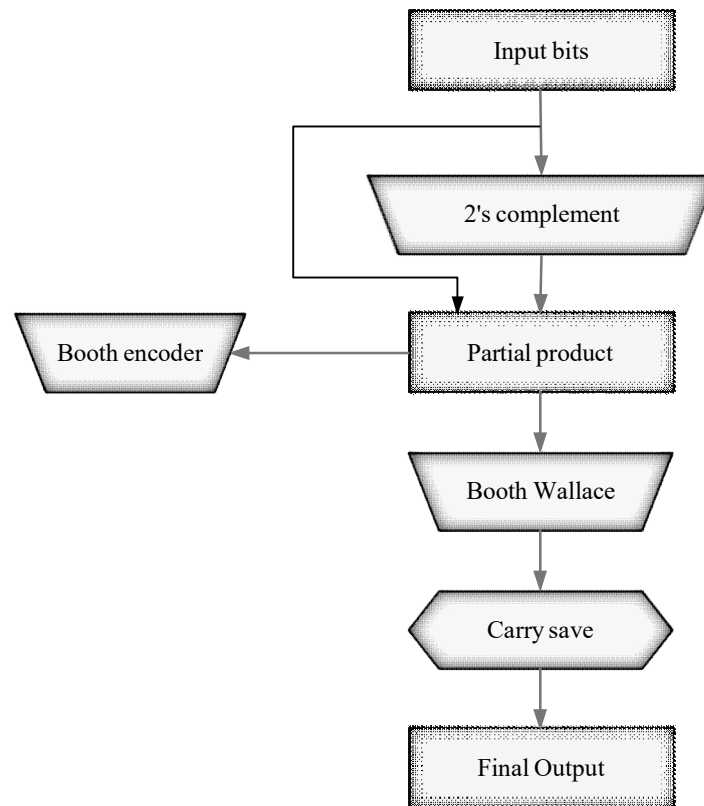

Fig 4. Flow chart of booth multiplier

A) COMPLEMENT COMPARATOR

In the case of this complement comparator, the multiplicand or the given data is used to generate the 2's complement, and the complemented results are obtained. These supplemented results are used when there is a requirement; otherwise, the direct result is used in some circumstances according to preset cases.

B) ENCODER

To perform the multiplication, the beginning bits are given to the encoder, and then the encoder's applied bits are treated as one bit for the two bits, and finally the multiplication is performed on the bits.

C) PARTIAL PRODUCT GENERATOR

The decoded bits are obtained at the partial product generator, resulting in the development of fewer partial products during number multiplication. As a result, the quantity of partial products is reduced.

D) WALLACE MULTIPLIER

The Wallace multiplier also functions as an array multiplier. This multiplier employs half adder and full adder adders. During the multiplication operation, every bit is multiplied by every other bit.

E) CARRY SAVE ADDER

Because the fast addition of partial products is performed and the result is obtained so quickly, this adder is preferred above other adders. When compared to regular multiplication, the Booth multiplier finds the operand that functions as multiplier and does multiplication for the algorithm since it reduces the number of steps while completing addition. In the case of multiplication, the operation is conducted for each bit of the multiplier with the multiplicand, and then the formation of partial products occurs in the appropriate order, followed by the addition of all partial products obtained.

The most intriguing aspect is that the additions made in this multiplication are data dependent, making this a perfect algorithm. Multiplication of signed numbers is not achievable in the same way that it is for unsigned numbers because signed numbers in 2's complement form cannot yield the exact result if the same multiplication technique is used for unsigned numbers. As a result, the booth algorithm is applied, which reduces the final result's sign. Thus, the booth method achieves high-speed multiplication and has applications such as digital signal processing and radar.

BOOTH ALGORITHM

  1. Adding a '0'bit to the LSB of the multiplier and considering from the right most of the multiplier to combine two bits from the right side to the left side and respective multiplier.
  2. 00:11: no operation is performed.
  3. 01: indicate the end of the string 1s before multiplying the partial products.
  4. 10: Begin the string of 1s by removing multiplicand from partial products.
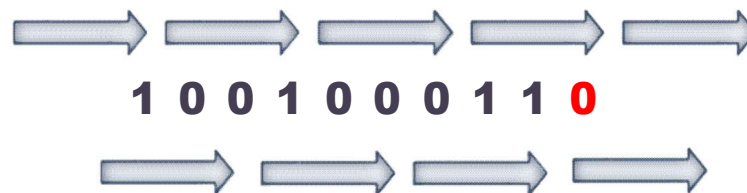

Fig 5. bit combining of booth recorder

The required method of obtaining results for recognising the highest speed multiplier is to enhance parallelism, which was effective in obtaining fewer number of subsequent calculation levels. Likewise, the booth algorithm for a radix-4 compares three bits using an overlapping technique. Because this multiplication can minimise the number of partial products by half when compared to standard multiplication.
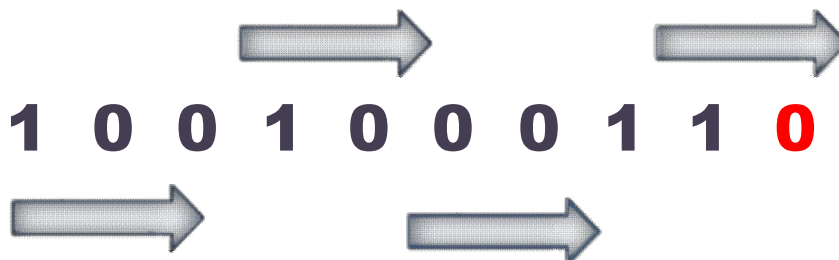

Fig 6. Bit pairing as per booth recorder

Thus, by using three bits, the speed of multiplication can be increased while the number of multiplication steps is cut in half compared to conventional multiplication. Booth multiplication also has several advantages, such as the fact that when three bits are the same, no operation can be performed, reducing the number of adders and the complexity of the multiplier. This multiplier has a distinct operation for successive bit operations and does not require addition and subtraction for each step of multiplication.

Also, the multiplication of signed numbers is not possible as same as unsigned numbers because the signed numbers in 2's complement form cannot give the exact result if the same process of multiplication is applied for unsigned numbers. That is why booth algorithm is used and deteriorates the sign of the final result. Thus, booth algorithm performs high speed multiplication and it find its way in different uses like digital signal processing, radar etc.

WALLACE MULTIPLIER

During 1965, computer scientist Luigi WALLACE designed the WALLACE hardware multiplier. WALLACE multiplier is a parallel multiplier that has been extracted. It is slightly faster and necessitates fewer gates. The parallel multiplier employs a variety of techniques. The WALLACE system is a parallel multiplier scheme that effectively minimises the number of adder stages required to complete partial product summing. Wallace multiplier is created by reducing the number of rows in the matrix number of bits at each summation stage using full and half adders. Despite the fact that WALLACE multiplication has a regular and less complex structure, the operation is slower owing to the serial multiplication method. Furthermore, the WALLACE multiplier is less expensive than the Wallace tree multiplier. As a result, the WALLACE multiplier is created and analysed by taking into account the many techniques of applying complete adders including distinct logic types (a) Wallace Multiplier Implementation (b) The WALLACE multiplier algorithm is based on a matrix structure. The matrix's partial product is produced in the first step by AND stages.

WALLACE TREE MULTIPLICATION

The Wallace tree is a lengthy multiplication variant. The first step is to multiply one factor's digits (per bit) by the other factor's digits. Each of these partial products has the same weight as the sum of its elements. The weighted sum of all these partial products yields the final product.

As previously stated, the first step is to multiply each bit of one number by each bit of the other, which is performed using a simple AND gate, yielding bits; the partial. The partial products of bits am by bn bits have weight $2^{(m+n)}$ in the second step.

The resulting bits are reduced to two numbers in the second stage, which is performed as follows: If there are three or more wires of the same weight, add the following layer: -

- Insert any three wires with the identical weights into a complete adder. As a result, for each of the three input wires, an output wire of the same weight and an output wire of greater weight will be produced.
- If you still have two wires of the same weight, put them in a half adder.
- Connect the last wire to the next layer if there is just one left.

The final step is to feed the two resulting numbers to an adder, which produces the final product.

Example:

n=4, multiplying $a_3a_2a_1a_0$ by $b_3b_2b_1b_0$

1. First, we multiply every bit by every bit:
   - weight 1 – a_0b_0
   - weight 2 – a_0b_1, a_1b_0
   - weight 4 – a_0b_2, a_1b_1, a_2b_0
   - weight 8 – a_0b_3, a_1b_2, a_2b_1, a_3b_0
   - weight 16 – a_1b_3, a_2b_2, a_3b_1
   - weight 32 – a_2b_3, a_3b_2
   - weight 64 – a_3b_3
2. Reduction layer 1:
   - Pass the only weight-1 wire through, output: 1 weight-1 wire
   - Add a half adder for weight 2, outputs: 1 weight-2 wire, 1 weight-4 wire
   - Add a full adder for weight 4, outputs: 1 weight-4 wire, 1 weight-8 wire
   - Add a full adder for weight 8, and pass the remaining wire through, outputs: 2 weight-8 wires, 1 weight-16 wire
   - Add a full adder for weight 16, outputs: 1 weight-16 wire, 1 weight-32 wire
   - Add a half adder for weight 32, outputs: 1 weight-32 wire, 1 weight-64 wire
   - Pass the only weight-64 wire through, output: 1 weight-64 wire

3. Wires at the output of reduction layer 1:
   - weight 1 – 1
   - weight 2 – 1
   - weight 4 – 2
   - weight 8 – 3
   - weight 16 – 2
   - weight 32 – 2
   - weight 64 – 2
4. Reduction layer 2:
   - Add a full adder for weight 8, and half adders for weights 4, 16, 32, 64
5. Outputs:
   - weight 1 – 1
   - weight 2 – 1
   - weight 4 – 1
   - weight 8 – 2
   - weight 16 – 2
   - weight 32 – 2
   - weight 64 – 2
   - weight 128 – 1
6. Group the wires into a pair of integers and an adder to add them.

STEPS INVOLVED IN WALLACE TREE MULTIPLIERS ALGORITHM

• Divide each bit of one of the arguments by each bit of the other, getting N outcomes. The wires carry varying weights depending on the position of the multiplied bits.
• Reduce the amount of partial products to two full adder layers. Divide the wires into two numbers and add them with a standard adder.

WALLACE TREE MULTIPLIER USING RIPPLE CARRY ADDER

Ripple Carry Adder is a way for doing a bigger number of additions with the carry ins and carry outs that will be linked. As a result, the ripple carry adder employs many adders. To add multiple-bit numbers, a logical circuit can be built utilising numerous complete adders. Cin, the Cout of the previous adder, is input by each full adder. This type of adder is known as a ripple carry adder because each carry bit "ripples" to the next complete adder. Take any three values with the same weights and feed them into a complete adder. As a result, the output wire will be the same weight.

• At the first stage, a partial product is formed after multiplication. The data is collected using three wires and added using adders, and the carry of each stage is combined with the next two data in the same stage.
• Using the same process, partial products are reduced to two layers of full adders.
• At the final stage, the same ripple carry adder mechanism is used, yielding product terms p1 to p8.
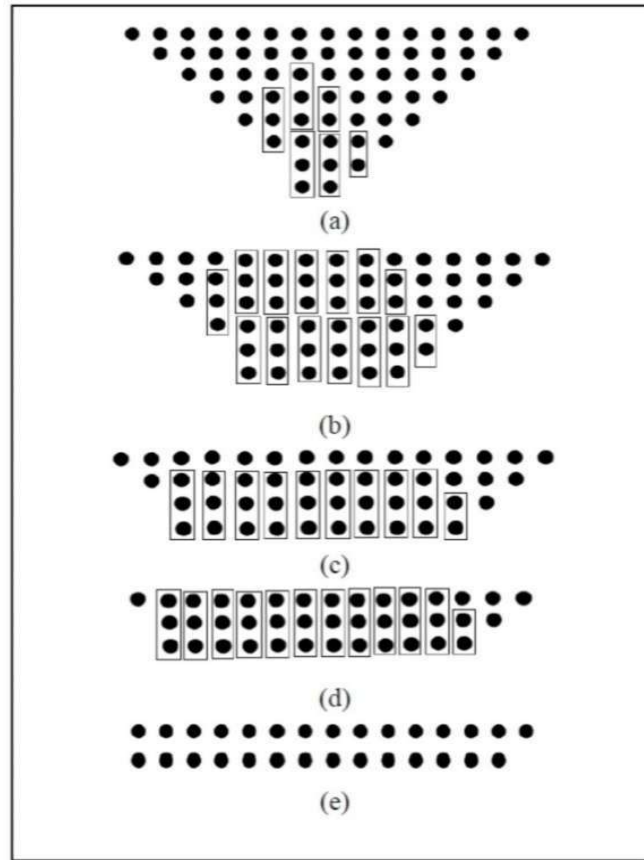
Fig 7. Column compression scheme for 8x8 Wallace multiplier

APPROXIMATE MULTIPLIER

Approximate circuits are considered as an error-tolerant applications that can tolerate some loss of accuracy with improved performance and energy efficiency. Multipliers are the key arithmetic circuits in many of such applications such as digital signal processing (DSP).

ROUNDING BASED APPROXIMATION

The main idea in the proposed approximate multiplier is to make use of the ease of operation when numbers are two to the power n (2n). To elaborate on the operation of the approximate multiplier, first, let us denote that the rounded numbers of the input of A and B by Ar and Br respectively. The multiplication of A by B is written as

$$A*B=(A_r - A)*(B_r - B) + A_r * B + B_r * A - A_r * B_r \qquad (1)$$

The key observation is that the multiplications of Ar*Br,A*Br and Ar *Br may be implemented just by the shift operation. The hardware implementation of (Ar-A) * (Br-B), however, is rather complex. The weight of this term in the final result, which depends on the differences of the exact numbers from their rounded ones, is typically small. Hence, we propose to omit this part from equation 1, helping simplify the multiplication operation. Hence, to perform the multiplication process, the following expression is being used

$$A*B = (A_r * B) + B_r * A - A_r * B_r \qquad (2)$$

Thus, one can perform the multiplication operation using three shift and two addition/subtraction operations. In this approach, the nearest values for A and B in the form of 2nshould be determined. When the value of A (or B) is equal to the 3* 2p−2 (where p is an arbitrary positive integer larger than one). It has two nearest values in the form of 2n with equal absolute differences that are 2Pand 2p−1. While both values lead to the same effect on the accuracy of the proposed multiplier, selecting the larger one (except for the case of p = 2) leads to a smaller hardware implementation for determining the nearest rounded value, and hence, it is considered in this paper. It originates from the fact that the

numbers in the form of 3* 2p−2 are considered as do not care in both rounding up and down simplifying the process, and smaller logic expressions may be achieved if they are used in the rounding up. zero).

 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*Br, A*Br and Ar*Br are calculated. A single 2n-bit Brent-Kung adder is used to calculate the summation of Ar*Br, A*Br. output of this adder and the result of Ar*Br are the inputs of the sub tractor block whose output is the absolute value of the output of the proposed multiplier. Finally, if the sign of the final multiplication result should be negative, the output of the sub tractor will be negated in the sign set block. To negate values, which have the twos complement representation, the corresponding circuit based on 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. The significance of the error decreases as the input widths increases. If the negation is performed exactly (approximately), the implementation is called signed MRoBA (SMRoBA) multiplier [approximate SMRoBA (ASMRoBA) 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 MRoBA (UMRoBA) multiplier. Shifted to left to generate the final output, an approximate 44 WTM has been proposed that uses an inaccurate 4:2 counter. In addition, an error correction unit for correcting the rewritten as outputs has been suggested.

To construct larger multipliers, these 44 inaccurate Wallace multipliers can be used in an array structure. Approximate multipliers are based on either modifying the structure or complexity reduction of a specific accurate multiplier and performing the approximate multiplication through simplifying the operation.

PROPOSED MULTIPLIER

PROPOSED APPROXIMATE MULTIPLIER

We will discuss the differences between the traditional and proposed multiplication flows in this section. Then we present our proposed error correction and high accuracy 4-2 compressor circuit. It then adjusts the multiplier by employing dynamic input truncation. The overall design of our suggested approximate multiplier is introduced in Atlast.
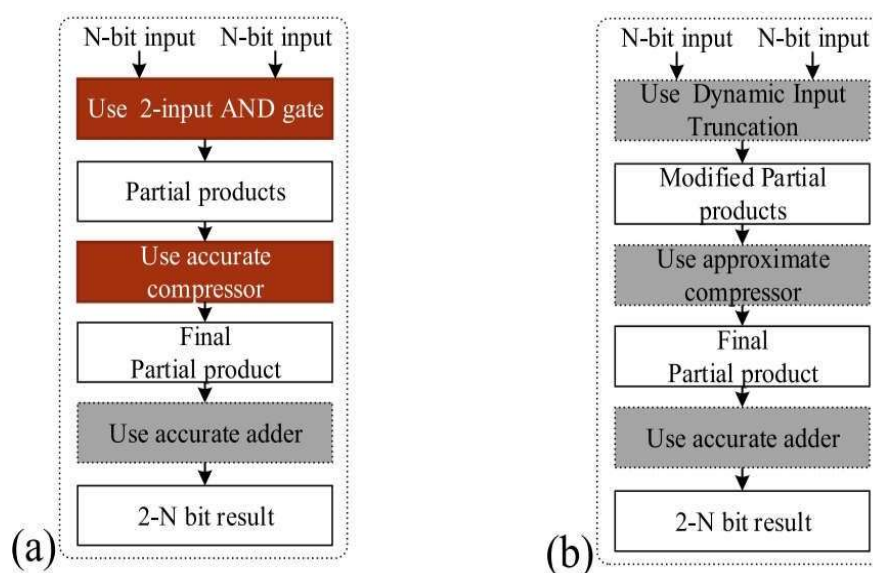
PROPOSED FLOW OF APPROXIMATE MULTIPLIER

Fig 8. (a) traditional (b) proposed multiplication flow

This diagram depicts the general customary flow for multiplication that yields an accurate result. The accurate partial products are first formed using two input AND gates, and then compressed using precise compressors. Finally, to compress the results, the accurate adders sum the compressed partial products. The (b) section of the above figure depicts the projected flow for the proposed approximate multipliers. The steps of creating partial products and compressing the partial products distinguish traditional multiplication from proposed multiplication.

DYNAMIC INPUT TRUNCATION

Dynamic input truncation is a technique for adjusting the accuracy and necessary power. We propose a dynamic input truncation strategy that uses the and gate to achieve the adjustable approximate multiplier. The trunc signal conserves power by reducing the PPD in multiplication to zero. Each bit of an 8 8 multiplier corresponds to 8 bits of the multiplicand; thus, we propose to cut hardware costs by sharing gates with an extra AND gate.
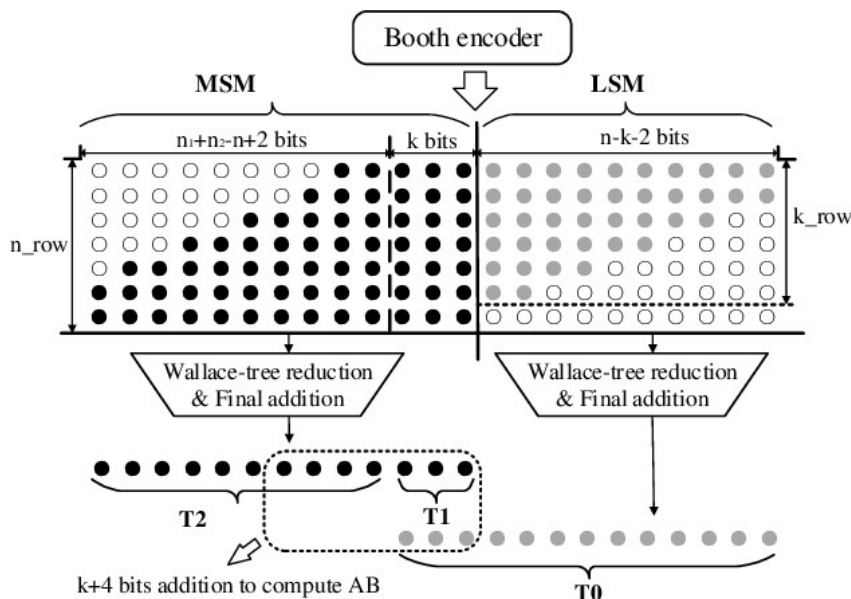

Fig 9. structure of truncated multiplier

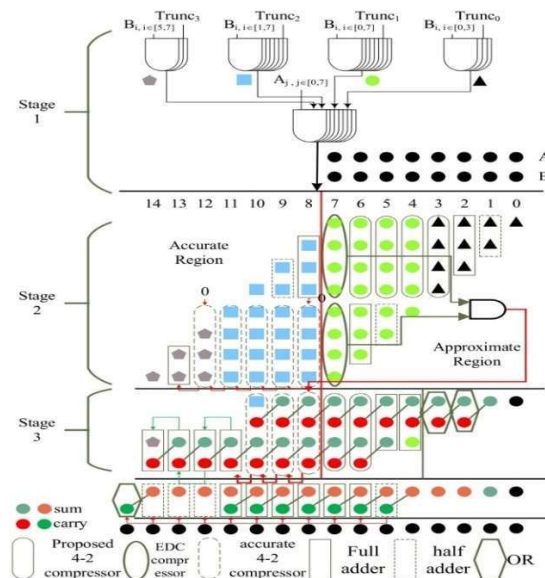**THE PROPOSED TECHNIQUE OF APPROXIMATE MULTIPLIER**

Fig 10. An approximate multiplier with the proposal technique

The figure above depicts an approximation of a multiplier using the proposed technique. Even though the multiplier's input width is limited to 8 bits, the proposed technique can be expanded to larger multipliers. There are three steps in the proposed approximation multiplier. In the first stage, each partial product is generated by two 2-input AND gates, as previously shown, with the gate sharing technique used to cut hardware costs even further. The precision of the resulting partial product can be determined using the shortened signal, depending on the needs.

In our proposed approximate multiplier, we design a 4-bit truncated signal with each bit controlling more than one partial product column, which we call the "3-4-4-4 partition," specifically, each bit from MSB to LSB, corresponding to the colours khaki, sky blue, green, and black in Stage 2.

The way the columns are partitioned provides different possibilities for regulating trunc signals, allowing users to dynamically adjust the recommended multiplier based on their needs. We ran several experiments to test several partitions, and the results reveal that the 3-4-4-4 partition, as well as the 3-3-3-3-3 partition, both maintain a good combination of power savings, accuracy, and area overhead. The finer the partition, the more flexible the control over the amount of power saved and accuracy losses. However, it will suffer from large area overhead in exchange. As a result, we adopt the 3-4-4-4 split throughout all tests.

2D FIR DIGITAL FILTER

The 2D FIR filtering is also called as linear spatial filtering. It is designed using 2D convolution.

$$y(m, n) = x(m,n)*h(m,n) \tag{3}$$
$$y(m,n) = \sum_{k1=0}^{k2-1} x(k1, k2) \sum_{k2=0}^{k1-1} h(m - k1, n - k2) \tag{4}$$

where k1 and k2 are the picture's dimensions in pixels, h (k1, k2) are the filter coefficients, x(m,n) is the original image, and y(m,n) is the filtered image.Figure 1 depicts the direct form structure of a 2D FIR filter with N=4. To create 2D FIR filters in hardware, three basic elements are required: an adder, a multiplier, and a delay element. The multiplier has the greatest influence on speed, area, and power of these three factors. The multiplier's speed determines the execution time and speed of the FIR filter processor.

However, the majority of existing architectures are fundamentally based on the traditional multiplier-based design, resulting in hardware-expensive multipliers as well as excessive energy usage. Because of these performance limitations, they are typically inappropriate for embedded systems with severe energy efficiency requirements.

The number of multiplications completed in a unit of time is used to determine the performance of the microcontroller and digital signal processor. As a result, selecting a high-speed multiplier increases the performance of the 2D FIR filter. The goal of this project is to create a 2D FIR filter using Vedic Mathematics. Many prior studies concentrated on the implementation of FIR filters using various multipliers.
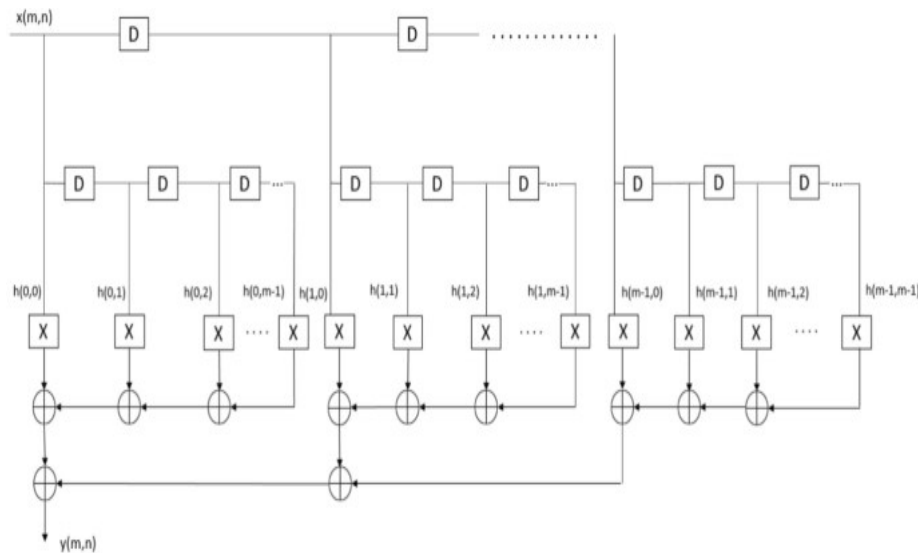
Fig 11. 2D FIR filter for image processing applications

RESULTS AND DISCUSSION

Low power, high speed, and space efficient circuits are preferred for multipliers in the VLSI era. Arithmetic circuits, adders, and multipliers are essential in the design of any signal processing of medical imaging applications. The performance of the adders and multipliers has a significant impact on the overall performance of the circuits. For high-performance applications, efficient adder and multiplier circuits are required. We offer a speed, area, and power efficient approximate multiplier architecture that is appropriate for signal processing, multimedia processing, machine learning, scientific computing, and other applications in this work.

This paper proposes an approximate multiplier with error tolerance. When compared to the Vedic multiplier, booth multiplier, and Wallace multiplier, the approximation multiplier is more efficient with less power, delay, and area. The proposed approximate multiplier is developed using Xilinx ISE and virtex-4 FPGA to investigate its performance. The results are compared to existing multipliers such as the Wallace tree multiplier, the booth multiplier, and the Vedic multiplier.

**Table 1. Comparison of proposed approximate multiplier with existing multiplier**

Attributes	Wallace tree multiplier	Booth multiplier	Vedic multiplier	Proposed approximate multiplier
No. of slice LUTs	96/53200	33/53200	124/53200	29/53200
Path delay in ns	11.954	5.400	14.739	4.828
Power in W	13.693	7.231	14.232	7.156

The area consumed by the multiplier is determined by the number of slices LUTs occupied by the FPGA. The delay is measured in nanoseconds. The results reveal that the suggested approximate multiplier is more efficient than the other multipliers in terms of latency, area, and power. When compared to the present multipliers, the utilisation of slice LUTs is also quite low.
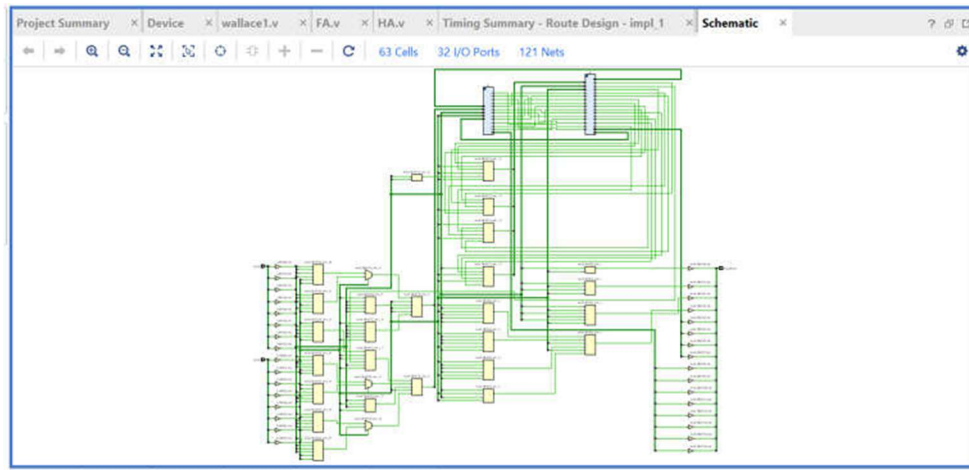
Fig 18. schematic of Wallace tree multiplier

The above figure shows that the schematic diagram of Wallace tree multiplier
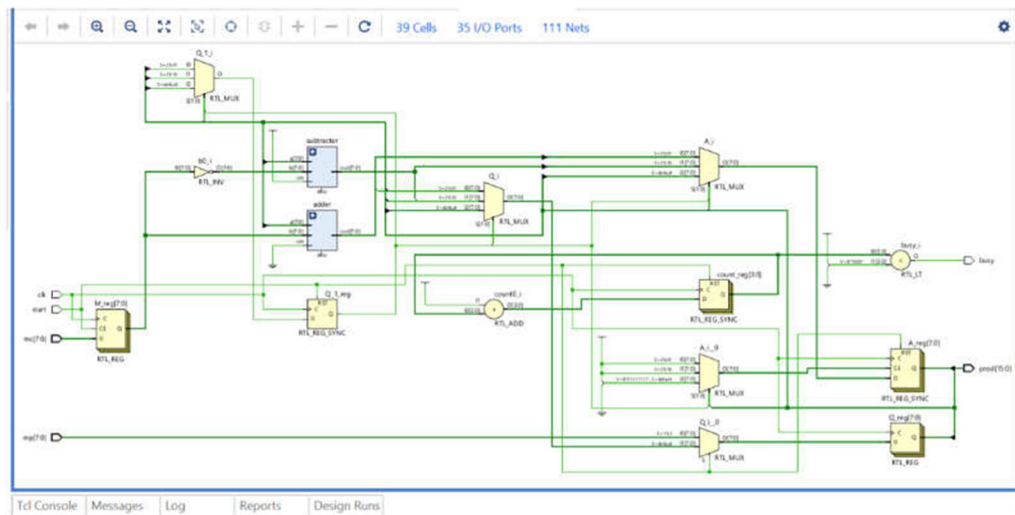
Fig 19. schematic of booth multiplier

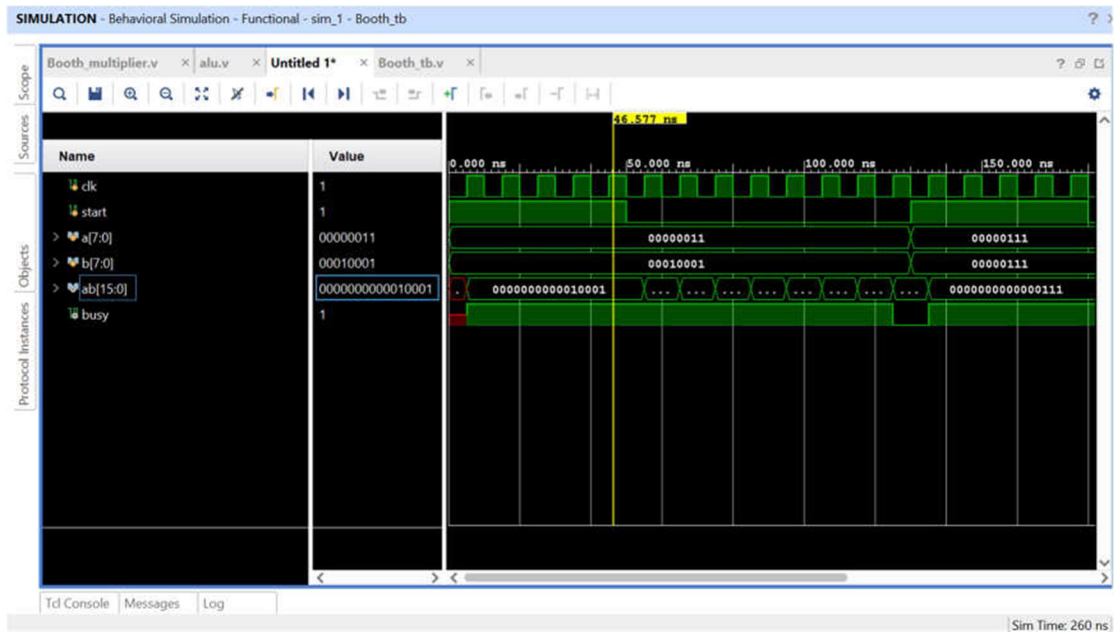The above figure shows that the schematic of the booth multiplier

Fig 20. Simulation of booth multiplier

The above figure shows that the simulation of the booth multiplier
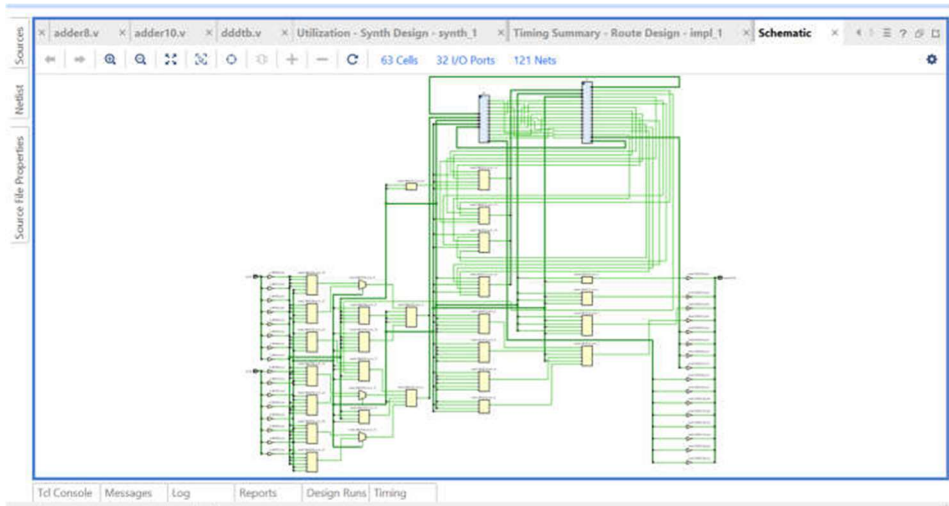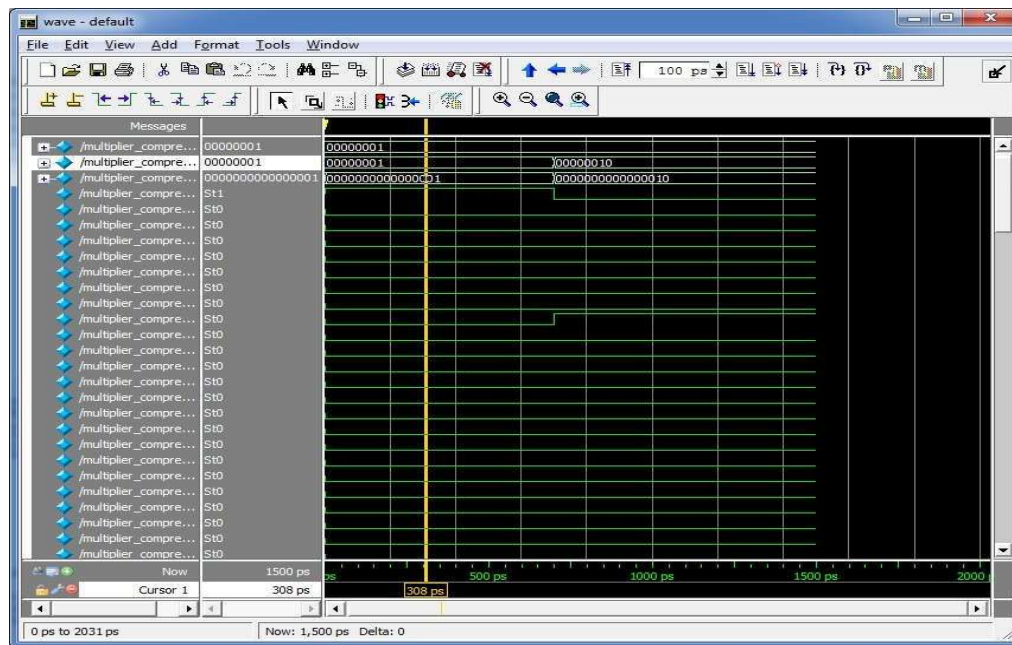

Fig 21. Schematic of Vedic multiplier

The above figure shows that the schematic of the Vedic multiplier

Fig 22. Simulation waveform for booth approximate multipliers

The above figure shows that the simulation wave form for the booth approximate multiplier.

CONCLUSION

In this work, we discovered that the approximation multiplier outperforms the Wallace tree multiplier, the Vedic multiplier, and the booth multiplier. Image processing programmes make use of multipliers. The suggested multiplier is written in Verilog HDL, synthesised with Xilinx, and simulated. When compared to the other multipliers, the findings demonstrate a significant improvement in terms of speed, power, and area. The reduction in power consumption achieved by approximate multipliers can have significant benefits, especially in energy-constrained systems such as mobile devices or battery-powered applications. Lower power consumption leads to longer battery life, increased efficiency, and reduced heat dissipation, which can enable the development of more compact and portable devices. Overall, approximate multipliers offer a promising approach to achieving low power consumption while maintaining acceptable levels of accuracy in various computing systems. Continued research and advancements in approximation techniques can further enhance their capabilities and broaden their application

REFERENCES

[1] W. Liu, J. Xu, D. Wang, C. Wang, P. Montuschi and F. Lombardi, "Design and Evaluation of Approximate Logarithmic Multipliers for Low Power Error-Tolerant Applications," in *IEEE Transactions on Circuits and Systems I: Regular Papers,* vol. 65, no. 9, pp. 2856-2868, Sept. 2018.

[2] K. C. Pathak, A. D. Darji and J. N. Sarvaiya, "Low power Dadda multiplier using approximate almost full adder and Majority logic-based adder compressors," *2022 IEEE Region 10 Symposium (TENSYMP)*, Mumbai, India, 2022, pp. 1-6.

[3] S. Mazahir, O. Hasan, R. Hafiz and M. Shafique, "Probabilistic Error Analysis of Approximate Recursive Multipliers*," in IEEE Transactions on Computers*, vol. 66, no. 11, pp. 1982-1990, 1 Nov. 2017.

[4] Y. Zhao, T. Li, F. Dong, Q. Wang, W. He and J. Jiang, "A New Approximate Multiplier Design for Digital Signal Processing," 2019 IEEE 13th International Conference on ASIC (ASICON), Chongqing, China, 2019, pp. 1-4.

[5] M. K. Sukla, K. Sethi and A. K. Panda, "Low-power and Area Efficient Approximate Multiplier with Reduced Partial Products," *2020 IEEE VLSI DEVICE CIRCUIT AND SYSTEM (VLSI DCS),* Kolkata, India, 2020, pp. 181-186.

[6] 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*," in IEEE Transactions on Very Large-Scale Integration (VLSI) Systems,* vol. 25, no. 2, pp. 393-401, Feb. 2017.

[7] I. Alouani, H. Ahangari, O. Ozturk and S. Niar, "A Novel Heterogeneous Approximate Multiplier for Low Power and High Performance," in *IEEE Embedded Systems Letters,* vol. 10, no. 2, pp. 45-48, June 2018.

[8] S. Kim and Y. Kim, "High-performance and energy-efficient approximate multiplier for error-tolerant applications*," 2017 International SoC Design Conference (ISOCC),* Seoul, Korea (South), 2017, pp. 278-279.

[9] A. Mehta, S. Maurya, N. Sharief, B. M. Pranay, S. Jandhyala and S. Purini, "Accuracy-configurable approximate multiplier with error detection and correction," *TENCON 2015 - 2015 IEEE Region 10 Conference*, Macao, China, 2015, pp. 1-4.

[10] S. Pabithra and S. Nageswari, "Analysis of Approximate Multiplier Using 15–4 Compressor for Error Tolerant Application," *2018 International Conference on Control, Power, Communication and Computing Technologies (ICCPCCT),* Kannur, India, 2018, pp. 410-415.

[11] A. G. M. Strollo, E. Napoli, D. De Caro, N. Petra and G. D. Meo, "Comparison and Extension of Approximate 4-2 Compressors for Low-Power Approximate Multipliers," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 67, no. 9, pp. 3021-3034, Sept. 2020.

[12] K. Shirane, T. Yamamoto, I. Taniguchi, Y. Hara-Azumi, S. Yamashita and H. Tomiyama, "Maximum Error-Aware Design of Approximate Array Multipliers*," 2019 International SoC Design Conference (ISOCC)*, Jeju, Korea (South), 2019, pp. 73-74.

[13] T. U. Swathi Krishna, K. S. Riyas, Y. Premson and R. Sakthivel, "15–4 Approximate Compressor Based Multiplier for Image Processing*," 2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI),* Tirunelveli, India, 2018, pp. 671-675.

[14] F. -Y. Gu, I. -C. Lin and J. -W. Lin, "A Low-Power and High-Accuracy Approximate Multiplier with Reconfigurable Truncation," in *IEEE Access*, vol. 10, pp. 60447-60458, 2022.