# Low Area and Power-Efficient FPGA Implementation of Improved AM-CSA-IIR Filter Design for the DSP Application

Arunjyothi Eddla and Venkata Yasoda Jayasree Pappu
Department of EECE, GITAM Institute of Technology, Hyderabad, India
Email: {aeddla; jpappu}@gitam.edu

***Abstract*—Digital Signal Processing (DSP) grew enormously in the past few decades and it was extensively practiced for numerous engineering applications like biomedical signal processing, radar signal processing, adaptive antenna design, intelligent control, etc. In DSP, the Finite Impulse Response (FIR) filter and Infinite Impulse Response (IIR) filter plays a vital role in the design of a complex signal processing system. Generally, The FIR filters are stable, linear in phase, with fewer finite-precision errors and easy to implement. However, most of the IIR filters are used in signal processing applications due to the less computational complexity which is compared to the FIR filters. Moreover, the IIR filter requires only fewer filter coefficients and requires only less amount storage registers. In this research, an IIR filter is implemented with Array multiplier and Carry Skip Adder (CSA) to improve hardware utilization. The IIR filter input is generated randomly which is stored in Random Access Memory (RAM). Similarly, the coefficient is generated by utilizing the Parks-McClellan algorithm where the generated value is stored in Read Only Memory (ROM). The proposed AM-CSA IIR (Array Multiplier-Carry Skip Adder Infinite Impulse Response) filter performances are evaluated in terms of filter output, Input Output Block (IOB), Look Up Table (LUT), Flip Flops (FF), slices, delay, and fractional delay. Overall the proposed design attained LUT= 115 (0.0564%), FF=40 (0.01%) and IOB= 22 (3.77%) and reduced fractional delay of about 0.0057ns.**

***Index Terms*—Carry skip adder, digital signal processing, infinite impulse response filter, Parks-Mcclellan algorithm, array multiplier, and AM-CSA IIR**

## I. Introduction

In the field of digital signal processing, digital filters are most commonly used to suppress the harmonic components which rely on less analog circuitry and potentially allow a better Signal to Noise Ratio (SNR). Several digital filters (IIR) are characterized based on the strong potential to achieve different design constraints like area, speed, power consumption and delay. IIR filters are well known to have some unique attributes that improve the filtering process but suffer from non-linear phase responses. In different circumstances, different techniques like phase linearization techniques are evaluated to perform the filter operation but failed to achieve the flawless phase response. In [1] to improve the phase response "Almost Linearization" technique has been implemented in DFT modulated filter where structure characteristics are also majorly concerned. Mostly, IIR filter design, transfer function recognition are mainly concentrated and practiced a structure evolution optimal design [2]. In [3], [4] the authors designed the IIR filter using the Partial Swam Optimization algorithm to offer the advantages of stability, robustness in avoiding local minima problem and noise cancellation. Several case studies have been carried out on digital filters [5] with the acknowledgment of filter coefficients and fix the precision point of input/output. Meanwhile, for designing a Finite Word Length (FWL) IIR filter, a statistical model [6] has been introduced and the filter coefficients are tuned using audio equalization techniques [7].

Further IIR filters used lattice filter structure by owing its advantages, so Vikas Prathak in [8], [9] designed an IIR anti-notch lattice structured filter which decreases the computational complexity and harmonics. Although it suffers from some hindrance like, quantization degradation and truncation due to taps of the filter has certainly caused some performance degradation. To resolve these sufferings an optimal solution in the designed IIR filter is realized in [10], [11] that aids in achieving a stable, optimized, practically robust design. Additionally, the demand for IIR filters in various applications is drastically increased and these latest works [12], [13] attempt to reduce the computational complexity, improved non-linear phase response, and reduced area utilization and power consumption.

Some recent year researches of IIR filters are developed such as distributed arithmetic (DA) based IIR [14], polyphase IIR filter [15] and IIR wavelet filter banks [16]. In DA based IIR filter design, a hardware based LUT is used in the internal blocks which used to minimize the need of adders and memory elements. The filtering process of the polyphase IIR filter is frequently done in 2 stage process for reducing the computational workload. In IIR wavelet filter banks, the minimum level of error is achieved based on the magnitude response-based coefficients.

This paper focuses on design and implementation of AM-CSA IIR filter on different Field Programmable Gate Array (FPGA) platforms. Some of the major contributions of the proposed digital IIR filter are stated below.

Manuscript received December 4, 2021; revised February 18, 2022; accepted March 11, 2022.
Corresponding author: Arunjyothi Eddla (email: aeddla@gitam.edu).



doi: 10.18178/ijeetc.11.4.294-303

- The main objective of this AM-CSA IIR filter is to utilize the lesser hardware resources while minimizing the power of the overall design. The proposed digital IIR filter is applicable for higher order filters and requires a minimum amount of hardware resources, so that the realization of IIR filters have improved for real time systems.
- Due to the less hardware utilization of array multiplier and less latency of the carry skip adder, these modules are integrated in the IIR filter to achieve less utilization of resources. This integration is used to minimize the memory storage which is considered as the key factors in selecting the digital IIR filters.
- Pipelining technique has been introduced by register or latch between each sub unit, so that it reduces the critical path delay and achieves high throughput.
- Another key feature is that, filter coefficients are generated using the Park's McClellan algorithm and the integration tends to increase the speed of the filtering process.
- Further, this AM-CSA IIR filter design reduces the fractional delay output which substantially improves the performance of the design.

The organization of this research paper is detailed as follows: Section II describes the literature review of existing methods, and Section III provides Problem statements along with the solution. In Section IV, the proposed model is explained. The results and comparison of the proposed design model with existing designs are explained in Section V. Finally, Section VI provides the conclusion of this research paper.

## II. Literature Review

There are many existing techniques related to the filter designs in FPGA that were developed for digital signal processing and wireless communication system applications. A brief evaluation of some contributions to the existing literature is given as follows:

Chen [17] presented a structure evolution-based optimization algorithm that allows the frequency response specifications. At first, the structural generation method creates a diverse which valid the digital filter structure with the sequence of instruction. Next, the presented algorithm was evaluated in terms of system accumulative error, reliability that provides better performance when compared to the other existing algorithms. However, the developed optimization algorithm lagged in premature convergence and local search-ability.

Seshadri and Ramakrishnan [18] developed FIR and IIR filters using the era EP4CE115F29C7 FPGA device using Quartus II 13.1 synthesis tool. In the present paper, the conventional Moving Average (MA) FIR filters and the fast MA FIR filters were being exemplified using look-ahead arithmetic. The conventional IIR filters were exemplified using the Combination of Integrator and Comb sections (CIC) method where the fast IIR filter used the look-ahead arithmetic. The developed fast IIR filter provided better performance when compared to the conventional IIR filter. However, the implementation of the IIR filter using fixed-point arithmetic was harder.

Roy [19] presented an IIR filter design implemented on FPGA using the parallel and pipeline process. Here, the filter co-efficient and delay elements are required to implement the IIR filter effectively. The different IIR filter structures are implemented in the Artix FPGA platform that helps to reduce the critical path. The pipeline registers are inserted in the IIR filters to support the higher frequencies using look-ahead techniques. But, these techniques increased the hardware complexity and also inexact the pole-zero cancellation may occur due to finite word length.

Datta and Dutta [20] presented a high-performance IIR filter design that was implemented on FPGA. Here, the developed IIR filter was implemented using the parallel pipeline-based Finite Impulse Response (FIR) filter. The high-speed IIR filter was designed by utilizing the combined gain parallel pipelined two FIR filters of 3-tap to improve the performance of the IIR filter. The developed IIR filter minimized the area that results in energy optimization and enhanced the maximum frequency. Since the maximum clock frequency was limited, the operating speed of the IIR filter was reduced.

Abinaya [21] developed a CIC filter using Xilinx ISE 14.7 designed in Verilog HD land implemented on Kintex7 FPGA. The developed CIC filter contained two blocks as integrator and comb blocks. These two blocks were cascaded using a decimator. When compared with the traditional CIC filter, the designed filter provides better performance in terms of area, power consumption, and speed. Significantly, the area utilization was reduced to a satisfactory extent due to the minimum hardware resources in the multiplier block with low passband drop. Meanwhile, the high degree symmetric polynomial functions were not analyzed in the stopband and passband which became an issue in the performance evaluation.

Eddla and Jayasree Pappu [22] proposed the low area FPGA implementation of FIR filter with optimal combinational blocks. In this work, Interpolated Spectral Parameter Approximation (ISPA) filter was implemented using carry look ahead adder and the Vedic multiplier. This work helped to reduce the 6% of hardware utilization compared to the conventional methods. However, the power consumption of the proposed work was high (49.62mW) which increase the system complexity.

Potsangbam and Kumar [23] presented the integrated pipelining and parallel processing structure for Finite Impulse Response (FIR) and IIR filters. This pipelining and parallel design was used to improve the delay and power consumption. However, the designed IIR filter was consumed high amount of LUTs. Jan Kikkert [24] developed the IIR filter-based phasor measurement units to calculate the phase, frequency, voltage and rate of frequency variation. However, this work was failed to analyse the hardware resources of IIR filter.

Mahabub [25] implemented the higher order IIR filter for obtaining the noise free Electroencephalogram (EEG)

signals. This higher order IIR filter was provided faster response and utilized lesser memory. However, the power usage of the designed IIR filter was high during the filtering process. Iniguez-Lomeli [26] used the IIR filter to pre-process the raw bio medical signals. Faleh Hassan [27] presented genetic algorithm based IIR filter. The direct form II of IIR filter was caused dynamic range over the intersection of delay units, when the pole units precede the zero units.

Pathak [28] developed the Band Pass Notch (BPN) IIR filter to identify the positions of hot spot in any protein family. The retiming was used in the IIR filter to improve the speed. However, this BPN IIR filter was used high amount of registers than the IIR filter without retiming. Ladekar [29] implemented the higher order IIR filter i.e., Coupled All Pass Filter (CAPF) with retiming to process the EEG signals. The partial-fraction expansion was used to realize the parallel form in the filter design. Some of filter structures such as Direct Form-II (DF-II), Cascade, Parallel, Lattice-Ladder and Scattered Lookahead (SLA) structures with retiming are used to analyze the CAPF structure. However, the designed CAPF with retiming architecture was used high amount of resources.

## III. Problem Statement

Current problems associated with the IIR filter are stated in this section and explains how the proposed method overcomes the problems faced by the IIR filter. The issues faced by the IIR filter are mentioned as follows:

In the filter design, an extra delay is added among the partial product adder and partial product generator which causes the delay to be increased for processing the higher-order IIR filter. The frequency of the IIR filter is affected due to the higher delay obtained during the filtering process. Moreover, the area consumed by overall architecture is increased due to the utilization of a high amount of logical elements like LUT, slices, and FF in the filter design.

Motivated by these existing researches, the filter design is improved to solve the problems related to the higher hardware utilization and high delay. The solution given by the proposed research is given as follows:

*Solution*: The integration of the array multiplier and carry skip adder in the IIR filter design helps to reduce the utilization count of logical elements. Hence, the area utilization of the IIR filter is reduced which consequently improves the speed and frequency of the IIR filter. Therefore, the delay during the filtering process is minimized in the IIR filter.

## IV. Proposed Method

In this proposed method, the logical elements are reduced by replacing the multipliers and adders which is present in the IIR filter. Here, the multiplier is replaced by the Array multiplier and the adder is replaced by the CSA. Since the logical elements used in the IIR filter are reduced that certainly minimizes the hardware utilization of the overall proposed design. The architecture of the proposed system is shown in Fig. 1, where $X(n)$ is the input to the proposed IIR filter and $Y(n)$ is the obtained output.

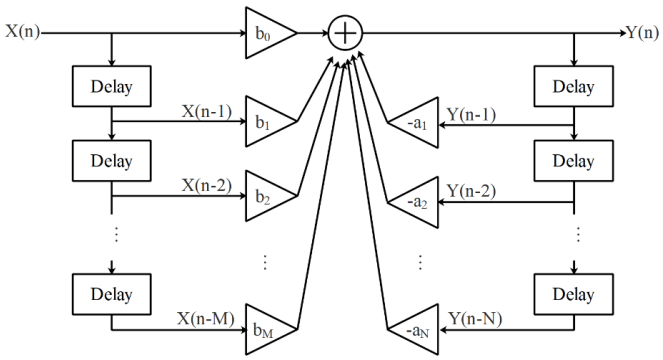


Fig. 1. The architecture of the IIR filter.

The transfer function of the IIR filter is shown in the Eq. (1).

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1+\sum_{k=1}^{N} a_k z^{-k}} \quad (1)$$

where the coefficients of the IIR filter are $a_k$ and $b_k$: $N$ and $M$ denotes the filter order, and $Z^{-1}$ represented as the delay elements and $n$ represented as number of input taps.

The process of the proposed IIR filter is explained as follows.

- Initially, the input data of the IIR filter is generated randomly using the true random number generator and the obtained value is stored in RAM. The coefficients (h) for the filter are generated using the Parks-McClellan algorithm where the values are stored in the ROM.
- For example, the input data of about 2 and the first filter coefficient of about 2 are given as input to the processing element. Here, the multiplication process of the input data and the coefficient takes place with the help of the Array multiplier.

Similarly, the remaining values are performed in the IIR filter where the output is delivered in $Y(n)$.

### A. Conceptual Explanation of AM-CSA IIR Filter

This section briefly discusses the functionality of all the processing elements used in the filter design and also

explains the filtering process which is being carried out at each stage of the filter design. The objective of the IIR filter design is to provide an efficient filtering process with a sharp cut-off frequency response. The proposed IIR filter is an 8 tapping filter that produces the filter output at each instant of the clock signal. Two important signals namely clock and reset signal together forms the control circuitry to control the entire filtering operation. In this 8 tap filter, two input data are given such as input data $X(n)$ and filter coefficient 'h' while these two input values are stored in the memories (i.e. RAM and ROM). The address for each input data and filter coefficients are generated with the help of the address generator and stored in the respective memories which are accessed at the interval of time. However, the input data $X(n)$ is 8-bit and it is generated with the help of True Random Number Generator (TRNG) which uses $ random function. Then the next input (i.e. filter coefficient) is generated using the most popularly used algorithm Parks McCellan algorithm. At every clock instant, the data's in the memory are called using the respective address for further process.

### B. Parks McCle Llan Algorithm

For the filter design in the field of signal processing, the most promisingly used algorithm is the Parks McClellan algorithm [30]. It was originally designed by James McClellan and Thomas Parkin in 1972 and written in Fortran (Programming Language). It is an iterative algorithm (runs for about mentioned values) and uses an indirect method to find the filter coefficients and labelled as equiripple, Chebyshev and Remez algorithm. In the IIR filter design, filter coefficients are stored in ROM where those filter coefficients are found out using the Parks McClellan algorithm [21]. By utilizing the Chebyshev approximation, the algorithm reduces the error in pass and stopband. Some basic steps are carried out in the Parks McClellan algorithm to identify the filter coefficients are stated below,

Step 1: To initialize and select an extrema set of frequency $\{w_i(0)\}$.

Step 2: Calculate the Chebyshev approximation on the chosen extrema set of frequency and then for the present extrema set to compute the min-max error value ($\delta(m)$).

Step 3: The error function is computed by performing the polynomial interpolation on the entire set of frequencies.

Step 4: On the entire set frequencies, go search for the local extrema $|E(m)(\omega)|$.

Step 5: If $\max(\omega \varepsilon \Omega)|E(m)(\omega)| > \delta(m)$, update the extrema set to $\{\omega_i(m+1)\}$ by taking new frequencies, where $|E(m)\omega|$ has its local maxima and make sure that the error alternates on the ordered set of frequencies as in step 4 and 5. If the alteration is not satisfied, then return to step 2 and repeat the process.

Step 6: If $\max(\omega \varepsilon \Omega)|E(m)\omega| \leq \delta(m)$ then the alteration theorem is satisfied, inverse discrete Fourier transform which is computed using set $\{w_i(0)\}$ and interpolation is done to obtain the filter coefficients.

Using the above Parks Mcclellan algorithm steps the filter coefficients $(h_0, h_1, h_2, h_3, h_4, h_5, h_6 \text{ and } h_7)$ are calculated and then each filter coefficient is processed with the given input $X(n)$ are explained in the next section.

### C. Array Multiplier

Recently, in today's world, the utilization of the multiplier block in the DSP application becomes quite important. In every filter design, a multiplier is used to enhance the speed and throughput of the operation. Due to the current technologies, the design targets like high speed, accuracy and low power consumption are mainly concentrated and multipliers are designed by satisfying the targets. In this IIR filter design, an array multiplier is used to improve the speed of the operation, reduces the complexity of the design with its unique regular structure and reduce the power usage.

#### 1) Design of 4-bit array multiplier

An array multiplier is a digital combinational circuit used for the multiplication process where it deploys add and shift algorithms. Basically array multiplier forms an array-like structure that includes Full Adder (FA) and Half Adder (HA). The block diagram of the array multiplier is shown in Fig. 2. The multiplication process behind the array multiplier is,

- Partial product generation
- Partial product reduction
- Final addition

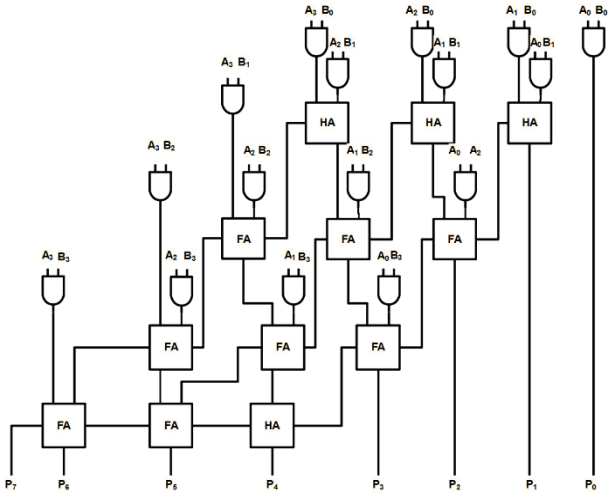


Fig. 2. Block diagram of 4×4 array multiplier.

Let $A$ and $B$ be the two $n$-bit numbers used for the multiplication process, where $A$ is the multiplicand and $B$ is the multiplier. We consider $A$ and $B$ as a 4 bit inputs which performs the array multiplier. It produces $P$ product output which is of 2$n$ bit shown in the below equation, for example, if $n$=4 it produces 2×4=8 bit product output which is presented in Fig. 3:

$$P(\text{Product term}) = A(\text{multiplicand}) \times B(\text{multiplier})$$

The mathematical expression of getting array multiplier product terms are given as below:

$$P_0 = A_0 B_0 \tag{2}$$

$$P_1 = A_1 B_0 + A_0 B_1 \tag{3}$$

$$P_2 = A_2B_0 + A_1B_1 + A_0B_2 \quad (4)$$

$$P_3 = A_3B_0 + A_2B_1 + A_1B_2 + A_0B_3 \quad (5)$$

$$P_4 = A_3B_1 + A_2B_2 + A_1B_3 \quad (6)$$

$$P_5 = A_3B_2 + A_2B_3 \quad (7)$$

$$P_6 = B_3 \quad (8)$$

$$P_7 = A_3 \quad (9)$$

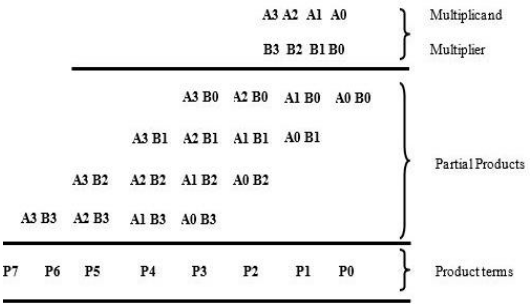


Fig. 3. 4-bit multiplication process.

*2) Partial product generation*

In the partial product generation stage, the partial products are generated for the required 4-bit numbers. In this multiplier, the partial products are generated using the AND gates. Fundamentally, each bit of the multiplier *B* is ANDed with each bit of the multiplicand *A*, i.e., if any one of the input is logic 0 then the output is logic 0 and it produces logic 1 output only if both the inputs are said to be logic 1. Using this principle, the partial products are generated and it requires ($A \times B$) AND gates, i.e., for 4-bit multiplication it requires 16 AND gates.

*3) Partial product reduction*

After the partial products are generated in the previous stage, it goes to the partial product reduction stage. Based on the reduction stage, several tree multipliers are classified and in this filter design, an array multiplier is used. It forms the array structure of full adder and half adder for employing simultaneous addition of the various partial products. The generated 18 partials are reduced using the full adder (for adding 3 partial products) and half adder (for adding 2 partial products). Overall it requires ($A$–1)$B$ bit adders, for 4-bit multiplication it requires 12 adders.

*4) Final addition*

Once the partial products are reduced, the final addition process is carried out to produce the product terms. The final addition process is carried out with the reduced product terms obtained in step 2. Here, based on the output requirement the final parallel adder is chosen (i.e.) for 4-bit multiplication 8-bit parallel adder is used to increase the speed of the operation. In this multiplier, all the stages are sequentially executed and the addition process is parallel performed to produce the final product term output of 8-bit $(P_0, P_1, P_2, P_3, P_4, P_5, P_6 \text{ and } P_7)$

In the IIR filter design, 8-bit input data $X(n)$ $(X_0, X_1, X_2, X_3, X_4, X_5, X_6 \text{ and } X_7)$ and each filter coefficient $(h_0, h_1, h_2, h_3, h_4, h_5, h_6 \text{ and } h_7)$ are multiplied individually using the operation of array multiplier and passed to the next processing stage, i.e. adder block.

*D. Adder*

Adders are considered to be the most important building block in various applications. Adders are effectively designed as it performs a series recursive addition operation which improves the speed of the overall operation. Several parallel adders are introduced like Ripple Carry Adder (RCA) and Carry Look Ahead adder (CLA), but these adders failed to improve the speed of the operation due to the less propagation delay of the carry. If the carry propagation is delayed it simultaneously reduces the speed of the operation, so these problems are overcome with the introduction of high-speed adders. High-speed adders are considered to be more effective based on their unique attribute to perform the addition operation of binary numbers in 1's, 2's complement, and sign-magnitude formats. In the IIR filter design, to perform addition operation one of the well-known high speed adder is used namely Carry Skip Adder (CSA).

*1) Carry skip adder*

Carry skip adder is also named as Carry by-pass adder which consists of special circuitry to improve the speed and reducing the carry propagation delay. In this adder, carry skip circuitry (special circuitry) is implemented based on the carry skip mechanism. This circuitry contains two main blocks namely, AND block and multiplexer block which are together defined as block propagate blocks. Fig. 4. Shows the implementation of a 4-bit carry skip adder and Fig. 5. Shows the logical architecture of full adder.

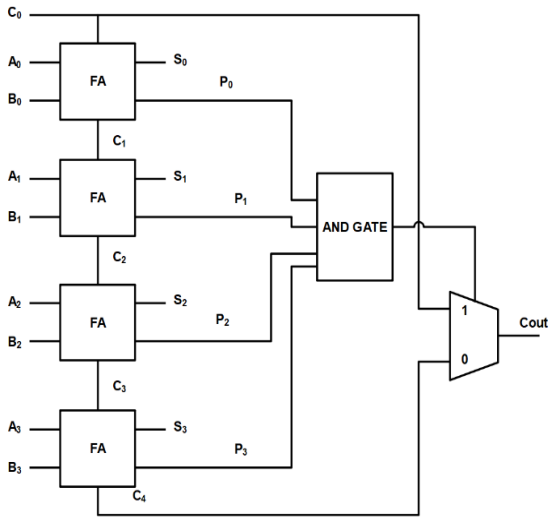


Fig. 4. Block diagram of 4-bit Carry Skip Adder.

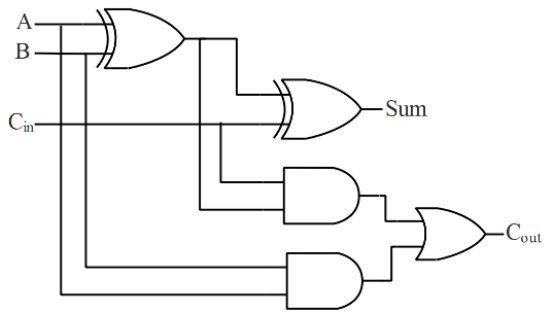


Fig. 5. Logic circuit of full adder.

*2) Carry skip mechanism*

In Fig. 4, 4-bit RCA with carry skip circuit is integrated to form the CSA circuit. The 4-bit RCA contains 4 full adder blocks and generates 4 propagating signals $P_0$, $P_1$, $P_2$, and $P_3$, based on the 2 inputs of each FA $(A_0, B_0, A_1, B_1, A_2, B_2, A_3, B_3)$, which is defined as

$$P_i = A_i \text{ XOR } B_i \tag{10}$$

If $A_i$ is not equal to $B_i$, then the propagate signal becomes logic 1 while, if $A_i$ is equal to $B_i$ then the propagate signal becomes logic 0. Then the propagate signal from each of the FA blocks is given to the AND block where AND block performs the basic AND operation and produces logic 1 output if all the $P_i$ is logic 1, otherwise, it generates logic 0 as output. Further, the AND calculated output is given to the Multiplexer block which acts as a data selector. Based on the given input the multiplexer selects the required value, if the input is logic 1 multiplexer selects the $C_0$ as the output (i.e. $C_{out} = C_0$). Here, the carry is bypassed to the output instead of propagating through all the blocks (i.e. carry is skipped instead of propagating and directly given to the output.). If the multiplexer input is logic 0 then it selects the carry from the last FA which is propagated from the initial FA. Significantly, in the CSA circuit overall delay produced by the carry propagation is reduced by using the carry skip mechanism and helps to improve the overall speed of the addition operation. Using this addition concept, the multiplied values in the IIR filter design get added and produced the desired filter output. Initially, during the first clock cycle, the multiplied values are added with values in the accumulator (i.e. initially zero) and get stored in the accumulator itself. For the next clock cycle, the same input data is multiplied with the next filter coefficient $h_1$ which are produced the processing element results. These results are used to perform the accumulator operation for the same clock cycle. Similarly, the above-mentioned operations are recursively repeated based on the clock signal for the same input data and for different filter coefficients the results are computed and tabulated in the resulting section. Due to the usage of CSA and AM, the overall IIR filter processed with high speed. The combinational and sequential blocks are operated with less fractional delay which helps to improve the filter performance.

## V. Simulation Setup

The proposed AM-CSA IIR filter architecture has been implemented using 4GB RAM with 3.30 GHz, and a 500GB hard disk. The proposed algorithm used Modelsim 10.5 software to simulate the waveform to verify the timing diagram. Verilog language has been used to write the coding for each FPGA module. Xilinx 14.4 ISE software is used to evaluate the FPGA performances and to take the RTL schematic of each module. The MATLAB r2018a software is used to generate the co-efficient using the Parks-McClellan algorithm.

### *A. Results and Discussion*

This section briefly discusses and analyzes the results obtained from the proposed AM-CSA IIR filter. Filter coefficients are the key parameters in design which are calculated by generating the desired MATLAB code. The pseudo-code for the Parks McClellan algorithm is programmed with the help of a keyword named 'firpm' to generate the filter coefficients. The 'firpm' is the keyword for the syntax in which $f_p$ (pass band frequency) and $f_s$ (stop band frequency) values are consider to be important. The pseudo code for the Parks McClellan algorithm is executed in MATLAB and the generated filter coefficient result is shown in the Fig. 6.

```
clc;
clear all;
close all;
fp = 0.1;
fs = 0.15;

[h, del] = firpm(8, [0 fp fs .5]*2, [1 1 0 0]);

length(h)
del

figure(1)
stem(0:8, h, 'filled')
title('Impulse response')
```

```
mmand Window
w to MATLAB? See resources for Getting Started.

>> h

h =

  -0.0592   0.1399   0.1818   0.2332   0.2548   0.2332   0.1818   0.1399  -0.0592

>> Y = round (255*h)

Y =

  -15   36   46   59   65   59   46   36  -15
```

Fig. 6. Parks-McClellan algorithm output.

In the pseudo-code, the $f_p$ and $f_s$ values are given as 0.1 Hz and 0.15 Hz respectively and then the pseudo-code complies. During the compilation process, step-by-step codes have complied where if the compiler sees the 'firpm' syntax it automatically calls a separate subroutine which is being written using the Parks McClellan algorithm. Here, the Parks-McClellan algorithm generates the filter coefficients Filter Coefficient (FC) according to the $f_p$, $f_s$ and the filter order. The filter order, stop band and pass band frequency are 8, 0.15 Hz and 0.1 Hz respectively. According to the filter order, the firpm syntax returns 8 different values. Further, these values are simplified by round offing the value using the syntax $Y$=round(255×h), where $Y$ is FC. Finally, the round-off values are obtained and these values are considered as the filter coefficients. There is no necessary to consider the sampling frequency in this design, because the Parks-McClellan algorithm generates the FC value based on the filter order, stop band and pass band frequency. The type of filter designed in this research is LPF.

If $f_p$= 0.1 Hz and $f_s$ = 0.15 Hz, the co-efficient output is 6 which is stored in ROM to perform the IIR filtering operation. Similarly, $f_p$ and $f_s$ values are changed and FC values are updated. These updated values are stored in different memory of ROM.

### *B. Fractional Delay*

The performance evaluation of the digital filters is based on considering the important parameter termed fractional delay which delays the signal by a fractional number of samples. The pseudo-code is written for the

fractional delay using the MATLAB software which result is presented in Fig. 7.

In the programmed code, the keyword named 'frac_delay_lpf' is used to compute the fractional delay value *b*. To compute the delay value, the subroutine considers the number of taps in the filter (*n* taps), sampling frequency ($f_s$) and cutoff frequency ($f_c$). As previously, various FC values are computed and these coefficient values are used to compute the 'b' value where the mean value is taken for all *b* values and finally fractional delay output (frac_delay_out=0.0057ns) is computed. The computed delay value proves that the fractional delay value of the proposed AM-CSA IIR filter design is comparatively low, so that speed of the filter is not affected and retains the performance.

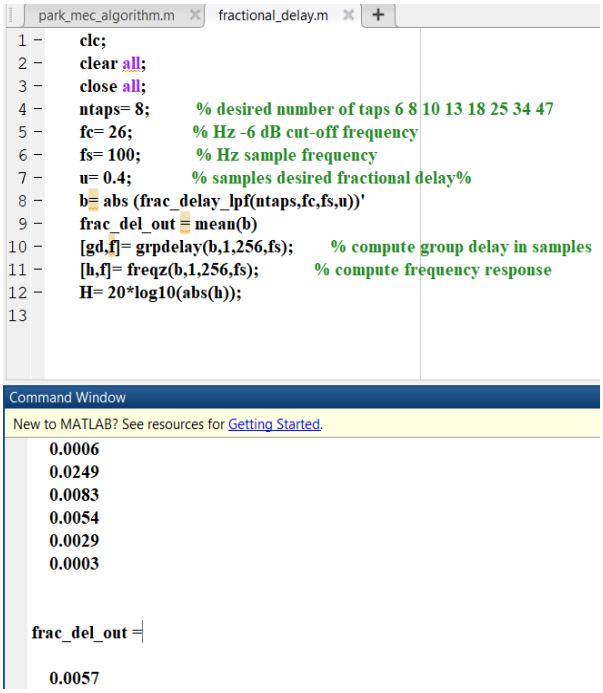


Fig. 7. Fractional delay output of AM-CSA IIR filter.

## C. FPGA Performances

For analyzing the FPGA performances of the AM-CSA IIR design, Verilog code is written for the IIR filter design using ModelSim 10.5 software. The schematic of the designed IIR filter is shown in the Fig. 8. The designed Verilog code is further simulated to check the functionality by generating the waveforms which are presented in Fig. 9.

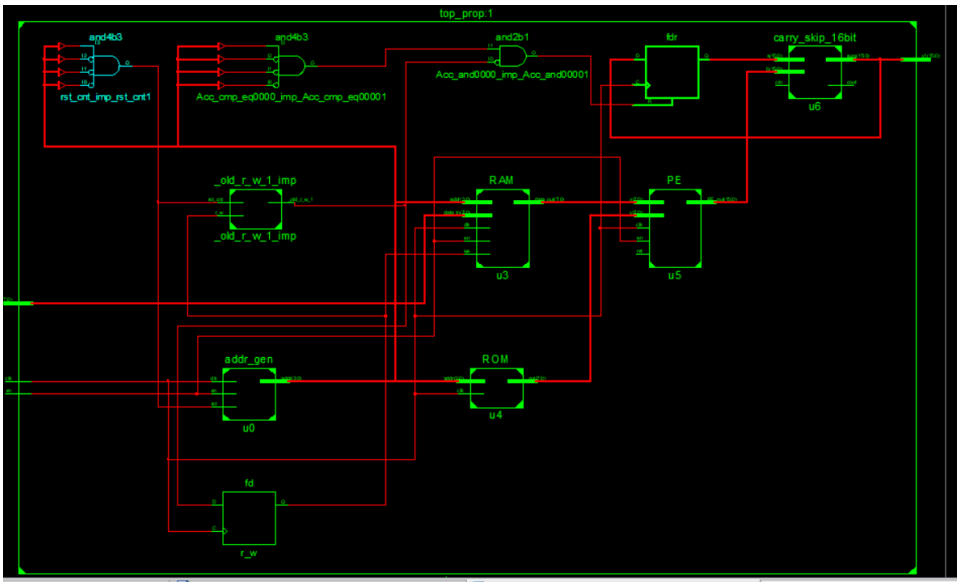


Fig. 8. The simulated output waveform of the AM-CSA IIR design.

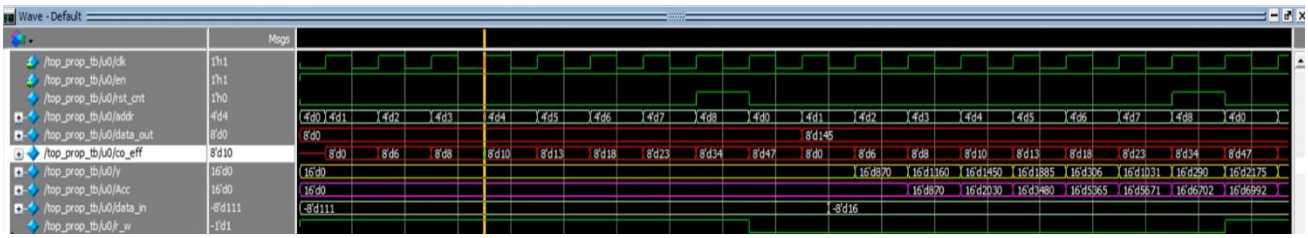


Fig. 9. The simulated output waveform of the AM-CSA IIR design.

In the simulated waveform, the first clock cycle of the initial data_out value is given as zero which is represented in 8 bit decimal value as 8'd0. Similarly, $A_{cc}$ and *y* values are represented in 16 bits. Here, *d* is denoted as decimal value. While the given value is zero, the filter performs the filtering operation and generates the output which is stored in the accumulator as $A_{cc}$ = zero. Further, the data_out value 145 is given as one input and the filter coefficient $h_0$=8'd6 as another input is given, it generates output as *y*=16'd870 which is then stored in the accumulator ($A_{cc}$=16'd870). Next, using the same data_out value with the next filter coefficient $h_1$=8'd8 the

multiplier output $y$=16'd1160 is obtained. Further, it gets added with the previously stored accumulator value and produces output as $A_{cc}$=16'd2030. Similarly, the remaining outputs are obtained at the respective clock instance and finally for $h$=8'd47 filter output values are obtained as $A_{cc}$=16'd6992. After checking the functionality with the simulated waveforms it is further synthesized in various FPGA platforms using the Xilinx 14.4 ISE software and their performance are evaluated in the given Table I, Table II and Table III.

TABLE I: UTILIZATION TABLE OF AM-CSA IIR FILTER IN VIRTEX 5 FPGA

| FPGA performances | Total resources | Occupied resources | % of utilization |
|---|---|---|---|
| Number of slice registers | 12480 | 33 | 0.264% |
| Flip Flops | 12480 | 33 | 0.264% |
| Number of slice LUTs | 12480 | 154 | 1.23% |
| Number of used as a logic | 12480 | 145 | 1.16% |
| Slices | 3120 | 58 | 1.8% |
| Bonded IOB | 172 | 23 | 13.37% |

TABLE II: UTILIZATION TABLE OF AM-CSA IIR FILTER IN VIRTEX 6 FPGA

| FPGA performances | Total resources | Occupied resources | % of utilization |
|---|---|---|---|
| Number of slice registers | 93,120 | 47 | 0.05% |
| Flip Flops | 93,120 | 40 | 0.043% |
| Number of slice LUTs | 46,560 | 124 | 0.266% |
| Number of used as a logic | 46,560 | 121 | 0.259% |
| Slices | 11,640 | 47 | 0.404% |
| Bonded IOB | 240 | 23 | 9.5% |

TABLE III: UTILIZATION TABLE OF AM-CSA IIR FILTER IN VIRTEX 7 XC7VX330T FPGA

| FPGA performances | Total resources | Occupied resources | % of utilization |
|---|---|---|---|
| Number of slice registers | 408000 | 46 | 0.0113% |
| Flip Flops | 408000 | 40 | 0.01% |
| Number of slice LUTs | 204000 | 115 | 0.0564% |
| Number of used as a logic | 204000 | 121 | 0.059% |
| Slices | 51000 | 61 | 0.12% |
| Bonded IOB | 600 | 22 | 3.77% |

The amount of hardware utilized by the AM-CSA IIR filter design in three different FPGA platforms like Virtex 5, Virtex 6 and Virtex7 xc7vx330t FPGA are tabulated in above Table I, Table II and Table III. For each FPGA platform, the designed IIR filter utilized a minimum amount of hardware resources, hence it is clearly stated that the proposed design is said to be more effective in various platforms. On the whole, Virtex 7 FPGA contains high available resources of slice registers (408000), flip flops (408000), LUT (204000), slices (51000) and bounded IOB (600). In this platform, minimum resources of about slice registers (0.0113%), flip flops (0.01%), LUT (0.0564%), slices (0.12%) and IOB (3.77%) are occupied by the proposed work when compared to the remaining platforms. According to the above table, Virtex 7 FPGA platform is the most promising platform for the hardware synthesis of the designed work.

### *D. Electrical Characteristic Performances*

In this section, the electrical characteristics performances are analyzed and the results are given in the Table IV.

TABLE IV: ELECTRICAL CHARACTERISTICS PERFORMANCES

| Performances | Virtex 5 | Virtex 6 | Virtex 7 |
|---|---|---|---|
| DSP | 145 | 124 | 114 |
| Throughput | 1.99 | 2.49 | 2.54 |
| Delay (ns) | 4.02 | 3.21 | 3.14 |
| RAM count | 1 | 1 | 1 |
| Frequency (MHz) | 159.3 | 167.21 | 178.1 |

The throughput of the system is evaluated as

$$\text{Throughout}=(\text{Number of bit per one clock})\times(\text{Highest clock speed}) \quad (11)$$

where Number of bits per one clock cycle is 8, and the highest clock speed is evaluated by using

$$\text{Highest clock speed} = \frac{1}{\text{delay}} \quad (12)$$

The DSP, delay and frequency performances are evaluated from the synthesis report using Xilinx ISE software.

From the electrical characteristics, it is clears that the proposed filter architecture contains less delay, RAM count and DSP blocks. Because of fetching the data speedily from RAM and ROM, the operating frequency of the filter design is crossed more than 150 MHz for all the Virtex families. Due to the usage of optimal AM and CSA, the combinational blocks are reduced which helps to achieve the small component count.

### *E. Comparative Analysis*

In this section, the comparative analysis of AM-CSA IIR filter design with the existing works are stated and discussed. The proposed design is compared with the prior works like FIR-based IIR design [20], Parallel pipeline IIR [19], Optimized FIR filter [22] and CAPF with retiming [29]. The comparison is tabulated in Table V, based on the number of registers, LUT, IOB utilized and amount of power consumed. In [19], the parallel pipeline filter required 591 LUTs to perform the filter operation. In [20], FIR based IIR design filter is implemented with less speed. In [22], optimal FIR filter required more memory to perform the Vedic multiplication operation. These all the issues are solved in AM-CSA-IIR filter work.

From Table V, it is proved that the proposed IIR filter design provides performance results in terms of registers, LUT and IOB utilization and power consumption. Additionally, the delay comparison is provided in the Table VI. They compared existing works results are obtained by synthesizing in different platforms like Virtex-5 XC5VLX50T FPGA [20], [29], Artix 7 [19], and Virtex 7 xc7vx330t FPGA [22]. When comparing the proposed AM-CSA IIR filter design with the aforementioned existing works, it is predominantly better than other works since it consumed only 4.0267% of overall hardware resources. The hardware utilization percentage is comparatively minimum and consider to be promising than the other works. The proposed AM and CSA contains less number of combinational blocks and registers. So, it helps to improve the filter performance and hardware utilization. In this comparison, it is

undoubtedly proven that the AM-CSA IIR filter design provides better results and required only a few filter coefficients that significantly reduces the resource utilization (LUT=0.0564%, Registers=0.0113% and IOB=3.77%). While focusing on the power consumption, the proposed AM-CSA IIR filter design comparatively consumes only a minimum amount of power (i.e. Virtex-5 XC5VLX50T FPGA=69.34mW, Artix 7= 48.67mW, and Virtex 7 xc7vx330t FPGA= 35.89mW) because of less utilization of the hardware resources. On the whole, in the Virtex 7 platform, the AM-CSA IIR filter attains a minimum percentage of resource utilization with the fractional delay of about 0.0057ns and consumes only 35.89mW of power.

TABLE V: COMPARATIVE ANALYSIS OF DIFFERENT METHODOLOGY FOR DIFFERENT FPGA FAMILIES

| Word length | Family | Device | Methodology | Registers | LUT | IOB | Power |
|---|---|---|---|---|---|---|---|
| 8 bit | Artix 7 | XC7A100T | Parallel pipeline IIR [19] | 527 | 591 | 206 | 67mW |
| | | | AM-CSA-IIR | 215 | 294 | 146 | 48.67mW |
| | Virtex 5 | XC5VLX50T | FIR based IIR design [20] | 172 | 120 | 35 | 0.12W |
| | | | DF-II [29] | NA | 15, 484 | NA | NA |
| | | | Cascade structure [29] | NA | 1733 | NA | NA |
| | | | Parallel structure [29] | NA | 1412 | NA | NA |
| | | | Lattice ladder [29] | NA | 11246 | NA | NA |
| | | | SLA [29] | NA | 5729 | NA | NA |
| | | | CAPF with retiming [29] | NA | 793 | NA | NA |
| | | | AM-CSA-IIR | 165 | 110 | 22 | 69.34mW |
| | Virtex 7 | XC7VX330T | Optimized FIR filter [22] | 48 | 128 | 26 | 49.62mW |
| | | | AM-CSA-IIR | 46 | 115 | 22 | 35.89mW |

TABLE VI: COMPARISON OF DELAY

| Methodology | Delay (ns) |
|---|---|
| DF-II [29] | 8.068 |
| Cascade structure [29] | 6.074 |
| Parallel structure [29] | 6.030 |
| Lattice ladder [29] | 13.498 |
| SLA [29] | 6.737 |
| Cascaded structure with retiming [29] | 6.074 |
| CAPF with retiming [29] | 5.530 |
| AM-CSA-IIR | 4.02 |

## VI. CONCLUSION

An innovative integration methodology for improving the filtering process with minimum hardware utilization and improved linearity performance has been presented in this paper. In this design, an improved optimized AM-CSA IIR digital filter is designed with the integration which efficiently optimized multiplier and adder. Due to the integration, the footprint area of the design is reduced and further proposed optimization tends to increase the speed of the filtering operation. Therefore, delay and power consumption of the processing elements are minimized to a greater extent and provides an optimistic performance in the filtering process. In the AM-CSA IIR filter design, the filter coefficients are accelerated in the MATLAB r2018a software, and then the design is simulated and synthesized in Modelsim 10.5 and Xilinx ISE 14.4 software. The filter coefficients are effectively generated using the Parks McClellan algorithm with $0.005f_s$ periodic reduction. Here, 3 different FPGA platforms are taken into account (i.e. Virtex 5, Virtex 6 and Virtex 7) to verify the performance of the design. From the simulation results and the FPGA performance comparison, it is clearly stated that the designed improved IIR filter avails better performance results in all the standard parameters such as minimum hardware utilization (LUT= 115 (0.0564%), FF=40 (0.01%) and IOB=22 (3.77%)) and reduced fractional delay of about 0.0057ns. Overall, the proposed design becomes a promising filter design in the DSP applications with improved filtering options, speed and reduced area, power and delay. In the future, a different optimal filter can be designed and implemented in FPGA hardware to reduce the power consumption and fractional delay.

## CONFLICT OF INTEREST

The authors declare no conflict of interest.

## AUTHOR CONTRIBUTIONS

The paper conceptualization, methodology, software, validation, formal analysis, investigation, resources, data curation, writing—original draft preparation, writing—review and editing, visualization, have been done by 1st author. The supervision and project administration, have been done by 2nd author.

## REFERENCES


[1] M. D. Cacqueray-Valmenier, A. Coskun, and I. Kale, "The use of almost linear phase IIR filters in DFT modulated filter banks for communication systems," in *Proc. 17th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, Edinburgh, UK, 2016, pp. 1-4.

[2] K. Malviya and A. Nandi, "Design of IIR filter using wallace tree multiplier," in *Proc. 2nd Int. Conf. on Power, Energy and Environment: Towards Smart Technology (ICEPE)*, Shillong, India, 2018, pp. 1-4.

[3] U. T. Shaikh, I. H. Kalwar, T. D. Memon, and F. Shaikh, "Design of IIR filter using PSO algorithm and its implementation in FPGA," *Indian Journal of Science and Technology*, vol. 10, no. 36, pp. 1-5, Sep. 2017.

[4] H. Lee, K. Kim, Y. Kwon, and E. Hong, "Real-time particle swarm optimization on FPGA for the optimal message-chain structure," *Electron.*, vol. 7, no. 11, pp. 274, Oct. 2018.

[5] A. Volkova, M. Istoan, F. De Dinechin, and T. Hilaire, "Towards hardware IIR filters computing just right: Direct form I case study," *IEEE Trans. Comput.*, vol. 68, no. 4, pp. 597-608, Apr. 2019.

[6] H. J. Ko and J. J. P. Tsai, "Robust and computationally efficient digital IIR filter synthesis and stability analysis under finite precision implementations," *IEEE Trans. Signal Processing*, vol. 68, pp. 1807-1822, Mar. 2020.

[7] G. Pepe, L. Gabrielli, S. Squartini, and L. Cattani, "Evolutionary tuning of filters coefficients for binaural audio equalization," *Applied Acoustics*, vol. 163, pp. 107204, Jun. 2020.

[8] V. Pathak, S. J. Nanda, A. M. Joshi, and S. S. Sahu, "VLSI implementation of anti-notch lattice structure for identification of exon regions in Eukaryotic genes," *IET Comput. Digital Tech.*, vol. 14, no. 5, pp. 217-229, Sep. 2020.

[9] V. Pathak, S. J. Nanda, A. M. Joshi, and S. S. Sahu, "Hardware implementation of infinite impulse response anti-notch filter for exon region identification in eukaryotic genes," *Int. J. Circuit Theory Appl.*, vol. 48, no. 12, pp. 2242-2256, Dec. 2020.

[10] O. S. Ulriche, M. L. Bertrand, G. N. E. R. Christian, and M. Jean, "A novel FPGA-based multi-channel signal acquisition system using parallel duty-cycle modulation and application to biologic signals: design and simulation," *Journal of Electrical Engineering, Electronics, Control and Computer Science*, vol. 7, no. 24, pp. 13-20, 2021.

[11] N. Agrawal, A. Kumar, V. Bajaj, and G. K. Singh, "Design of digital IIR filter: A research survey," *Applied Acoustics*, vol. 172, Jan. 2021.

[12] G. Deng, J. Chen, J. Zhang, and C. H. Chang, "Area-and power-efficient nearly-linear phase response IIR filter by iterative convex optimization," *IEEE Access*, vol. 7, pp. 22952-22965, Feb. 2019.

[13] P. J. Ahamed and M. A. Haseeb, "Implementation of digital IIR filter design based on field programmable gate array," *Mater. Today: Proc.*, vol. 45, no. 2, pp. 2573-2577, Jan. 2021.

[14] P. Kumar, P. C. Shrivastava, M. Tiwari, and A. Dhawan, "ASIC implementation of area-efficient, high-throughput 2-D IIR filter using distributed arithmetic," *Circuits, Systems, and Signal Processing*, vol. 37, no. 7, pp.2934-2957, 2018.

[15] H. K. Mewada and J. Chaudhari, "Low computation digital down converter using polyphase IIR filter," *Circuit World*, vol. 45, no. 3, pp. 169-178, 2019.

[16] R. W. Hamad, "Design and FPGA implementation of 11th order efficient IIR wavelet filter banks with approximate linear-phase," *Academic Journal of Nawroz University*, vol. 7, no. 4, pp. 207-212, 2018.

[17] L. Chen, M. Liu, J. Wu, J. Yang, and Z. Dai, "Structure evolution-based design for low-pass IIR digital filters with the sharp transition band and the linear phase passband," *Soft Computing*, vol. 23, pp. 1965-1984, Mar. 2019.

[18] R. Seshadri and S. Ramakrishnan, "Design and implementation of cost-effective FPGA implementation of fast digital FIR and IIR filters," *Concurrency and Computation: Practice and Experience*, vol. 33, no. 3, pp. e5246, Feb. 2021.

[19] R. Shirshendu, "Parallel and pipeline implementation of IIR Low pass filter on FPGA," *Digital System Design*, pp. 1-13, May 2020.

[20] D. Datta and H. S. Dutta, "High performance IIR filter implementation on FPGA," *J. Electr. Syst. Inf. Technol.*, vol. 8, no. 1, pp. 1-9, Jan. 2021.

[21] A. Abinaya, M. Maheswari, and A. S. Alqahtani, "Heuristic analysis of CIC filter design for next-generation wireless applications," *Arabian Journal for Science and Engineering*, vol. 46, pp. 1257-1268, Feb. 2021.

[22] A. Eddla and V. Y. J. Pappu, "Low area FPGA implementation of FIR filter with optimal designs using Parks-Mcclellan," *International Journal of Advanced Research in Engineering and Technology*, vol. 12, no. 2, pp. 361-376, Feb. 2021.

[23] J. Potsangbam and M. Kumar, "Design and implementation of combined pipelining and parallel processing architecture for FIR and IIR filters using VHDL," *International Journal of VLSI Design and Communication Systems*, vol. 10, no. 4, 2019.

[24] C. J. Kikkert, "A phasor measurement unit algorithm using IIR filters for FPGA implementation," *Electronics*, vol. 8, no. 12, 2019.

[25] A. Mahabub, "Design and implementation of cost-effective IIR filter for EEG signal on FPGA," *Australian Journal of Electrical and Electronics Engineering*, vol. 17, no. 2, pp. 83-91, 2020.

[26] F. J. Iniguez-Lomeli, Y. Bornat, S. Renaud, J. H. Barron-Zambrano, and H. Rostro-Gonzalez, "A real-time FPGA-based implementation for detection and sorting of bio-signals," *Neural Computing and Applications*, vol. 33, pp. 12121-12140, Mar. 2021.

[27] R. F. Hassan, "Performance investigation of digital lowpass IIR filter based on different platforms," *International Journal of Electrical and Computer Engineering Systems*, vol. 12, no. 2, pp. 105-111, 2021.

[28] V. Pathak, S. J. Nanda, A. M. Joshi, and S. S. Sahu, "FPGA implementation of high-speed tunable IIR band pass notch filter for identification of hot-spots in protein," *International Journal of Circuit Theory and Applications*, vol. 49, no. 11, pp. 3748-3765, 2021.

[29] M. Y. Ladekar, Y. V. Joshi, and R. R. Manthalkar, "Performance analysis in higher-order IIR filter structures with application to EEG signal," *Circuits, Systems, and Signal Processing*, vol. 40, pp. 4047-4063, Aug. 2021.

[30] J. H. McClellan and T. W. Parks, "A personal history of the Parks-McClellan algorithm," *IEEE Signal Process Mag.*, vol. 22, no. 2, pp. 82-86, Mar. 2005.



**E. Arunjyothi** received B.E. degree in electronics and communication engineering from Osmania University, 2006 and M.Tech. degree with specialization VLSI system design from JNTU Hyderabad in 2008 and pursuing Ph.D. degree from GITAM Deemed to be University, Visakhapatnam in the area of digital VLSI. Interested areas are VLSI design, digital system design etc.

**P. V. Y. Jayasree** received her M.E. and Ph.D. degrees in electronics and communications engineering from Andhra University, Visakhapatnam and JNT University, Kakinada in 1999 and 2010 respectively. She is currently a professor of electronics and communication engineering at GITAM University, Visakhapatnam, India. Her research interest lies in the areas of EMI//EMC, antennas, microwaves, etc.