Implementation of Fir Filter Using ROBA Multiplier for DSP Applications

Ms. K. sukanya[1], B. Raj kumar[2], B. Gopi[3], B. Ganesh Kumar[4]

[1]*Associate Professor Department of Electronics and Communication Engineering, TKR College of Engineering and Technology.*

[2,3,4]*UG Scholars, Department of Electronics and Communication Engineering, TKR College of Engineering and Technology, Medbowli, Meerpet.*

***ABSTRACT*- The increasing complexity of the DSP systems demanding higher computational performance in its architecture. But the traditional DSP arithmetic has limits in terms of speed of calculations. Moreover, in some applications speed is more important than accuracy. To further enhance performance, approximate arithmetic circuits are designed with some loss of accuracy to reduce energy consumption and increase the speed. These approximate circuits have been considered for error-tolerant applications. In this paper, we propose an FIR filter based on Rounding Based Approximate (ROBA) Multiplier. In this multiplier the operands are rounded to the nearest exponent of two. This approximation will lead to simplification of multiplication operation thus reducing area and increasing speed. As the multiplier is the slowest element in the system, it will affect the performance of the overall FIR filter. The proposed ROBA multiplier-based FIR filter was compared with HAAM and DRUM multiplier-based FIR filters. The results shown significant reduction in the power and area of the FIR filter with proportional improvement in the multiplier speed.**

***KEY WORDS*- Approximation, Energy Efficiency, High speed Multiplication, Digital Signal Processing, ROBA Multiplier.**

I. INTRODUCTION

Digital filters are widely used in the electronics industry due to their superior noise performance over analog filters. Unlike analog filters, digital filters allow noiseless mathematical operations at each stage, enabling precise signal processing. Key operations in digital filters include addition/subtraction, multiplication (typically by constants), and time delay. Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) filters are the two primary types used in VLSI signal processing. FIR filters, which do not use feedback and maintain linear phase, are simpler to design and are preferred in high-speed DSP processors. These filters often use fixed-point arithmetic and fractional representations to minimize bit width while maintaining accuracy.

In digital signal processing, multiplication is critical but hardware-intensive, affecting power consumption and delay. Therefore, designing efficient multipliers is vital. This paper proposes a high-speed, low-power approximate multiplier suitable for error-resilient DSP applications. It introduces a modified ROBA multiplication approach and presents three hardware architectures for both signed and unsigned operations. The proposed approximate multiplier is area-efficient and constructed by simplifying conventional algorithms based on rounded input values. This architecture is integrated into FIR filter design, optimizing bit width and hardware resources without compromising output precision or frequency response.

II. LITERATURE SURVEY

1. Gupta (2011) introduced approximate computing as a power-efficient approach for DSP applications. It shows how approximate multipliers, like ROBA, can reduce power and area significantly while maintaining acceptable output quality, especially in error-tolerant applications like image and audio processing.

2.Jiang(2017) The study proposes several approximate multipliers and evaluates their performance in FIR filters. It concludes that approximate multipliers reduce power and delay with minimal degradation in signal quality, validating their effectiveness in DSP systems

3.Hasan (2019) work explores truncated and approximate multipliers for FIR filters to achieve hardware efficiency. It highlights that replacing conventional multipliers with truncated or approximate versions (like ROBA) results in significant savings in silicon area and power without a major impact on filter accuracy.

III. METHODOLOGY

i. EXISTING METHADOLOGY

An Array multiplier can be interpreted as a multiplier that consists of a large array of adders. Alternatively, it can also be viewed as a multiplier that processes a smaller array of adders several times to complete the product.

It is used as a multiplication block in which an array of identical cells generate partial products and the partial products are accumulated simultaneously.

The parallel implementation isused in high performance machines, where the number of computations needs to be minimized. An m*n array multiplier requires, m*n AND gates and (m-1) n bit adders

APPROXIMATE MULTIPLIERS:

Approximation can be performed using different techniques such as allowing some timing violations (e.g., voltage over scaling or over-clocking) and function approximation techniques (e.g., modifying the Boolean function of a circuit) or a mixture. Approximate multiplier designs mainly use three approximation approaches:

- Approximation in generating the partial products.
- Applying truncation in the partial product tree.
- Using approximate adders &compressors to accumulate the partial products.

ii. PROPOSED METHADOLOGY

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

A*B=((Ar-A)*(Br-B))+(Ar*B)+(Br*A)-(Ar*Br) …..(1)

The key observation is that the multiplications of Ar × Br, A r × B, and Br × A 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 differences of the exact numbers from their rounded ones, is typically small. Hence, we propose to omit this part from (2), helping simplify the multiplication operation. Hence, to perform the multiplication process, the following expression is used:

A*B=(Ar*B)+(Br*A)-(Ar*Br) ……..(2)

The ROBA (Rounding Based Approximate) multiplier optimizes multiplication by replacing it with three shift and two addition/subtraction operations. It works by rounding operands A and B to the nearest powers of two (2^n), specifically favoring larger rounded values like 2^p over 2^{p-1} to simplify hardware logic, especially when A or B equals 3 × 2^(p−2). This approach allows smaller logic circuits and efficient hardware implementation. Unlike earlier methods that always underestimate the result, the ROBA multiplier can yield either larger or smaller results depending on how A and B compare to their rounded counterparts Ar and Br. If one operand is below and the other above its rounded value, the product may exceed the exact result. The method is most effective for positive inputs, as two's complement rounding doesn't align with 2^n for negative numbers. To address this, inputs are converted to absolute values, and signs are determined separately before unsigned multiplication.

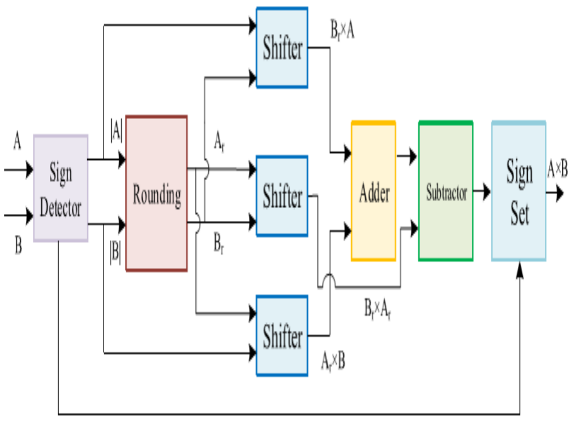

Fig 2: Block diagram for the hardware implementation of the proposed multiplier.

COMPARISON OF ROBA MULTIPLIER WITH OTHER APPROXIMATE MULTIPLIER

DRUM: The Dynamic Range Unbiased Multiplier (DRUM) is designed for approximate multiplication with high scalability and minimal average error. It dynamically focuses on the most significant bits of the operands, enabling efficient and configurable multiplications. DRUM uses two Leading One Detector (LOD) circuits to identify the highest order

'1' in each operand. Based on this, it selects *k–2* subsequent bits to form *k*-bit operands for accurate core multiplication, where *k* is designer-defined for precision control. The remaining lower bits are approximated using their expected value, modelled as a single '1' at position *t–k+1* (where *t* is the leading '1' index) with zeros below. This reduces error while maintaining a consistent approximation. These truncated and biased-corrected operands are multiplied using a *k*×*k* multiplier, and the result is shifted appropriately via a barrel shifter to produce the final output. The unbiased design of DRUM ensures that errors tend to cancel out over multiple operations, improving overall accuracy in DSP applications.

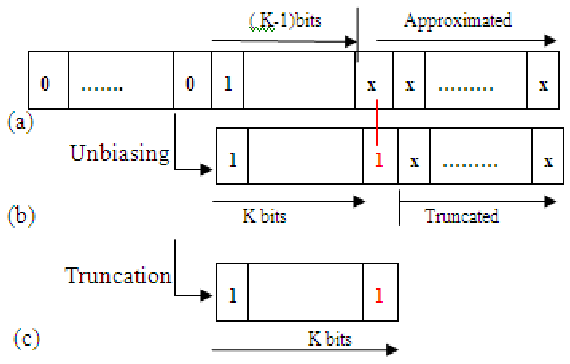

Fig 3: A general example of the approximation process. (a)Original number. (b) Number after biasing. (c) Final approximated input.

		0001 0111 0100 1101
(a)	*	0000 0001 0101 1010
		101111
(b)	*	101011
(c)		0111 1110 0101
(d)		0000 0000 0001 1111 1001 0100 0000 0000
(e)		0000 0000 0001 1111 0111 1110 0001 0010

Fig 4: A numeric example of the DRUM multiplier (n=16, k=6) (a) The input numbers (b) Approximate inputs. (c) Result before shifting (d)Approximate result.(e) Accurate result.

The DRUM multiplier hardware consists of two main components: steering logic and arithmetic logic. The steering logic dynamically selects the most significant bits of the input operands using LOD blocks, encoders, multiplexers, and a barrel shifter, then adjusts the final output position. The arithmetic logic is a small core multiplier, commonly a Wallace-Tree multiplier due to its efficiency. This modular design allows flexibility in choosing the core multiplier. By focusing on only the essential bits, the architecture significantly reduces power and area. If the leading '1' lies within the least significant *k* bits, those bits are directly forwarded for multiplication.

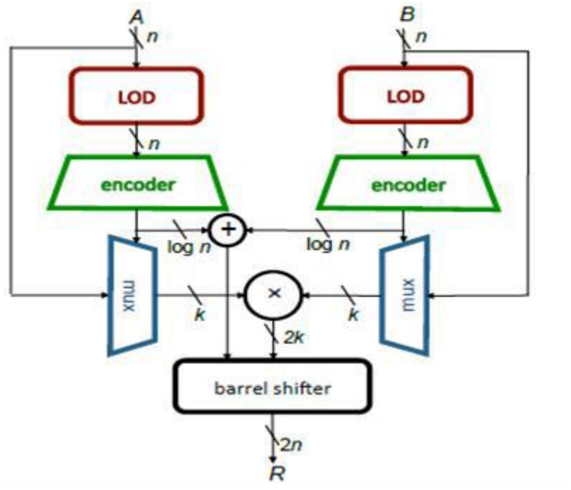

Fig 5: Block diagram DRUM Multiplier

IV. FIR FILTER IMPLEMENTATION

In the FIR system, the impulse response is of finite duration, this means that it has a finite number of nonzero terms. The response of the FIR filter depends only on the present and past input samples.

A. Normal FIR filter:

The conventional design of the FIR filter is shown in Figure 8.The implementation of an FIR requires three basic building blocks: Multiplication, Addition and Signal delay. FIR filter can be expressed as

$$y[n] = \sum_{k=0}^{N-1} h[k] \cdot x[n-k]$$

Where N represents the filter order, y [n] is the output signal and bk represents the set of filter coefficients. If x[n] is the input signal applied, x [n - k] terms are referred as taps or tapped delay lines. In the conventional design, present a simple structure of multiplier for FIR filters. It performs multiplication by generating partial products. If the multiplier digit is a 1, the multiplicand is simply copied down and represents the product. If the multiplier digit is a 0 the product is also 0. Therefore the area and delay will increase. It affects the performance of the FIR filter.

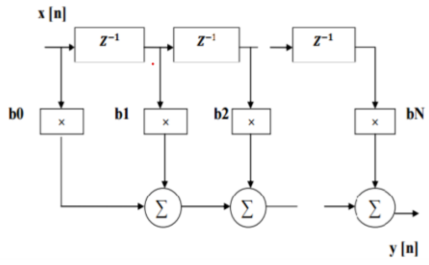

Fig 6: Block diagram of FIR filter.

FIR Filter Using ROBA multiplier

As the multiplier is the slowest element in the system, it will affect the performance of the FIR filter. So, ROBA multiplier is suggested since it reduces area and it is faster than other conventional multipliers.

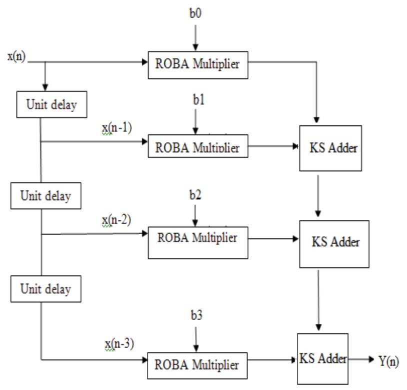

Fig 7: Block diagram of FIR filter using ROBA Multiplier

Kogge-Stone adder

Kogge-Stone adder is a parallel-prefix form carry look ahead adder. Kogge-Stone adder was developed by [3] Peter M. Kogge and Harold S. Stone which they published in1973. KS adder is a fast adder design as it generate carry signal in O(log2 n) time and has the best performance in VLSI implementations. KS adder has large area with minimum fan-out which increases its performance. Kogge-Stone adder is widely used in high performance 32-bit, 64-bit, and 128-bit adders as it reduces the critical path to great extent. In fig.3 each vertical stage produce propagate and generate bits. Generate bits are produced in the last stage and XORed with initial propagate and generate bits to produce sum.

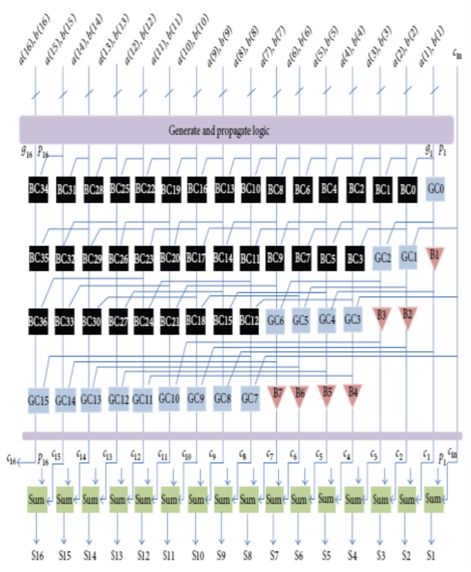
Generate and propagate logic
Sum
S16 S15 S14 S13 S12 S11 S10 S9 S8 S7 S6 S5 S4 S3 S2 S1

Fig 8: Illustration of 16bit Kogge Stone Adder.

V. RESULTS

Figure 8 shows Simulation result for ROBA multiplier and figure 13 shows the implemented FIR filter using proposed multiplier. The simulations are performed by using Xilinx ISE simulator.

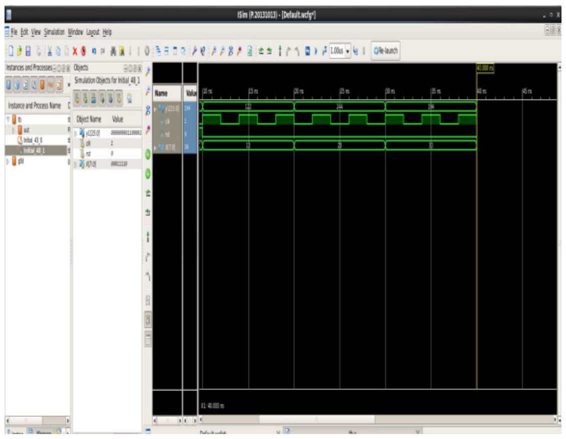

Fig 9: Simulation result of ROBA multiplier.

When compared the results of various approximate multipliers with respect to area, delay and power the following observations are made. From the table 1 the number of LUTs occupied by ROBA multiplier were very small when compared with the other approximate multipliers. This is due to the multiplication process has been significantly simplified by rounding the values to the nearest power of two. Even though this results a small error in the output value the area of the multiplier has been drastically reduced. This reduced area results faster multiplication and lower power consumption which is more suitable for error tolerant DSP applications.

Table1: Comparison of ROBA multiplier with other approximate multipliers

Name Of The System	ROBA Multiplier	HAAM Multiplier	DRUM Multiplier
Number of slice LUT's	192	1785	994
Number of bonded IOB's	129	130	130
DELAY	13.793ns	19.505ns	21.073ns
Power at 80MHz	1.19mW	1.215mW	1.307mW

VI. CONCLUSION

An approximate multiplier (ROBA) based FIR filter implementation was proposed in the paper. The results show that the proposed multiplier shows better performance in terms of area, power and delay. The rounding based approximation resulted in reduced area of the FIR filter when compare to other approximations. Moreover reduced area resulted in less delay an power consumption. The system shows 40% speed improvement an occupying only 1/5th of the area of other approximate multipliers.

REFERENCES

[1] Reza Zendegani, Mehdi Kamal "RoBA Multiplier: A Rounding-Based Approximate Multiplier for High-Speed yet Energy-Efficient Digital Signal Processing" IEEE Transactions on very large scale integration (VLSI) systems, VOL.25, NO.2, February 2017.

[2] Deepika Kumari M Design and Implementation of RoBA Multiplier on MAC Unit International Journal for Research in Applied Science & Engineering Technology (IJRASET) Volume 7 Issue VII, July 2019.

[3] J. N. Mitchell, "Computer multiplication and division using binary logarithms," IRE Trans. Electron. Compute. Vol. EC-11, no. 4, pp. 512–517, Aug. 1962

[4] K. Y. Kyaw, W. L. Goh, and K. S. Yeo, "Low-power high-speed multiplier for error-tolerant application," in Proc. IEEE Int. Conf. Electron Devices Solid-State Circuits (EDSSC), Dec. 2010, pp. 1–4.

[5] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," IEEE Trans. Compute.-Aided Design Integer. Circuits Syst., vol. 32, no. 1, pp. 124–137, Jan. 2013.

[6] M. Alioto, "Ultra-low power VLSI circuit design demystified and explained: A tutorial," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 59, no. 1, pp. 3–29, Jan. 2012.

[7] J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and probabilistic adders," IEEE Trans. Compute., vol. 62, no. 9, pp. 1760–1771, Sep. 2013.

[8] V. Maralinga and N. Ranganathan, "Improving accuracy in Mitchell's logarithmic multiplication using operand decomposition," IEEE Trans. Compute., vol. 55, no. 12, pp. 1523–1535, Dec. 2006.

[9] K. Bhardwaj and P. S. Mane, "ACMA: Accuracy-configurable multiplier architecture for error-resilient system-on-chip," in Proc. 8th Int. Workshop Reconfigurable Common.-Centric Syst.-Chip, 2013, pp. 1–6.

[10] H. R. Myler and A. R. Weeks, The Pocket Handbook of Image Processing Algorithms in C. Englewood Cliffs, NJ, USA: Prentice-Hall, 2009.

[11] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," IEEE Trans. Compute.-Aided Design Integer. Circuits Syst., vol. 32, no. 1, pp. 124–137, Jan. 2013.

[12] Young-Ho See and Dong-Wok Kim, "New VLSI Architecture of Parallel Multiplier-Accumulator Based on Radix-2 Modified Booth Algorithm," IEEE Transactions on very large scale integration (VLSI) systems, vol. 18, no. 2,february 2010. Cong Liu, (March 28, 2014), "A Low-Power, High-Performance Approximate Multiplier with Configurable Partial Error Recovery".

[13] Farshchi.F, Abrishami.M.S, And S. M. Fakhraie, (Oct. 2013) "New Approximate Multiplier For Low Power Digital Signal Processing," In Proc. 17th Int.Symp. Comput. Archit. Digit. Syst. (Cads), Pp. 25–30.

[14] Lin.C.H And Lin.L.C,(2013), "High Accuracy Approximate Multiplier With Error Correction," In Proc. 31st Int. Conf. Comput. Design (Iccd),Pp. 33–38.

[15] Mounika.K and Jawaharla.R,(July 2016). "Performance for Delay, Power and Area for Parallel Prefix Adders with Xilinx".

[16] Narayanamoorthy.S, Moghaddam.H.A, Liu.Z, Park.T, and Kim.N.S, (Jun 2015), "Energy-Efficient Approximate Multiplication for Digital Signal Processing and Classification Applications," Very Large Scale Integration. (Vlsi) Syst., Vol. 23, No. 6, Pp. 1180–1184.

[17] Reza Zendegani, Mehdi Kamal, Milad Bahadori, Ali Afzali-Kusha, and Massoud Pedram, (Feb 2017) "Roba Multiplier: A Rounding-Based Approximate Multiplier for High-Speed yet Energy-Efficient".

[18] Sumant Dalmiya ,(December 2015) "A Comparative Study Of Adders", B.E.

[19] Sumit Vaidya And Deepak Dandekar ,(2010) , "Delay-Power Performance Comparison Of Multipliers In Vlsi Circuit Design".

[20] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 57, no. 4, pp. 850–862, Apr. 2010.