Design and Implementation of an Efficient Distributed Arithmetic-Based FIR Filter Using Shared MAC Architecture

Kolluru Anuhya
Department of ECE
GMR Institute of Technology
Rajam, India
22341A0483@gmrit.edu.in

Balla Aparna
Department of ECE
GMR Institute of Technology
Rajam, India
22341A0418@gmrit.edu.in

Pavuluri Prasanth Reddy
Department of ECE
GMR Institute of Technology
Rajam, India
22341A04D7@gmrit.edu.in

Singuru Bhargava Ram
Department of ECE
GMR Institute of Technology
Rajam, India
22341A04G8@gmrit.edu.in

Dr. Yogesh Misra*
Department of ECE
GMR Institute of Technology
Rajam, India (Corresponding Author)
yogesh.m@gmrit.edu.in

***Abstract*— In Digital Signal Processing (DSP) applications, Finite Impulse Response (FIR) filters are crucial. However, traditional multiplier-based FIR filter architecture involves a large number of arithmetic blocks, resulting in increased hardware complexity and dynamic power consumption as the filter order is increased. To overcome the drawbacks of traditional FIR filter architecture, this work presents Distributed Arithmetic (DA) technique. Here, multiplier is avoided by replacing the multiplication operation with a lookup table-based operation and shift accumulate operation. In the proposed architecture, the shared MAC architecture is used in FIR filter implementation, resulting in improved efficiency with regard to hardware complexity and dynamic power consumption. The design is implemented for various tap lengths, and the performance is compared with the traditional multiplier-based FIR filter. Observations indicate that the proposed architecture is efficient regarding logic resource utilization and dynamic power consumption.**

Keywords*— *Digital Signal Processing, Finite Impulse Response (FIR) Filter, Shared MAC Architecture, Distributed Arithmetic (DA), Hardware Efficient Design, Multiplier-less Implementation

I. INTRODUCTION

Finite Impulse Response (FIR) filters represent one of the frequently utilized blocks in signal processing such as wireless communication systems. These filters are utilized due to the important characteristics that are being offered by them, including stability, simplicity, and accurate linear phase characteristics. This filter offers stability, as feedback does not exist, which causes instability in the filter. Another important characteristic of the filter is that it can be implemented with ease in both hardware and software platforms [1], [2]. FIR filters operate by performing a convolution operation between signal samples and filter coefficients. FIR filters are implemented by using a structure that includes a tap delay line, coefficient multipliers, and an accumulation stage. In the conventional FIR filter, a multiplier is utilized for each filter to perform multiplication operations between signal samples and filter coefficients. Although the parallel architectures using multipliers result in high-throughput designs along with a simple design methodology, the large number of multipliers results in increased hardware complexity, silicon area, and power consumption [3], [4]. These problems become more critical for high-order filter designs where it leads to a proportional increase in arithmetic operations along with hardware resources [5]. In order to overcome the above problems, various techniques have been proposed to implement FIR filter

efficiently. Among the various optimization techniques proposed, Distributed Arithmetic (DA) has become a widely used technique for realization of multiplier-less filter architecture [6], [7]. Conventional multiplication operation is replaced with lookup table (LUT) based partial product generation and then by shift and add process. The sum of products computation of FIR filter designs is performed using the bit-level representation of the input data.

This property makes the use of DA-based architectures particularly attractive for the implementation of FPGA and VLSI systems, as the use of lookup tables and shift operations can be efficiently implemented using the programmable logic resources of these platforms [8]. Despite the advantages of DA technique, the traditional DA-based filter architectures have some limitations in terms of increased memory requirements and scalability issues for higher order filters. The increase in tap size, leads to the high requirement of look up tables. This results in inefficient use of the resources for implementing the FIR filters. Therefore, there have been some research activities towards the development of more efficient DA-based FIR filter architectures that result in reduced memory requirements and efficient use of the resources for implementing the FIR filters [9]. Another efficient way to boost the hardware efficiency is to utilize the resource sharing technique, which allows the execution of an arithmetic operation by sharing the arithmetic unit for the execution of multiple arithmetic operations. The above-mentioned technique can be implemented by sharing the Multiply and Accumulate (MAC) architecture for the execution of multiple operations. The sharing of the MAC operations allows the minimization of the arithmetic units for the execution of the calculations. Not only does the above-mentioned technique improve the hardware efficiency, but it also helps in improving the energy efficiency by avoiding the switching activities in the circuit. Keeping the above points in view, the current work focuses on Distributed Arithmetic-based architecture using a shared MAC architecture. The proposed architecture is free from the use of any hardware multipliers, as distributed arithmetic-based LUT computations are used. The Shared MAC architecture enables the efficient reuse of arithmetic resources. The proposed architecture is capable of reducing the total logic utilization and dynamic power dissipation in the design. The proposed architecture is validated for various order FIR filters and is compared with the traditional parallel FIR architecture.

II. METHODOLOGY

A. *FIR Filter Mathematical Model*

An FIR filter generates an output signal by multiplying the input signal with a set of filter coefficients. The operation of FIR filter with N taps is mathematically expressed as $y[n] = \sum_{k=0}^{N-1} h[k]x[n-k]$ where x[n] represents the input signal, h[k] denotes filter coefficients, and y[n] represents filtered output signal. The term N refers to tap size. Each coefficient h[k] is multiplied with the delay element of input and that result is accumulated to get the output. In a normal direct-form FIR filter structure, a set of input samples is first delayed using a series of shift registers in a delay line. In normal hardware structure, a set of FIR filters consists of three main components: a delay line, a set of multipliers, and an adder tree. Even though this structure is easy to design, the inclusion of a multiplier increases hardware complexity, power consumption, and area in a system, especially when designing a high-order filter. As the number of filter taps increases, so does the number of multipliers in a system, thereby increasing hardware consumption in a system [10], [11]. Distributed Arithmetic (DA) offers an efficient method to realize the FIR filtering operation by replacing the multiplication operation with the use of a lookup table-based computation. Rather than directly computing the multiplication operation, the proposed technique of distributed arithmetic separates the input signal into bits and calculates the sum of the bits [12], [13].

B. *Distributed Arithmetic Based Computation*

Distributed Arithmetic is a highly efficient technique in the implementation of inner product operation in digital signal processing systems. The basic concept of the implementation of the Distributed Arithmetic method is to replace the multiplication operation with a set of table look-up operation, shifting operation, and addition operation. Distributed arithmetic reduces the complexity of the hardware in the implementation of FIR filter operation because there is no multiplication operation involved [11], [12]. From the FIR filter equation, the input samples x[n-k] can be represented in their binary form. Assuming each input is represented using B bits, the binary expansion of x[n-k] can be expressed as $x[n-k] = \sum_{i=0}^{B-1} b_{k,i}2^i$ where $b_{k,i}$ denotes the i^{th} bit of the input sample x[n-k], and B represents the input word length. Substituting this equation into the FIR filter equation gives $y[n] = \sum_{k=0}^{N-1} h[k](\sum_{i=0}^{B-1} b_{k,i}2^i)$. By rearranging the terms, the equation becomes $y[n] = \sum_{i=0}^{B-1} 2^i(\sum_{k=0}^{N-1} b_{k,i}h[k])$. From this expression it can be observed that the inner summation depends only on the coefficient values and the corresponding bit values of the

input samples. Since, the bit values $b_{k,i}$ can take either 0 or 1, the possible combinations of coefficient sums can be precomputed and stored in a look-up table. During operation, the LUT output corresponding to the current bit combination is retrieved and accumulated with appropriate shifting. In this regard, the distributed arithmetic method converts the process of multiplication and accumulation into the following simpler forms:

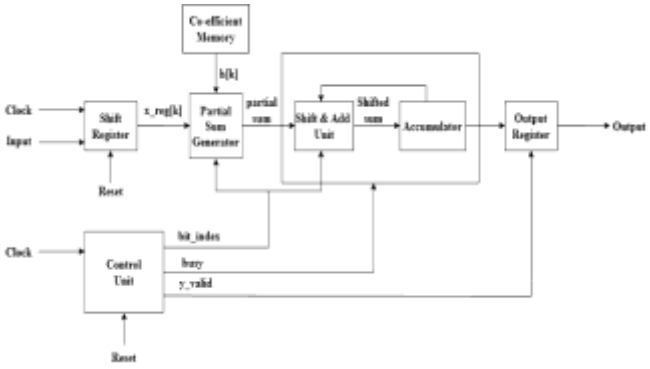
1. Generation of partial sums using the combination of coefficients.

2. Bit-wise shifting of the corresponding partial sums according to the binary weight 2^i

3. Accumulation of the shifted partial sums.

These forms are the basis for the proposed design [11], [14]. In the implemented design, the partial sum generator is responsible for the generation of the required coefficient combinations, whereas the shift unit is responsible for scaling the partial sums according to the bit index.

C. *Proposed DA-Based FIR Filter*



The proposed design shown in Fig. 1 eliminates the need to use traditional multipliers and performs the filtering operation by using distributed arithmetic techniques based on a combination of a look-up table-based partial sum generation, shifting operations, and accumulation. The architecture mainly comprises a shift register, coefficient memory, partial sum generator, shifting and addition block, accumulator block, output register, and control block. The input samples x[n] entering the FIR filter are first sent to a shift register that acts as a tap delay line to the FIR filter. In every clock cycle, the new input sample enters the first stage of the shift register while the previously stored input samples move to the subsequent stages of the shift register. This provides delayed versions of the input signal represented as x[n-k], with k indicating the tap number. The shift register provides the required input samples to the distributed arithmetic block. The coefficient memory stores the filter coefficients h[k] that are used during the process of computing the partial sums. The coefficients remain constant during the entire process of filtering. The coefficients are provided to the partial sum generator to perform the process of distributed arithmetic. The partial sum generator is the main component of the distributed arithmetic computation process. Unlike the direct multiplication of the input samples and the coefficients, the partial sum generator works on binary values of the input stored in shift register to generate the coefficient combinations. To achieve this, the bits of all the taps are examined for the specific bit position of the input data and the required coefficient values are summed to generate the partial sum. The generated partial sum is then passed through the shift and add unit, which carries out the scaling operation based on the binary weight of the processed bit. According to the distributed arithmetic formulation, each partial sum must be multiplied by 2^i where i represents the bit position. Instead of a multiplier, this scaling operation is carried out through a series of shift operations followed by addition. The shifted results are accumulated using the accumulator block, which forms the accumulation stage of the multiply-accumulate operation. Since the same accumulation hardware is reused for processing all bit positions sequentially, the architecture effectively implements a shared MAC structure. This reuse of hardware significantly reduces the number of required computational resources and improves the area efficiency of the design. The final accumulated result is stored in the output register, which provides the filtered output signal y[n]. The output register ensures that the computed result is synchronized with the system clock before being delivered to the output stage. The overall operation of the

Fig. 1. Block Diagram of proposed Filter

architecture is managed by the control unit, which produces the control signals necessary to manage the sequential distributed arithmetic computation process. These control signals are bit_index, busy, and y_valid, among others. The bit_index signal is generated to indicate the current bit position being processed, and the busy signal is generated to indicate whether the filter is busy with the computation process. Once all the bit positions have been processed, the control unit sets the y_valid signal to indicate that the data is available. Through the proposed distributed arithmetic computation and the sharing of the MAC operations, the proposed architecture offers an efficient implementation of the FIR filtering operation.

D. *Control Unit Operation*

The sequential process of the filter is controlled by a control unit, which handles data flow to the processing blocks and ensures proper control of the computation

processes carried out in the filter. As shown in Fig. 1, the control unit sends control signals such as bit_index, busy, and y_valid in order to control the process of the proposed distributed arithmetic-based computation and the output sample. At the beginning of the proposed process, when the system reset signal is asserted, all registers and counters in the filter are set to their default values. Specifically, the busy signal is set to zero, and the bit_index counter and output valid signal are set to zero. After this, a new process is initiated by the control unit in order to compute the input samples stored in the shift register. In this context, the signal bit_index acts as a counter and represents the current bit index in the proposed process of the distributed arithmetic-based computation. Considering that each input sample is represented with a total of B bits, this process is carried out sequentially for each bit index ranging from i = 0 to i = B - 1. During the execution of every clock cycle, the control unit increments the bit index as well as enables the partial sum generator to calculate the coefficient combinations for the current bit position.

During the execution of the distributed arithmetic operation, the control unit sets the busy signal to show that the architecture is busy carrying out the operation. The busy signal is used to show that the shift and add unit as well as the accumulator are always busy processing the calculated partial sums during the execution of the distributed arithmetic operation. This is to show that the shared MAC unit is busy. The control unit always checks the value of the bit index counter to ascertain whether the distributed arithmetic operation is complete. Where the counter is located in the last bit position, i.e., in position B-1, the sum of the counter is the last output of the FIR filter. It is in this position where the control unit sends a signal referred to as y valid to indicate that the output stored in the output register is ready to be used in the next stage of processing. In a situation where the output is generated by the architecture, it is in this position where the control unit resets a signal referred to as busy and another counter referred to as bit index counter in order to prepare it for the next stage of computation. It is in this way that the algorithm of distributed arithmetic is implemented by the architecture sequentially.

E. Hardware Operation Flow

The distributed arithmetic-based FIR filter that has been proposed is intended to be used sequentially. The process of filtering takes place after a series of clock pulses. The flow of the hardware operation can be explained by understanding the sequence of operations that take place during the implementation of the proposed filter. At the beginning of the operation, the input sample is made available during the application of the clock pulse. The shift register is used as a tap delay line to store the current input sample and the past input samples to perform the FIR filter operation. Clock is applied and the input samples are available in the shift register, the process of distributed arithmetic for the calculation of the output begins. The control unit initiates the process by loading the bit index counter and the busy signal to start the process of calculation. It checks the bits of the input samples through the partial sum generator by using the bits of the input samples based on the current bit position given by the bit index counter. Then it generates the sum of a suitable combination of filter coefficients to generate the partial sum. This generated partial sum goes through the shift and add process. In this process, the generated partial sum gets multiplied by a binary weight based on the current bit position. To achieve this, the generated partial sum gets multiplied by 2^i. However, it is more efficient to achieve this by shifting the binary number instead of the multiplication operation. Then it sends the shifted binary number to the accumulator. The accumulator adds the scaled partial sum to the previously accumulated value. Because the calculation carried out by the distributed arithmetic follows the pattern of bits within the input data, the accumulation operation is repeated for every bit within the input data. This repetitive nature of the accumulation operation essentially performs the multiply-accumulate operation that is part of the FIR filter equation. As the calculation progresses through the bits of the input data, the control unit continuously increments the bit index with every clock cycle. Once the last bit index is reached, the accumulated value represents the complete output of the FIR filtering operation. At this point, the control unit asserts the y_valid signal to indicate that the output register holds valid output data. The output register holds the output value computed by the calculation. Once the calculation is complete, the busy signal is reset. With this method, the proposed design effectively performs. the FIR filter process with a minimal amount of hardware through the sharing of arithmetic units across multiple clock cycles.

III. RESULTS

The proposed design using the shared MAC architecture has been implemented in Verilog HDL. It has been synthesized by making use of the Xilinx Vivado design tool. This has been achieved in order to test the hardware efficiency regarding hardware resources as well as power consumption. To test the performance of the design, 16 tap filter is implemented. This design makes use of distributed arithmetic in order to remove the multipliers. This has been achieved by making use of the shared MAC architecture. By observing the resource utilization report in the Fig. 2, the filter architecture is making use of only a small fraction of the total resources that are available in the Field Programmable Gate Array (FPGA). Total resources used in terms of slice LUTs and flip-flops in the design are very

Resource	Utilization	Available	Utilization %
LUT	168	53200	0.32
FF	327	106400	0.31
IO	51	200	25.50

Fig. 2. FPGA Resource Utilization of the proposed DA-based FIR filter using Shared MAC Architecture

low in comparison to the total resources that are available in the device. This indicates that the proposed design is very good in regard to hardware utilization. It is observed that the low utilization of LUTs is achieved by the distributed arithmetic used in the design.

This replaces the normal multiplication operation by making use of LUTs in the generation of partial sums. Also, the shared structure of the MAC operation reduces the requirement for additional computational resources. The power consumption characteristics of the proposed architecture were analyzed by using the power estimation report, which is obtained after implementing the architecture in Vivado. In the design, logic element contributes the highest percentage of dynamic power, i.e., 34%. The contribution of signal transitions is found to be around 33%, and the contribution of clock networks is found to be around 21%. The remaining percentage of dynamic power is found to be contributed by I/O resources, in which I/O resources contribute around 12% of the dynamic power. The relatively low power consumption in terms of dynamic power can be correlated with the use of distributed arithmetic. The use of distributed arithmetic eliminates the use of traditional multipliers. Instead, traditional hardware multiplication is replaced by LUT-based partial sum generation, as well as shift and accumulate techniques, in the proposed architecture. This causes a relatively low switching activity in the arithmetic circuitry. Furthermore, the use of shared MAC architecture enables the reuse of arithmetic circuitry in implementing multiple computation cycles in the design. This causes a relatively low number of active computational circuitry at any given time. From the above analysis, it is evident that power is efficiently used in the proposed design.

Table. 1. Scalability Analysis of DA-Based FIR Filter

To further evaluate efficiency of above architecture, filters of different numbers of filter taps have been implemented and tested. Specifically, the implementation of 16 tap filters, 32 tap filters, and 64 tap filters using the same architecture has been performed. The purpose of the test was to examine the change in hardware utilization and power as the tap size increases. Table 1 shows the obtained results.

Filter Taps	LUTs	DSP Blocks	Dynamic Power (Watts)	Total Power (Watts)
16	181	0	0.010	0.114
32	384	0	0.010	0.114
64	539	0	0.014	0.119

From Table 1, it is clear that hardware utilization increases gradually with tap size. This is expected because, by increasing the tap size, we must also increase storage elements as well as arithmetic operations. However, it is clear that hardware utilization is moderate due to the use of distributed arithmetic as well as shared MAC architecture. It is also clear that there is no utilization of DSP blocks even when implementing high-order filters. This is another way of confirming that the use of the distributed arithmetic technique is effective in eliminating the use of hardware multipliers, as well as DSP blocks. Regarding power consumption, it is clear that, as the order of the filter increases, there is a slight increase in power consumption. This is an indicator that power consumption is stable even when implementing high-order filters. Also, the timing constraints are satisfied, indicating that the design is effective even when the order of filter is high. These results demonstrate that the proposed architecture maintains efficient hardware utilization and stable power characteristics even when the filter order increases.

IV. CONCLUSION

The DA-based filter using the shared MAC architecture was introduced in this paper. The proposed approach eliminates the hardware multipliers and utilizes the distributed arithmetic approach for the realization of the FIR filter using the partial sum generation and the shift and accumulate operation. The proposed design efficiently

utilizes the arithmetic resources during the sequential processing operation using the distributed arithmetic approach and the shared MAC architecture. The design was implemented using the Verilog HDL language, and the results were obtained using the synthesis and implementation reports generated using the Xilinx Vivado design tool. The results obtained from the design indicate that the proposed design utilizes only a small portion of the available hardware resources in the FPGA device, eliminating the multiplier and the DSP blocks. The analysis of the proposed architecture in terms of scalability is done using various tap configurations. The proposed architecture is able to support higher-order filters with a moderate increase in logic resources. Also, the analysis of the power performed in the proposed work shows that the proposed architecture is able to maintain low dynamic power with minimal switching activities and arithmetic block reuse. Hence, the proposed architecture using the application of the distributed arithmetic concept is efficient in digital signal processing systems. The proposed design provides efficient trade-off regarding hardware and power.

V. ACKNOWLEDGEMENT

The lab setup to develop the proposed work was carried out in the lab of Department of Electronics and Communication Engineering of GMR Institute of technology, Rajam, India. The Authors are thankful to the management for providing the support.

REFERENCES

[1] S. Meher, J. Valls, T. B. Juang, K. Sridharan, and K. Maharatna, "50 years of FIR filter design and implementation," IEEE Circuits and Systems Magazine, vol. 13, no. 4, pp. 21–45, 2013.

[2] K. K. Parhi, "VLSI architectures for digital signal processing systems," IEEE Signal Processing Magazine, vol. 27, no. 2, pp. 103–110, 2010.

[3] R. Mahesh and A. P. Vinod, "New reconfigurable architectures for implementing FIR filters with low complexity," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 29, no. 2, pp. 275–288, Feb. 2010.

[4] J. Liang and J. Han, "Efficient FPGA implementation of FIR filters using distributed arithmetic," IET Circuits, Devices & Systems, vol. 7, no. 5, pp. 270–277, 2013.

[5] M. Garrido, "Efficient architectures for FIR filtering in FPGA-based systems," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 22, no. 2, pp. 412–421, 2014.

[6] S. Meher, "Area-efficient and high-speed FIR filter architecture using distributed arithmetic," IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 60, no. 10, pp. 673–677, Oct. 2013.

[7] H. Samueli, "An improved search algorithm for the design of multiplierless FIR filters with powers-of-two coefficients," IEEE Transactions on Circuits and Systems, vol. 36, no. 7, pp. 1044–1047, 2012.

[8] S. Chen and J. Cong, "Low-power FIR filter design for FPGA-based DSP applications," ACM Transactions on Reconfigurable Technology and Systems, vol. 8, no. 3, pp. 1–21, 2015.

[9] A. K. Pathakamuri et al., "Optimal realization of distributed arithmetic-based MAC adaptive FIR filter architecture," Electronics (MDPI), vol. 13, no. 17, pp. 3551, 2024.

[10] O. Gustafsson, "Low-complexity multiplierless FIR filter structures," IEEE Transactions on Circuits and Systems II, vol. 63, no. 11, pp. 1070–1074, 2016.

[11] R. Sharma and P. K. Meher, "Efficient hardware architectures for digital filters and signal processing applications," IEEE Transactions on Very Large Scale Integration Systems, vol. 30, no. 4, pp. 513–522, 2022.

[12] H. Yoo and D. V. Anderson, "Hardware-efficient distributed arithmetic based digital filters," IEEE Transactions on Consumer Electronics, vol. 63, no. 4, pp. 439–447, 2017.

[13] P. S. Kumar and M. K. Mandal, "Energy-efficient FIR filter implementation using distributed arithmetic," IET Signal Processing, vol. 13, no. 4, pp. 410–418, 2019.

[14] S. R. Patel and K. Maharatna, "Recent advances in FPGA-based DSP architectures for digital signal processing systems," IEEE Access, vol. 12, pp. 34521–34539, 2024.