E-ISSN 2581 –7957
P-ISSN 2277 – 3916

CVR Journal of Science and Technology, Volume 26, June 2024
DOI: 10.32377/cvrjst2608

# Area-Delay-Power Efficient VLSI Architecture 2D FIR Filter using Modified Multipliers and Adders

Dr. Venkata Krishna Odugu[1]. Dr. B Janardhana Rao[2] and Dr. G Harish Babu[3]
[1]Associate Professor, CVR College of Engineering/ECE Department, Hyderabad, India
Email: venkatakrishna.odugu@gmail.com
[2]Associate Professor, CVR College of Engineering/ECE Department, Hyderabad, India
Email: janardhan.bitra@gmail.com
[3]Sr. Assistant Professor, CVR College of Engineering/ECE Department, Hyderabad, India
Email: harish.sidhu12@gmail.com

***Abstract:*** **In this paper, a low power, area, and delay 2D Finite Impulse Response (FIR) filter architecture is derived from an analysis of a memory-efficient design. The completely direct-form 2D FIR filter is where the idea of parallel processing is first presented. As a result, the FIR filter may make better use of its memory by reusing its contents. With a block size of L and a filter length of N, a non-separable 2D FIR filter structure is developed and implemented. The FIR filter's arithmetic module makes use of high-speed, low-power multipliers and Carry Look Ahead (CLA) adders, with the output calculated by a pipelined adder unit. Verilog HDL code is used to represent the proposed architecture, and the CADENCE environment's NC Simulator and RTL Compiler synthesis tool are used to verify the design. Existing memory-efficient 2D FIR filter hardware architectures are compared to the produced area, power, and delay reports. Using Modified CLA (MCLA) adders and pipelining, we were able to cut down on power consumption by 44% and delay by 20%.**

***Index Terms:*** **Memory reuse, 2D-FIR filter, Low Power Multiplier, Parallel Prefix Adder, Carry Look Ahead adder, and pipelining.**

## I. Introduction

The most common applications for 2D digital filters are in image and video processing as well as bio-medical signal processing [1]. In biometric systems, 2D filters are sought after for use in feature extraction [2] and face recognition [3]. While the idea of a 2D filter may be used to both FIR and Infinite Impulse Response (IIR) fits, the stability and simplicity of the construction make 2D FIR filters more widely used. Some research is done on current architectures to implement a memory-efficient and less hardware-complex 2D FIR design. In [4], we learn about the 2D symmetry filters. Several symmetrical IIR and FIR filters are described, along with a study of their hardware metrics and VLSI (Very Large Scale Integration) designs. Here, after the requisite symmetry is achieved, the un-symmetric frequency response is split into sub-components. IIR and FIR filters with reduced multiplier counts and four-fold symmetry are presented in this study.

New 2D VLSI filter designs based on sub-filter blocks with a local connectivity structure and no global broadcasting are derived from the specified generalized formulae in [5]. With this study, we are able to create IIR filters with decoupled numerators and FIR filters with quadrant symmetry, both of which have the advantage of requiring fewer multipliers.

For 2D FIR filters, many systolic topologies are used for optimization of area, power, and latency. There aren't many studies that thoroughly investigate the idea of 2D Filters. To construct 2D systolic FIR and IIR filters, the authors of [6] develop a novel systolic transformation approach and modify reordering algorithms. Lower quantization error, local broadcast, no latency, and good critical routes are all accomplished by the combination of these two methods.

By rearranging the delay components and summing them up, novel VLSI systolic array FIR and IIR filter architectures [7] may be produced. A low-latency, locally broadcast architecture with a suitable number of multipliers and delay components is provided in this.

One-dimensional (1D) and two-dimensional (2D) filter bit-level VLSI designs are explored in [8]. These architectures are consistent, modular, and adaptable to a wide variety of specialized setups. Throughputs and hardware utilization are enhanced with reduced delay as a result of our effort. The modularity and simplicity of these structures make them suitable for optimization.

To counteract the worldwide signal broadcast, these preexisting works incorporate several delay or storage devices along the data stream. In current architectures, memory complexity is a serious problem. Memory's complexity impacts the building's footprint and energy needs [9]. In [10], a memory-centric 2D FIR filter is presented, although it incurs a power and latency cost in non-separable and separable models. The hardware modules in this architecture are L times more powerful than in prior efforts, but the throughput is only L times higher. The greater density of hardware components translates into higher requirements for both space and energy.

In this work, we offer a memory-efficient, low-power, small-area, and low-delay design for a 2D FIR filter. The completely direct-form 2D FIR filter presented here takes advantage of block-based input processing to minimize data storage and maximize memory reuse. Registers are located only in the input data route in the completely direct-form construction, whereas in the entirely transpose-form structure, they are located at the intermediate signal level [10]. Various area and power efficient 2D FIR filter using parallel processing and single input processing filter architectures are presented in the works [11-14].



Received on 18-10-2023, Revised on 17-11-2023, Accepted on 17-12-2023.

CVR Journal of Science and Technology, Volume 26, June 2024
DOI: 10.32377/cvrjst2608

The original, totally direct-form structure is transformed into an efficient block-based design that takes advantage of previously allocated memory. The following key contributions of this work are:

- Bypass Zero Feed Multiplicand Directly (BZ-FMD) multiplier is used to improve the Functional Unit (FU) in the arithmetic module of the 2D FIR filter design.
- Power consumption is dominated by dynamic power in VLSI design. In the suggested multiplier, the decrease in switching activities lowers the dynamic power.
- For the speedy addition of partial products, the multiplier's performance is further enhanced with a parallel prefix adder.
- Using CLA's fast adder logic, we have created this lightning-quick parallel prefix adder. In order to improve the standard CLA's efficiency in terms of speed, area, and power, it changes. These MCLAs are also used to add the filter at the end.
- The final addition of the FIR filter is computed using MCLAs, a pipelined addition procedure. Overall, the 2D FIR filter structure uses less energy and complete computation time thanks to MCLA adders and pipelining ideas.

The remaining sections of this paper is described as follows: In section 2, we propose the design of a 2D FIR filter, which eliminates computational redundancy. The suggested architecture for implementing a 2D FIR filter and its constituent sub-modules is outlined in section 3. Sections 4 and 5 explain the results and conclusions of the synthesis.

## II. Memory Reuse

The suggested 2D FIR filter architecture takes into account the entirely direct-form structure. The FIR filter's memory reuse is investigated by analyzing the input data flow of the direct-form structure. The redundancy of input samples is displayed in Fig. 1 for the filter length N = 4, which is useful for comprehending the memory reuse idea. If the computation of the output at the $m^{th}$ row is taken into account, the results are y(m, n), y(m,n+1), y(m,n+2), and y(m,n+3). Four columns and four rows of 2D input are needed for the 4 x 4 filter, totaling 16 samples. Serial-In-Parallel-Out (SIPO) Shift Register Blocks (SRB) and shift registers are utilized to provide row and column history samples, respectively.


Fig.1. Data flow in the fully direct form structure for N=4 with four outputs *{ y(m,n), y(m,n+1), y(m,n+2), y(m,n+3)}*, [10].

In a totally direct data flow, only 28 of the 64 samples are unique, while the other 36 are repeated four times. The outputs y(m,n), y(m,n+1), y(m, n+2), and y(m, n+3) are emphasized in Fig.1 along with their associated duplicate samples. Direct-form structures can eliminate redundancy through parallel computing or by employing a block-based structure. To calculate a precise result, it is necessary to examine previous sample values. Every clock cycle [10] allows access to the SRBs and the retrieval of these historical samples.

## III. Implementation of Proposed 2D FIR Filter Architecture

To eliminate unnecessary repetition in the filter's data flow, the suggested block-based FIR filter is constructed in a methodical design. In order to save on storage space in relation to the input data flow, the memory reuse concept is implemented to cut down on redundant sample data. The filter length is N = 8, and there are blocks of size L = 4.

Figure 2 depicts the construction of a 2D FIR filter that uses non-separable blocks. Parallel inputs of size L are used to construct the fully direct form based 2D FIR filter. The two primary components of this design are the memory and arithmetic modules.

P = M/L = 512/4 = 128, hence the memory module has an array of 28 shift registers with a capacity of 128 words and 8 input register units (IRU). Every four of the 28 shift registers form a cluster called an SRB. Seven SRBs (SRB1, SRB2,.... SRB7) are needed for this design.

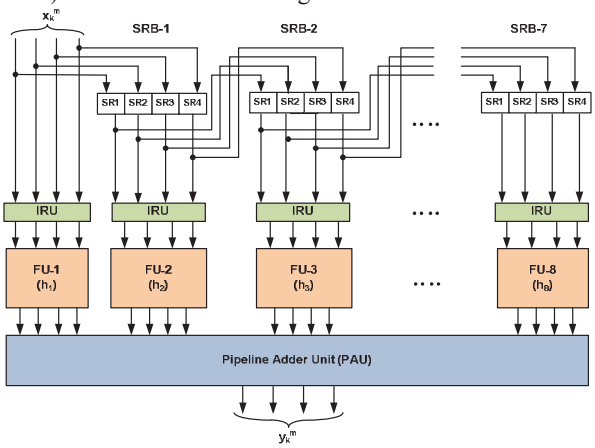


Fig. 2. Block-based non-separable 2D FIR filter architecture.

Each time the clock ticks, the filter processes the block of 4 inputs and generates 4 outputs. This is how the 512 by 512 input picture matrix is processed in serial sequence, block by block, to produce the outputs [15]. By organizing the image into blocks, we may finish it in MP = 512 x 128 clock cycles rather than the usual MP = 512 x 512. Throughput is raised while latency is decreased. Each time the clock ticks, the SRB component supplies N minus one, or seven, input blocks, which correspond to N minus one, or seven, successive input rows.

Eight IRUs are sampled based on the previous seven input blocks and the current input block. For the case of L = 4 and N = 8, Fig. 3 depicts the register layout on the inside that would be used for redundancy-avoiding logic. To generate input vectors of length 8, it uses a total of 7 registers, or D-Flip Flops (N - 1). The input samples, totaling 8 points, come

CVR Journal of Science and Technology, Volume 26, June 2024
DOI: 10.32377/cvrjst2608

from both the past and the present. Since L= 4 and N= 8, each of the 8 - IRUs produces a 4 x 8 matrix of [Ak]; the first IRU gets input samples from the current block and applies this matrix to the FU. Seven identical IRUs produce equivalent 7-FUs with 4x8 matrices.

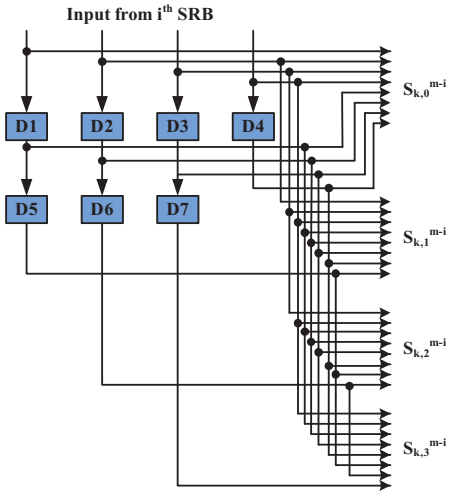


Fig. 3. The internal structure of IRU.

### A. *Arithmetic Module*

The arithmetic module begins with the FU block. To multiply the input vectors by the filter coefficients, N = 8 FUs are needed [ $h_i$]. Each of the eight IRUs sends FU four input vectors. The (i + 1)$^{th}$ FU takes the input vector and the (i + 1)$^{th}$ row of the impulse response matrix from the (i +1)$^{th}$ IRU and calculates the inner product of the two. The notation for the FU output matrix is [ $V_i$].

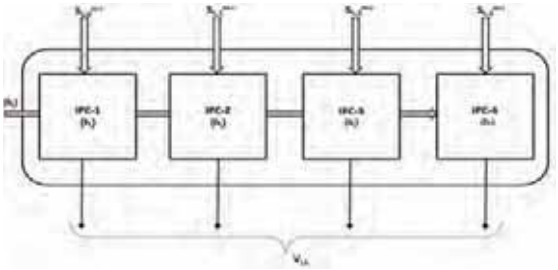

Fig. 4. The Internal block diagram of FU.

Fig.4 shows how FU is put together on the inside. There are four inner product cells (IPC) in FU. IPC is used to multiply the input vectors and the impulse response matrix elements that go with them. Figure 5 shows IPC's core reasoning for putting things together. An adder circuit is used to add the 8 partial products together to get the 8-point core product.

### B. *Multiplier Implementation*

Each IPC needs N multipliers and N-1 adders to combine the input samples and filter coefficients. In this part, we will talk about the improved multiplier and adder circuits. The basic process of multiplication is made up of two steps: making partial products and adding them together. The swapping in a multiplier or any other circuit affects how much power it uses. The general dynamic power in the VLSI circuits is shown by Equation (12.

$$P_{Dynamic} = \alpha . C_L . f_{clk} V_{DD}^2 \quad (1)$$

Where $C_L$ is Load capacitance, $f_{clk}$ is clock frequency, $V_{DD}$ is the power supply and $\alpha$ represents switching activity. The number of switching activities represents the power consumption of the circuit.

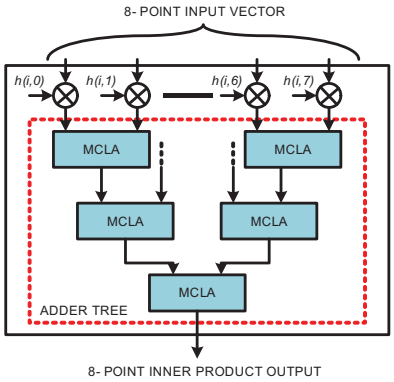


Fig. 5. Internal combinational logic diagram of IPC.

The traditional shift-add multiplier is changed, and a new multiplier is made using the Bypass Zero, Feed A Directly (BZ-FAD) method. This method is described in [15]. The switching activities of the BZ-FAD multiplier depend on (i) moving the bits of the multiplier, (ii) switching the bits of the partial product, (iii) switching the activities of the adder, and (iv) switching the activities of the multiplexer for the final addition. Bypass Zero, Feed Multiplicand Directly (BZ-FMD) is a change to the BZ-FAD that cuts down on delay and cuts down on the number of hardware blocks [16].

Figure 6 shows how the BZ-FMD multiplier is put together. It has an adder, an expander, a product register, a feed register, and a driver. Some parts of the BZ-FAD multiplier are taken out to cut down on delay and space. To make things less complicated, a binary counter is used instead of a ring counter and is put in the control block.

At first, the supervisor checks to see if the 0th bit of the multiplier is a '0' or a '1'. Instead of a ring counter, the driver has a synchronous binary counter that checks each bit and adds one to it every time the clock goes around. The managing block is in charge of everything in the adder, multiplexer, and feed register blocks. The input from the feed register is sent to the adder block or multiplexer through the control block. The multiplier bit decides the output of the multiplexer it should be the feeder register value of a previous partial product or adder output. The adder processing is skipped for the multiplier bit as '0' and the feeder register feeds the previous partial product value directly to the MUX. Otherwise, the sum of the multiplicand value and the previous partial product is directly given to MUX. The switching activity required for the zero-bit addition is eliminated and directly the multiplicand is fed to MUX.

CVR Journal of Science and Technology, Volume 26, June 2024
DOI: 10.32377/cvrjst2608

The multiplier structure effectively reduces the switching activities associated with the shifting of the partial product. In this particular multiplier, the upper half of the partial product bits undergo a rightward shift during processing, but the lower half of the partial product bits are retained and immediately put in the product register. Traditionally, the complete partial product is subjected to a rightward shift, however, in the suggested multiplier, only half of the bits representing the product undergo a shift. The suggested multiplier reduces the switching actions for partial product shifting by 50%.

The proposed multiplier design aims to minimize the switching activities associated with the shifting of multiplier bits, addition, and shifting of partial products. The decrease in switching activity results in a significant reduction in dynamic power usage. This paper presents the implementation of an 8-bit multiplier for the purpose of multiplying filter coefficients and input samples.

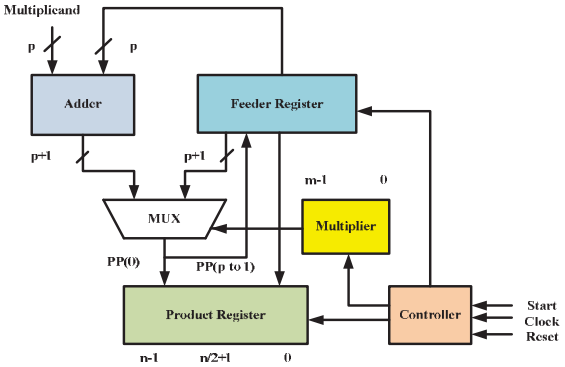


Fig.6. Architecture of multiplier.

### *C. Parallel Prefix adder based on Modified CLA (MCLA)*

The process of multiplication involves a mix of shifting and summing. Higher bit multiplication necessitates the utilization of multiple adders. Therefore, the efficiency of the multiplier is contingent upon the performance of the adder as well. In the implementation of the N-tap filter, the addition of individual tap outputs and the calculation of the final filter output necessitates the use of N adders. The implementation of adder optimization is necessary to mitigate power consumption and delay. This section is a detailed description of a modified high-speed Carry Look Ahead (CLA) adder. The CLA equation has been updated in order to optimize hardware utilization and enhance performance and power efficiency. In this study, the 8-bit and 16-bit parallel prefix MCLA adders are given. The utilization of 8-bit Modified Carry Look-Ahead (MCLA) units is common in the process of adding partial products inside a multiplier. Additionally, 16-bit MCLA adders are employed in the construction of an adder tree.

The Modified Carry Look-Ahead (MCLA) algorithm replaces the usual carry ci with a modified carry. Following the computation of propagating and generating terms, a parallel prefix addition technique is employed to minimize the time required [17]. In this Modified CLA (MCLA), a modified carry $M_i$ is determined in the place of conventional carry $c_i$. After the calculation of propagating and generate terms, a parallel prefix addition concept is used to reduce the time [17].

The modified carry $M_i$ and sum$S_i$of the MCLA is given by equations (2) and (3) respectively.

$$M_i = g_i + g_{i-1} + p_{i-1}.g_{i-2} + p_{i-1}.p_{i-2}.g_{i-3} + \cdots + p_{i-1}.p_{i-2} \ldots . p_1.g_0 \quad (2)$$

Where $g_i$ and $p_i$ are generating and propagate terms in equation (2). This equation is modified to improve the efficiency of the adder. The real carry is given by the equation

$$c_i = M_i.p_i \quad (3)$$

The modified carry for even and odd bit positions are different. The $M_i$ for even and $M_{i+1}$ for odd given by equations (4) & (5),

$$M_i = (G_i^*, P_{i-1}^*) \odot (G_{i-2}^*, P_{i-3}^*) \odot \ldots \ldots \ldots . \odot (G_0^*, P_{-1}^*) \quad (4)$$

$$M_{i+1} = (G_{i+1}^*, P_i^*) \odot (G_{i-1}^*, P_{i-2}^*) \odot . \ldots \ldots \ldots . \odot (G_1^*, P_0^*) \quad (5)$$

The true carry is calculated by first calculating the modified carries for the even and odd bit locations. Employing a Formula (3). The equation (6) is used to determine the total.

$$S_i = d_i \oplus (p_{i-1}.M_{i-1}) \quad (6)$$

Only the least significant bits are used in the aforementioned carry computation algorithms. What follows is a parallel computation of the upper half of the bits carry. The modified carry is calculated in this manner by plugging in values for the intermediate propagate term and the intermediate generate term into Equation (7).

$$c_i = \left(G_{i:k} + P_{i-1:k-1}.G_{k-1:j+1}\right).p_i \quad (7)$$

Equation (8) describes the carry of the eighth bit in a 16-bit adder.

$$M_8 = (G_{8:7}, P_{7:6}) \odot (G_{6:3}, P_{5:2}) \odot (G_{2:-1}, P_{1:-2}) = \left(G_{8:7} + P_{7:6}.G_{6:-1}, P_{7:6}.P_{5:-2}\right) \quad (8)$$

The remaining upper half bits carries are determined using the equations from (9) to equation (1).

$$c_8 = (G_{9:8} + P_{7:6}.G_{6:-1}).p_8 \quad (9)$$

$$c_9 = (G_{9:8} + P_{8:7}.G_{7:0}).p_9 \quad (10)$$

$$c_{10} = (G1_{10:7} + P_{9:6}.G_{6:-1}).p_{10} \quad (11)$$

$$c_{11} = (G_{11:8} + P_{10:7}.G_{7:0}).p_{11} \quad (12)$$

$$c_{12} = (G_{12:7} + P_{11:6}.G_{6:-1}).p_{12} \quad (13)$$

$$c_{13} = (G_{13:8} + P_{12:7}.G_{7:0}).p_{13} \quad (14)$$

$$c_{14} = (G_{14:7} + P_{13:6}.G_{6:-1}).p_{14} \quad (15)$$

$$c_{15} = (G_{15:8} + P_{14:7}.G_{7:0}).p_{15} \quad (16)$$

Taking into account [13] the above-mentioned modifications to the CLA equations, we design and build 8-bit and 16-bit parallel prefix adders, examples of which are

CVR Journal of Science and Technology, Volume 26, June 2024
DOI: 10.32377/cvrjst2608

illustrated in Figs.7 and 8, respectively.

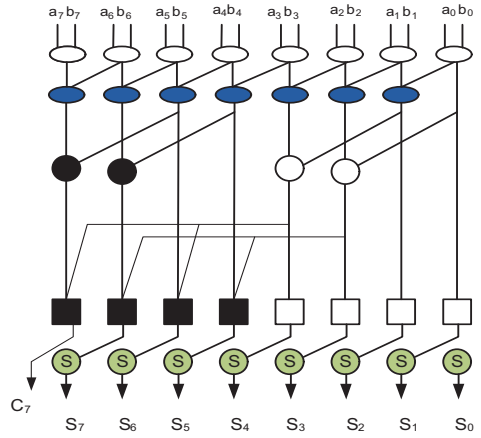


Fig.7. The 8-bit parallel prefix MCLA adder for multiplication

In Fig. 9, we see the logic cells that are needed to perform a simultaneous prefix sum of 8-bit and 16-bit adder structures. The AND, OR, and XOR logic gates are used to implement all of the logic cells.

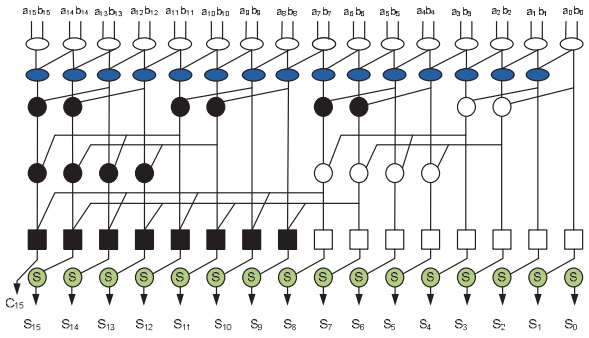


Fig.8. The 16-bit Parallel Prefix MCLA adder for filter outputs summation.

The adder block utilizes a specialized MCLA adder. To compute the sum of IPC, the MCLA adders are organized in a tree structure known as an Adder Tree (AT).

Pipeline Adder Unit (PAU) refers to the FIR filter's last adder block, which functions as the second block of the arithmetic module and generates the filter's output. D-FFs and MCLAs make up the PAU, which performs the final addition of the vectors generated by the FUs. Fig. 10 depicts an inside perspective of the PAU. Parallel processing optimizes the data flow in the input channel, while pipeline computing expedites the computation of the final output. The power consumption and critical route latency of VLSI designs can be minimized by employing pipelining and parallel processing [18, 19].

It takes one clock cycle to generate an output for each 4-sample input block. The delay of all arithmetic blocks is used to determine the minimum clock period used to define each clock cycle. This design has a clock period of $T = T_M + T_{PAU} + T_{MCLA} (2\log_2 N - 1)$, where $T_M$ is the time needed to compute one multiplier, $T_{PAU}$ is the time needed for the PAU, and $T_{MCLA}$ is a delay of the MCLA adder in the adder tree. It takes 128 clock cycles to process a single row of the input picture, and just 128M cycles to process the full image matrix (where M is the size of the image matrix in bytes

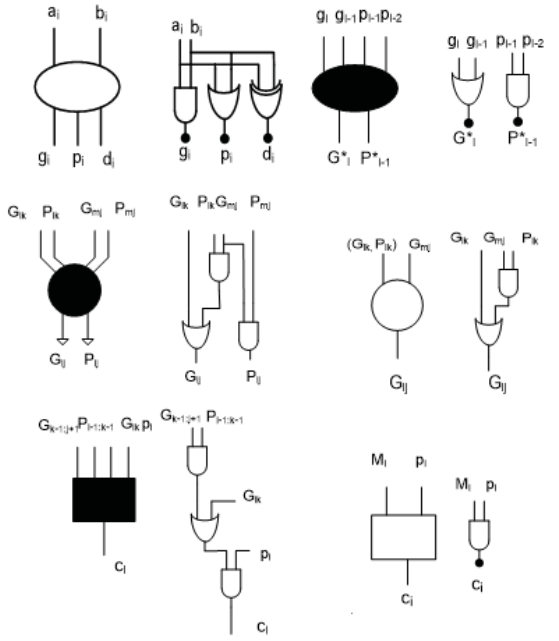


Fig.9. Internal logic cells are used in MCLA adder

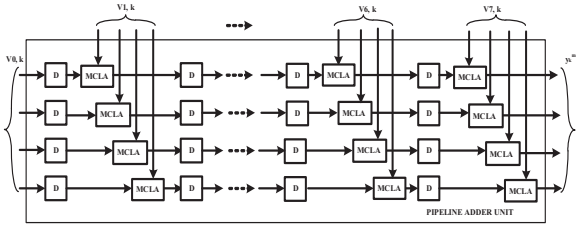


Fig.10. Pipeline Adder Unit (PAU) of 2D FIR structure.

## IV. Implementation Results

The block-based, non-separable structure has $LN^2 = 4 \times 64 = 256$ multipliers, $L (N^2 - 1) = 4 \times 63 = 252$ MCLAs, and $[(M + N) (N - 1)] = 512 + 8 \times 7 = 28672$ registers. Every time the clock goes around, this structure sends out four signals. Non-separable structure's memory reuse efficiency is $L–1 = 3$, and its memory bandwidth per output (MBWPO) is $(L+N)(N–1)/L = 21$. The MBWPO costs 3 times less than the buildings that are already there [10].

The proposed 2D FIR filter design is written in HDL for block sizes L = 2 and 4 and filter sizes N = 4 and 8. The NCSim model from CADENCE tools is used to test the code. CADENCE tools' Encounter RTL translator is used to make the synthetic HDL code in the TSMC 90nm CMOS technology library. The synthesis of design is done with the TSMC 45nm CMOS library's general building blocks library, and D-FFs are used as registers and shift registers. The width of the original sample signal is thought to be b = 8 bits, and the width of the signal in the middle is d = 16 bits.

In Table I, the suggested non-separable 2D FIR filter design with block sizes of L = 2 and 4 and filter lengths of N = 4 and 8 is compared with known 2D FIR filter architectures. For reference, an FIR filter with N=4 blocks of size L= 4 is also put in place. Fig.11 is a picture that shows how the proposed design compares in terms of power to current 2D FIR filter architectures.

TABLE I
COMPARISON OF THE AREA AND POWER PARAMETERS OF DIFFERENT NON-SEPARABLE FIR FILTER ARCHITECTURES (L = 4)

| Structure | Length of the filter (N) | Area ($\mu m^2$) | Power (mW) | | |
|---|---|---|---|---|---|
| | | | Static | Dynamic | Total |
| Proposed 2D FIR | 4 | 32598 | 0.2985 | 2.9856 | 3.2841 |
| | 8 | 40215 | 0.4215 | 3.9547 | 4.3762 |
| Khoo [5] | 4 | 1009878 | 3.9107 | 4.9441 | 8.8548 |
| Mohanty*et. al* [10] | 4 | 791361 | 2.8016 | 3.2918 | 6.0934 |

When comparing structures [10], we find that when N is equal to 4, total power consumption is decreased by 44%, and when N is equal to 8, total power consumption is reduced by 20%. Power consumption is reduced by 62% in the suggested design compared to the [5] architecture. As can be shown in Table I, the space savings is more than that of current structures. Table II displays a comparison of the power of non-separable FIR filters of filter orders N = 4 and N = 8.

TABLE. II
COMPARISON OF POWER FOR BLOCK SIZE L = 2 AND 4

| Structure | N | L | Area ($\mu m^2$) | Power (mW) | | |
|---|---|---|---|---|---|---|
| | | | | Static | Dynamic | Total |
| Proposed Non-separable FIR filter | 4 | 2 | 12654 | 0.1985 | 1.1254 | 1.3239 |
| | | 4 | 32598 | 0.2985 | 2.9856 | 3.2841 |
| | 8 | 2 | 21478 | 0.2014 | 2.0146 | 2.2130 |
| | | 4 | 40215 | 0.4215 | 3.9547 | 4.3762 |

Using an RTL compiler synthesis tool, we compare the suggested design using MCLA adders for the multiplier and adder block against the same design using regular CLA adders. Figures 12 and 13 provide visual comparisons of power, area, and delay between the proposed design and MCLA and CLA, respectively.

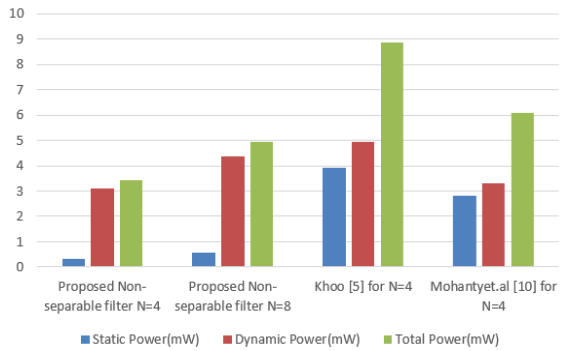


Fig.11. Graphical Comparison of power between proposed and existing structures.

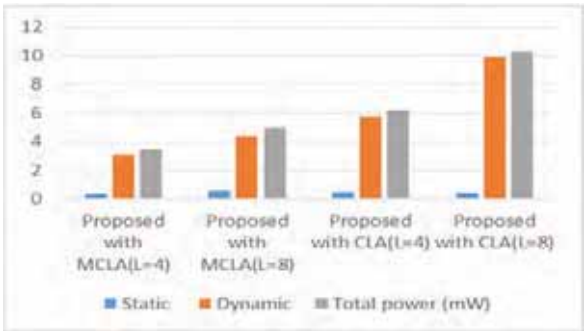


Fig.12. Comparison graph of power for the proposed design with MCLA and with CLA

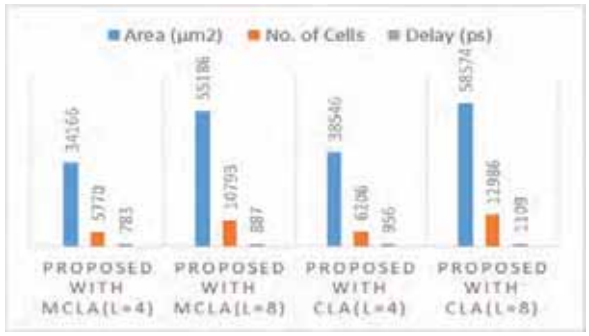


Fig.13. Comparison graph of area and delay for the proposed design with MCLA and with CLA for N = 8

## V. CONCLUSION

An organized approach is taken to realize a 2D FIR non-separable filter architecture with minimal power, area, and delay memory consumption. By employing a block-based model for output assessment in parallel, we can boost throughput by a factor of L. Memory reuse in a completely direct-form structure helps minimize the amount of storage space needed. Parallel prefix adders and low-power BZ-FMD multipliers are used to realize the suggested filter architecture. In order to create a parallel prefix adder, the MCLA addition idea is utilized. Reduced power consumption and a shorter critical path are achieved by the use of parallel processing and pipelining techniques in the filter's final addition. The non-separable 2D FIR filter using MCLA achieves superior power and area results compared to the state-of-the-art FIR architectures. The designs are developed and synthesized in RTL Compiler tools from the TSMC 45nm CMOS library, with the input image size set at 512 by 512, the input block size set at L = 2 and 4, and the filter length set at N = 4 and 8. The suggested MCLA adders in the 2D FIR filter structure resulted in a 50% reduction in power consumption and a 20% reduction in delay compared to the standard 2D FIR filter. The experimental findings demonstrate that the suggested design outperforms the state-of-the-art in terms of area, latency, and power efficiency in the context of memory-efficient architectures.

## REFERENCES

[1] H. Mohammadzade, L. T. Bruton, "A simultaneous div-curl 2D Clifford Fourier Transform filter for enhancing vortices, sinks, and sources in sampled 2D vector field images," in Proc. IEEE International Symposium on Circuits and Systems, May. 2007, pp. 821- 824.

[2] T. Barbu, "Gabor filter based face recognition technique," in *Proc. Rmanian Acad. Ser. A*, 2010, vol. 11, no. 3/2010, pp. 277–283.

[3] S. E. Grigorescu, N. Petkov, and P. Kruizinga, "Comparison of texture features based on Gabor filters," *IEEE Trans. Image Process.*, vol. 11, no. 10, pp. 1160–1167, Oct. 2002.

[4] P. Y. Chen, l. D. Van, H. C. Reddy, and C. T. Lin, "A new VLSI 2-D four-fold-rotational-symmetry filter architecture design", in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS), May 2009, pp. 93-96.

[5] I. H. Khoo, H. C. Reddy, L. D. Van, and C. T. Lin, "Generalized formulation of 2-D filter structures without global broadcast for VLSI implementation", in Proc., *IEEE MWSCAS, Seattle, WA, USA*, Aug. 2010, pp. 426-529.

[6] L. D. Van, "A new 2-D systolic digital filters architecture without-global broadcast", IEEE Trans. Very Large Scale Integr. Syst., vol. 10, no. 4, pp. 477-486, Aug. 2002.

[7] Van, Lan-Da, et al. "A new VLSI architecture without global broadcast for 2-D digital filters." 2000 *IEEE International Symposium on Circuits and Systems. Emerging Technologies for the 21st Century. Proceedings (IEEE Cat No. 00CH36353)*. Vol. 1. IEEE, 2000.

[8] Mohanty, B. K., and P. K. Meher. "High throughput and low-latency implementation of a bit-level systolic architecture for 1D and 2D digital filters." *IEE Proceedings-Computers and Digital Techniques* 146.2 (1999): 91-99.

[9] B. K. Mohanty and P.K. Meher, "A high-performance FIR Filter Architecture for Fixed and Reconfigurable Applications", IEEE Trans. On VLSI Systems, vol. 24, issue 2, pp. 444-452, 2016.

[10] B. K. Mohanty and P.K. Meher, and A. Amira, "Memory Footprint Reduction for Power-Efficient Realization of 2-D Finite Impulse Response Filters", in *IEEE Trans Circuits Syst. I*, vol. 61, no. 1, Jan. 2014.

[11] Odugu, Venkata Krishna, C. Venkata Narasimhulu, and K. Satya Prasad. "Design and implementation of low complexity circularly symmetric 2D FIR filter architectures. " *Multidimensional Systems and Signal Processing* 31 (2020): 1385-1410.

[12] Odugu, Venkata Krishna, and Satya Prasad K. "An efficient VLSI architecture of 2-D finite impulse response filter using enhanced approximate compressor circuits." *International Journal of Circuit Theory and Applications* 49.11 (2021): 3653-3668.

[13] Odugu, Venkata Krishna, C. Venkata Narasimhulu, and K. Satya Prasad. "Implementation of Low Power Generic 2D FIR Filter Bank Architecture Using Memory-based Multipliers." *Journal of Mobile Multimedia* (2022): 583-602.

[14] Odugu, Venkata Krishna, C. Venkata Narasimhulu, and K. Satya Prasad. "A novel filter-bank architecture of 2D-FIR symmetry filters using LUT based multipliers." *Integration* 84 (2022): 12-25.

[15] M. Mottaghi-Dastjerdi, A. Afzali-Kusha, and M. Pedram, BZ-FAD: A low-power low area multiplier based on shift-and-add architecture, IEEE Trans. Very Large Scale Integration (VLSI) Systems. (2009) 302–306.

[16] Pinto, Rohan, and Kumara Shama. "Low-Power Modified Shift-Add Multiplier Design Using Parallel Prefix Adder." *Journal of Circuits, Systems and Computers* 28.02 (2019): 1950019.

[17] Poornima N and V S KanchanaBhaaskaran, Area efficient hybrid parallel prefix adders, J. Procedia Materials Science. 10 (2015) 371–380.

[18] A. P. Vinod and E.M. Lai, "Low power and high-speed implementation of FIR filters for software defined radio receivers" IEEE Trans. Wireless Commun., vol. 7, no. 5, pp. 1669-1675, Jul. 2006.

[19] O. Venkata Krishna, C. Venkata Narasimhulu and K. Satya Prasad "Design and Implementation of Block Based Transpose Form FIR Filter" in IJCA, Issue 8 Volume 1, Jan-Feb. 2018.