International Journal of Engineering Trends and Technology
ISSN: 2231 – 5381 /doi:10.14445/22315381/IJETT-V69I9P224
Volume 69 Issue 9, 203-211, September, 2021

FPGA Implementation Of VM-CSA Fir Filter With Reduced Area And Delay Using Optimal Designs

[1]Arunjyothi Eddla, [2]Dr.VY.Jayasree Pappu

[1]*Ph.D Scholar Dept. of ECE, GITAM Institute of Technology, Visakhapatnam, Andhra Pradesh, India*
[2]*Professor GITAM Institute of Technology, Visakhapatnam, Andhra Pradesh, India.*

[1]aeddla@gitam.edu, [2]jpappu@gitam.edu

Abstract *- Because of its various properties, such as Bounded-Input-Bounded-Output (BIBO) stability, phase linearity, and ease of implementation, the Finite Impulse Response (FIR) filter is widely employed in digital signal processing applications. High-order filters, on the other hand, need a large number of multipliers. As the number of multipliers grows, the hardware complexity and power of the FIR filter grows as well. In digital signal processing applications, the design of the low area and low power FIR filters is critical. To reduce hardware consumption, an essential form of FIR filter called the Interpolated Spectral Parameter Approximation (ISPA) filter is introduced in this study. To reduce the number of logical components in the ISPA filter, the Carry SKIP Adder (CSA) and Vedic Multiplier (VM) are combined. Furthermore, raising the ISPA filter's working frequency reduces the ISPA filter's latency. The Parks-McClellan algorithm will be used to create the coefficient. The ISPA filter, as well as the optimum adder and multiplier, are implemented using Modelsim 10.5 and Xilinx 14.4. The suggested ISPA filter's performance is evaluated in terms of filter output, LUT, flip flops, slices, power, and delay.*

Keywords**:** *Carry Look Ahead adder, Finite Impulse Response, Interpolated Spectral Parameter Approximation, Parks-McClellan, and Vedic Multiplier*

I. INTRODUCTION

The Finite Impulse Response (FIR) filter is now widely regarded as the most fundamental circuit utilized in DSP hardware [1]. Because of their absolute stability and linear phase characteristics, FIR digital filters are widely used in mobile communication systems for channel equalization, matched filtering, and pulse shaping [2]. Because these circuits serve critical functions in today's DSPs, their speed and power optimization are critical. In high-performance DSP applications, there is a quality factor.

DSP applications often have a tradeoff between power consumption and speed, necessitating the development of low-power, high-speed digital filter circuits. FPGA is becoming a popular platform for digital VLSI design [3], [4]. This is owing to the FPGAs' great flexibility, reusability, low power, moderate cost, ease of updating (thanks to the use of hardware description languages (HDL)), and feature expansion (as long as the FPGA is not exhausted) features. A changeable structure is provided by FPGAs, which consist of an array of adjustable logic modules coupled with programmable routing resources and surrounded by programmable input/output blocks [5]. It has given the FPGA-based design, and implementation of a low-power FIR circuit that is similar to peer research works for DSP applications, as well as their performance evaluations based on resource use, delay, and power concerns. Over a million comparable logic blocks (logic gates and tens of thousands of flip-flops) are contained in an FPGA [6], [7].

This implies that when a digital circuit contains hundreds of gates, standard logic design approaches such as creating logic diagrams are no longer viable.

Today's digital systems are created by writing software in the form of hardware description languages (HDLs) [8]. Simulating VHDL design and synthesizing the design to actual hardware are both done with computer-aided design tools.

Although there has been previous work on designing FIR circuits in the VLSI realm for DSP applications, the FPGA-based design challenges have not been addressed [9].

The work on FPGA-based FIR circuit design has mostly concentrated on bespoke design and implementation, with no comprehensive study of the various FIR circuits in terms of FPGA design parameters such as resource usage, latency, or power [10].

The following is a list of the research work's key contributions:

• FIR filter design with less area efficiency
• The PM algorithm is used to generate the coefficient values.
• The input data was produced using the $random function.
• Using an optimal adder and multiplier allows you to create a

FIR filter that uses minimal hardware resources.
• Fractional delay is reduced.

This research study is organized as follows: Section 2 reviews a few current research publications on the topic "FIRfilter." Section 3 provides a thorough discussion of the suggested FIR filter technique. Section 4 describes the experimental assessment of the suggested FIR filter technique.

The current study work's conclusion is presented in Section 5.

II. Literature Survey

The Distributed Arithmetic (DA)based Least Mean Square (LMS) adaptive filter was created by Kalaiyarasi, D., and Reddy, T.K. [11] to reduce the power and logic components. The Carry Save Accumulator is used to perform the accumulation and shifting operations (CSA). Furthermore, the LMS was utilized as a weight update block to determine FIR filter coefficients. In the DA, Offset Binary Coding (OBC) was used to decrease the ROM size. Then, thanks to the simultaneous operation of filtration and weight updation, the throughput was considerably enhanced. However, there are no pipeline or parallel processing aspects in the DA-based LMS. As a result, the filter's surface area expands.

J. Prasad, D.M. Geetha, and K. Srinivasan [12] showed that the FIR filter might be used in conjunction with Arid-dry pad sensors to improve medical imaging applications. With the aid of pictures, sensors were utilized to get the patient's brain signal in this study. The FIR filter was created using an adder and a Wavepipelined Vedic multiplier. To enhance the multiplier execution parameters, the characteristic of abnormal state pipelining was employed. To compensate for the Vedic multiplier's combination delay, a delay was introduced between the partial product adder and partial product generator. As a result, the entire architecture's delay increases.

Diaz, C., Sanchez, G., Avalos, J.G., Sanchez, G., Sanchez, J.C., and Perez, H. [13] created a parallel neural multiplier, known as the Spiking Neural P (SN P) multiplier, utilizing a small digital neuromorphic architecture.

The parallel neural multiplier was used to compute the FIR filters at high processing rates. Because there were fewer neurons, the neural SN P multiplier was employed to consume less space. However, while processing the higher-order FIR filter, the SN P systems have a long latency.

The low complicated architectural design was proposed for hearing aid applications by Sundar, P.P., Ranjith, D., Karthikeyan, T., Kumar, V.V., and Jeyakumar, B [14]. Using a DA-based FIR filter, the hearing aid was built without multipliers. The high-order filters of high-speed hearing aids were designed using shift-accumulate operations and look-up tables. The low-complexity hearing aid architecture was created using a multiplier-less architecture with a single DA unit. However, a large number of logical elements were utilized in this adaptive filter design, including a weight increment block, memoryless inner product block, compressor adder, and sign-magnitude separator, and

The design of a narrow transition band FIR filter was created for creating hardware efficient digital systems and this FIR filter was used in the interpolated bandpass filtering method, according to Roy, S. and Chandra A [15]. By taking into account the optimization problem, the complexity of the FIR filter was decreased. Furthermore, the decrease in complexity was achieved under a variety of restrictions, including filter length and normalized peak ripple magnitude threshold. The interpolated bandpass technique-based FIR structure resulted in lower power dissipation and lower hardware consumption. However, the system's complexity was kept to a bare minimum.

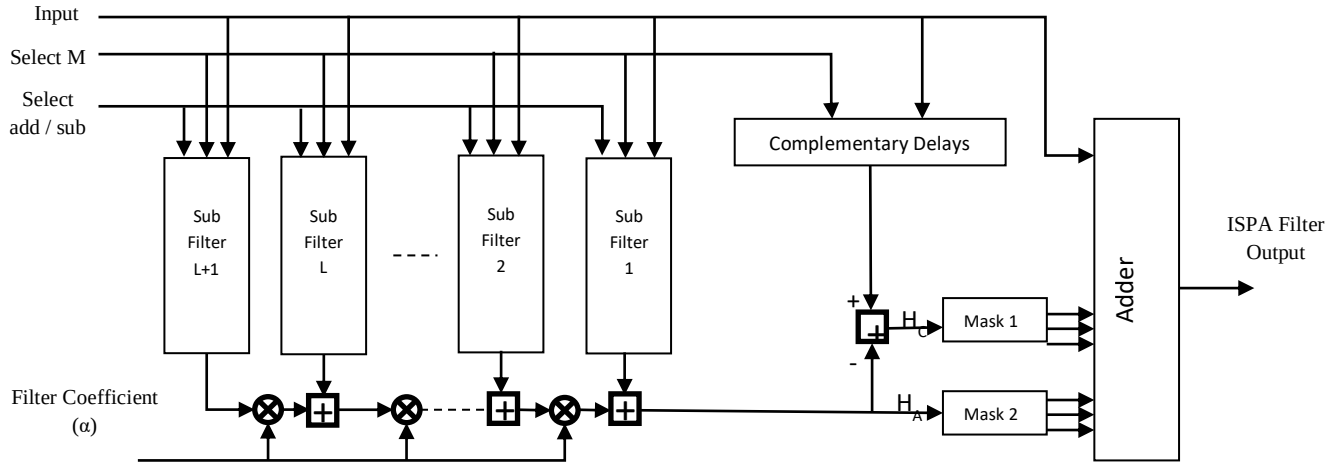

Fig 1. Architecture of ISPA filter

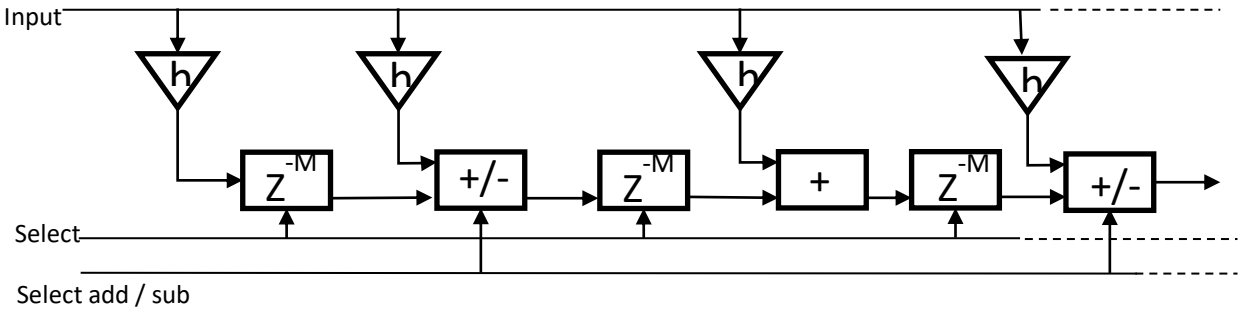

Fig 2. Architecture of Sub

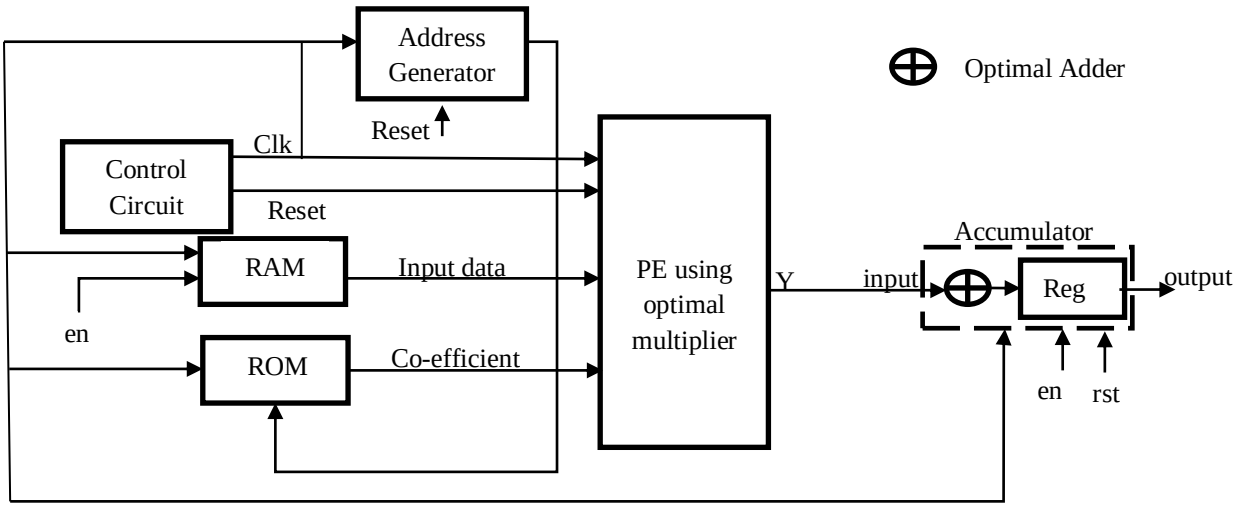

Fig 3. Overall block diagram of ISPA

III. Problem statement

In this part, the current difficulties with the FIR filter are discussed, as well as how the suggested approach addresses these issues. The following are the issues that the FIR filter has to deal with: The partial product adder and partial product generator have an additional delay in filter design [2], and the delay is exacerbated owing to the processing of high order FIR [3]. The larger delay obtained during the filtering process affects the frequency of the FIR filter. Furthermore, the overall architecture consumes more space due to the use of a large number of logical components in the filter design [4]. The more advanced logical elements

A. Solution

The ISPA filter's incorporation of an optimum adder and multiplier helps to reduce the number of logical elements such as LUTs, slices, and FF. As a result, while constructing the ISPA filter, the area is reduced. Furthermore, by lowering the size, the ISPA filter's speed is enhanced, and the frequency is improved by reducing the power used in logical components. As a result, the ISPA filter minimizes the latency during the filtering process.

IV. Proposed Method

An optimum adder and multiplier are presented in the ISPA filter in this proposed system to decrease hardware consumption. Furthermore, the ISPA filter's logical components are reduced by incorporating the best adder and multiplier. Poor stopband attenuation and larger passband ripples are disadvantages of the Spectral Parameter Approximation (SPA) filter. The suggested ISPA filter

Overcomes the shortcomings of the spectral parameter approximation filter, resulting in a narrow transition bandwidth and a relatively large cut-off frequency. Fig 1 and 2 depict the architecture of the ISPA filter and the sub-filter utilized in the ISPA, respectively.

The genuine random number generator generates the input in Fig 1, and the coefficients of the filter are represented by h in Fig 2. The Filter Design and Analysis (FDA) tool is used to create the filter coefficients. The developed ISPA filter delivers a constantly changing frequency cut in the small frequency range with transition bandwidth. Fig 3 depicts the internal construction of the ISPA filter.

The following is a description of the proposed ISPA filter's operation:

- First, the input data is collected and saved in the Random Access Memory (RAM), as well as the FDA tool's coefficient in the Read-Only Memory (ROM) (ROM). The input data is produced at random using a genuinely random number generator, and the coefficient is calculated using MATLAB's FDA tool. The Processing Element (PE) receives the input data and coefficient value, and this PE executes the multiplication operation between the first input data and the first coefficient value at the first clock cycle. Using the best multiplier results in a more efficient multiplication operation.

- The PE's output (Y) is supplied as an input to the accumulator, and this output is added to the accumulator's starting value, i.e., 0+Y=Y. Following that, the initial tap stores the sum value in the register. In addition, to increase ISPA performance, an optimum adder will be created.

- At the second clock cycle, the identical multiplication and addition procedure is applied to the second input data and second coefficient value. • The value from the addition process is added to the value saved in the register during the previous tap; the process is referred to as 8 taps if it continues for 8 clock cycles. Once the 8 tap in the filter design is done, the filter output is received.

A. Initial blocks and True Random Number Generator

The ISPA filter is an eight-tap filter that operates on the clock and reset signals. A control circuit regulates both of these signals. For every single tapping, the address of the filter coefficient and 8-bit input data are generated by the address generator. As a result, eight addresses for eight filter coefficients and eight 8-bit input data are produced. Making use of the $ random function True random number generator is used to create 8-bit input data, which is then stored in RAM memory. The produced input data is stored in RAM, while the filter coefficients are stored in the ROM register. Input data in the RAM and ROM are called based on the address and passed as input for the next process depending on the clock cycle.

B. Parks McClellan Algorithm

This method was proposed by James McClellan and Thomas Parkin in 1972 and was initially written in Fortran (Programming Language). The Parks McClellan technique is recursive and employs an indirect approach to determining filter coefficients. Filter coefficients are stored in ROM in the ISPA architecture, and these filter coefficients are determined using the Parks McClellan method. The algorithm's main goal is to use Chebyshev approximation to decrease errors in the past and stop bands. Equiripple, Chebyshev, and Remez are some of the other names for this method. The ParksMcClellan algorithm is implemented in the following steps:

Step1 (Initialization): Make an educated estimate and select the most extreme set of frequencies$\{\omega_i^{(0)}\}$.

Step 2 (Approximation of a Finite Set): Calculate the Chebyshev approximation on the chosen present extrema set and the min-max error value ($\delta^{(m)}$) on the present extremal set.

Step3 (Interpolation): Perform polynomial interpolation over the entire set of frequencies to compute the error function E(ω).

Step4 (Extrema Search): Search for the local extrema of $|E^{(m)}(\omega)|$ over the set **Ω**.

Step5 (Extrema set update): Update the extrema set to $\{\omega_i^{(m+1)}\}$ by taking new frequencies if $\max_{(\omega\varepsilon\Omega)} |E^{(m)}(\omega)|>\delta^{(m)}$, where $|E^{(m)}(\omega)|$ has its local maxima and ensure that error alternates on the ordered set of frequencies as in step 4 and 5. If the alteration theorem is not satisfied, then return to step 2 and iterate.

Step6 (Parameter test): If the alteration theorem is satisfied ($\max_{(\omega\varepsilon\Omega)} |E^{(m)}(\omega)| \leq \delta^{(m)}$)**.** Compute an inverse discrete fourier transform using set $\{\omega_i^{(0)}\}$ and interpolation formula to obtain the filter coefficients.

C. Optimal Multiplier

In a Finite Impulse Response (FIR) filter, the multiplier is an important processing element. In Digital Signal Processing (DSP) applications, the FIR filter is critical. Its performance is mostly determined by the multiplier block, which must be built to execute quick multiplication while still ensuring high efficiency and low power consumption. In modern technology, multiplier architecture has been built to meet the following requirements: fast speed, small size, low power, and high throughput. There are many other types of multiplier designs. However, the Vedic multiplier is considered to be one of the quickest and lowest-power multipliers that meets the aforementioned criteria.

a) Vedic Multiplier

The essential design idea for the multiplication process is shown in Fig 4 below. Consider a 2-bit multiplication procedure, in which two 2-bit integers, A and B, are multiplied to produce P in 4-bit.

P (product output) = A (multiplicand of n-bit) * B(multiplier of n-bit)

Let, A= $(11)_2$and B= $(11)_2$

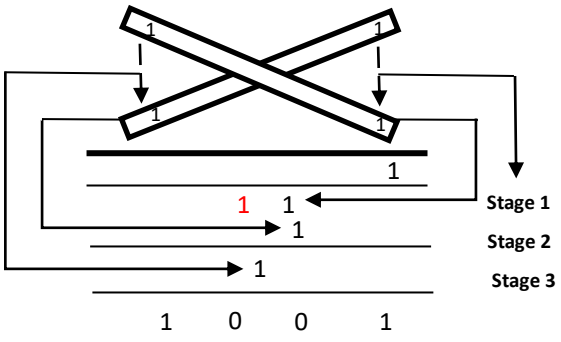

Fig 4. 2x2 multiplication based on Vedic mathematics.

output is P= $(1001)_2 = 9_{10}$

The partial products are produced in the aforementioned Vedic multiplication procedure using the steps below.
Step 1: Create the first partial product by ANDing the LSB bits of A and B [(i.e.) A0.B0].

Step 2: After the first step, the one-bit position is moved, much like in basic multiplication, and a second partial product is produced by ANDing the diagonal bits [A0.B1] and [A1.B0].

Step 3: After completing the second step, the one-bit position is moved, and the MSB bits of A and B are ANDed to produce the third partial product [(i.e.) A1.B1].

may be expanded. This architecture multiplies 8-bit input from RAM and coefficients in ROM expressed in 8-bit to generate an efficient product output Y based on this procedure.

***D. Carry skip Adder*:** Carry skip adder is also named as carrying by-pass adder, and it consists of special circuitry for improving the speed and reducing the carry propagation delay.

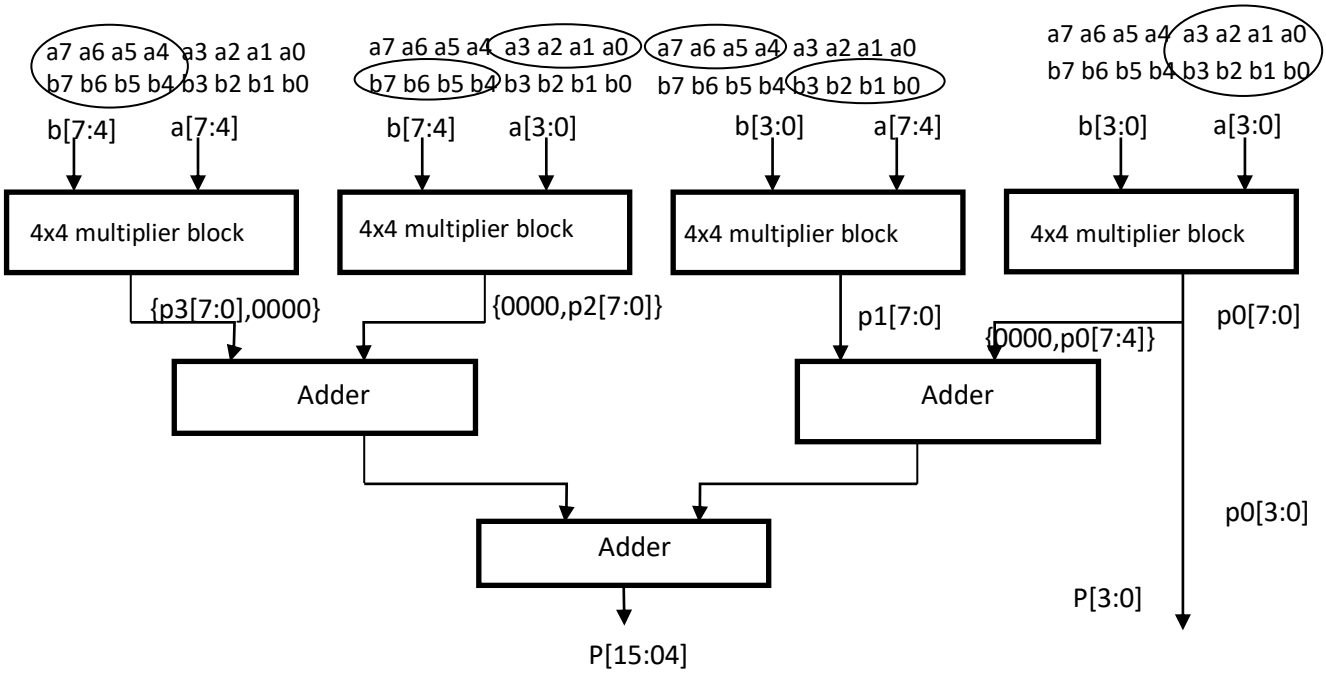

Fig 5. Architecture of 8x8 vedic

Finally, combine all of the partial products to achieve the desired result. Similarly, a 4x4 multiplier is created using a 2x2 multiplier, and an 8x8 multiplier is created using a 4x4 multiplier. 8x8 Vedic multiplier architecture is used in the proposed Interpolated Spectral Parameter Approximation (ISPA) filter design. The block diagram of the 8x8 Vedic multiplier is shown in Fig 5

1) 4x4 Vedic multiplier block and
2) Adder blocks are significant architectural features.

Mathematics. The output of each multiplier product is then sent into the adder, which performs the addition operation. The output p0[3:0] is immediately accepted as an 8x8 output (P[3:0]) in the fourth multiplier block, while the remaining p0[7:4] is passed to the following adder block to execute addition. To acquire the remaining output P[15:4], the second stage's additional output is provided as input to the following adder. For n-bit multiplication, this module

The construction of an 8x8 Vedic multiplier necessitates the use of four 4x4 Vedic multiplier blocks and three adder blocks connected in cascade (i.e., the first stage output becomes the input to the next stage). The 8-bit inputs A and B are [a7,a6,a5,a4,a3,a2,a1,a0] and [b7,b6,b5,b4,b3,b2,b1,b0], respectively. The 1st multiplier block receives the first four bits from the MSB side of A and B. The diagonal bits of A and B are then provided to the 2nd multiplier (i.e., a[3:0] and b[7:4]), and a[7:4] and b[3:0] are given as input to the 3rd multiplier block. 4-bits from the LSB side of A and B are supplied as input to the fourth multiplier block. The multiplication process is performed by the Multiplier block using Vedic

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 together defined as block propagate

block. The following Fig 6. shows the implementation of a 4-bit carry skip adder.

E. Carry skip mechanism

In the above Fig, 4-bit Ripple Carry Adder (RCA) with carrying skip circuit is integrated to form the CSA circuit. The 4-bit RCA contains 4 full adder blocks, and it generates 4 propagate 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, and A_3, B_3) which is mentioned in the below equation

$P_i = A_i$ XOR B_i

delay produced by the carry propagation is reduced 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 FIR 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, using the same input data but with the next filter coefficient, 'h1' multiplied values are produced, and this value is being added with the value stored in the accumulator during the previous clock cycle.

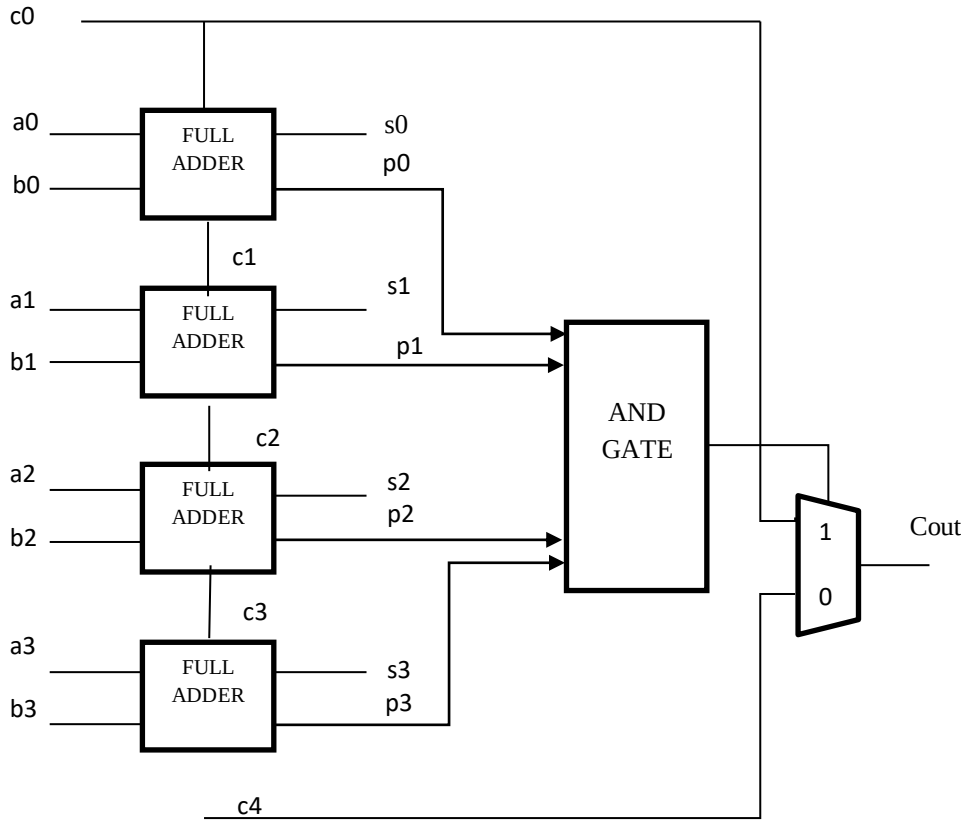

Fig 6. Block diagram of 4-bit Carry Skip

If Ai is not equal to Bi, then the propagate signals become logic '1' while if Ai is equal to Bi, then the propagate signal becomes logic '0'. Then the propagate signal from each of the FA blocks is given to the AND block, and in the AND block, it performs basic AND operation and will produce logic '1' output if the entire P_i is logic '1'; otherwise, it generates logic '0' output. Further, after this, 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', the multiplexer selects the C_0 as the output (i.e., $C_{out} = C_0$). It means that 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 being propagated from the initial FA. Significantly, in this CSA circuit, the overall

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.

V. Setup for Experiments

The suggested design was built utilizing a 500GB hard drive and 4GB RAM running at 3.30 GHz. To validate the timing diagram, the suggested technique was simulated in Modelsim 10.5 software.

The code for each and every FPGA module was written in the Verilog language. To assess FPGA performance and create RTL schematics for each module, Xilinx 14.4 ISE software is utilized. The Parks-McClellan technique is used to generate the co-efficient using the MATLAB r2015b program.

A. Results and discussion

This section covers the experimental findings and explanation of the Interpolated Spectral Parameter Approximation (ISPA) filter.

Table1.Filter coefficients generated.

fs value	x (filter coefficient)
0.15	6
0.145	8
0.14	10
0.135	13
0.13	18
0.125	23
0.12	34
0.115	47

The Parks-McClellan method is used to create the filter coefficients stored in the ROM. The Parks-McClellan algorithm is coded with the aid of the term 'firpm' in MATLAB r2015b software to create the filter coefficients. The frequency values fp (passband frequency) and fs (frequency scale) are crucial in the firpm syntax (stopband frequency). The code is run in MATLAB, and the output is displayed in Fig 7.

Initially, fp=0.1 and fs=0.15 are set, and the code is run, with firpm calling a different subroutine based on the Parks-McClellan method to create the del result of 0.0241. To make the del value easier to understand, it is rounded off with the phrase x=round(255*del) to get x=6, which is one of the filter coefficients. Similarly, by simply changing

the values of fs by 0.5, filter coefficients are generated, which are reported in table 1 and stored in various ROM memory.

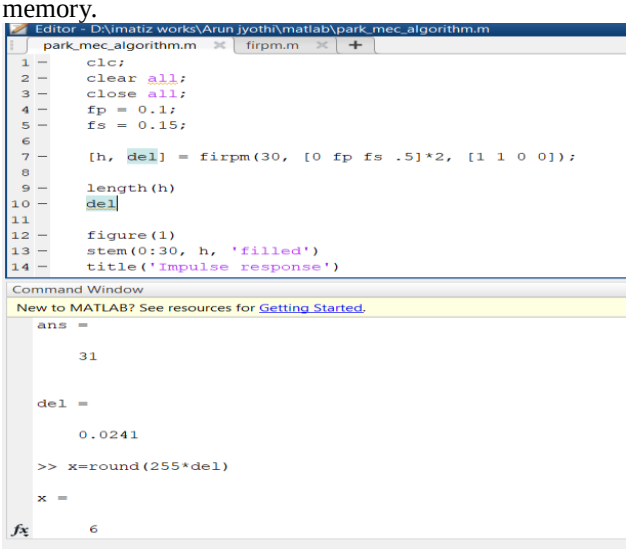

Fig 7. MATLAB output of Parks-McClellan algorithm

B. Fractional delay

An important function in the filter is a fractional delay, which delays the processed input signal a fractional of the sampling time period. MATLAB code is written for calculating fractional delay, and the result is shown in Fig 8.

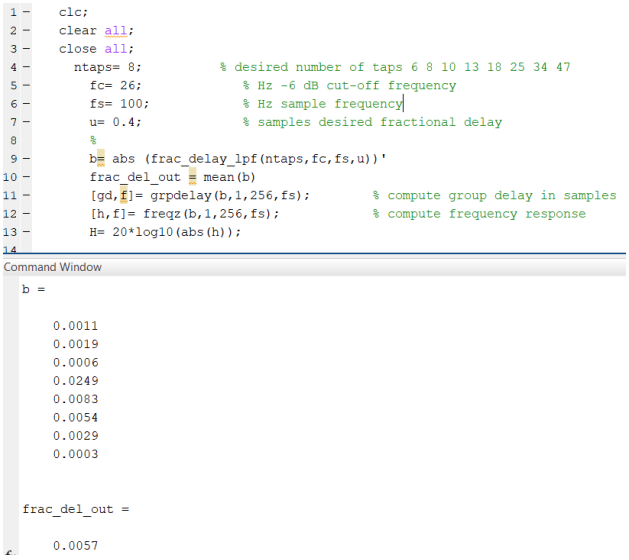

Fig 8. Fractional delay output

The keyword 'frac delay lpf' launches a separate function that uses taps (ntaps), sampling (fs), and cut-off frequency to determine the 'b' value (fc). The 'b' value is computed for the necessary eight taps (6,8,10,13,18,25,34,47) and the mean value for all the b

values is chosen to compute the fractional delay output (frac del out=0.0057). Because the obtained delay value is shorter than that of other filters, the processing of the input signal is delayed by the smallest amount possible, which has no effect on the pace of the operation and ensures high throughput.

Table.2. Comparative results for fractional delay

	CSF [19]	GAF [20]	Proposed
Fractional delay	0.1327	0.5	0.0057

Tab.2 displays the fractional delay findings in comparison. The fractional delay of the Chebyshev Sense Filter (CSF) [19] is 0.1327, whereas the fractional delay of the Genetic Algorithm Filter (GAF) [20] is 0.5. However, as compared to traditional approaches, the suggested method has a lower fractional delay (0.0057).

C. FPGA Performances

The suggested algorithm is implemented in Verilog and simulated with Modelsim 10.5 software to generate the output waveform displayed in Fig 9. In the waveform, co

eff=8'd6 (filter coefficient) multiplied value y=16'd216 and accumulator value Acc=16'd216 are acquired at the first clock cycle for data out=8'd36 (8-bit input data from RAM). Similarly, for data out=8'd129, the co eff=8'd8 multiplied value y=16'd1032 is acquired during the second clock cycle. By multiplying the multiplied value with the previously stored accumulator value, the accumulator output Acc=16'd1248 is produced. Similarly, the values are computed up to eight clock cycles, yielding Acc=16'd10382 as the final filter output.

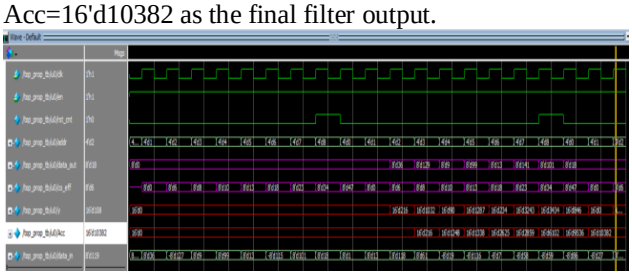

Fig 9. Simulation output waveform.

The waveform values measured are consistent with the theoretical value, indicating that the design is functioning. The simulated design is then synthesized on a variety of FPGA platforms, with the results shown in the tables below.

Table. 3 Hardware utilization of proposed FIR in Virtex 5 FPGA

FPGA performances	Total resources	Occupied resources	% of utilization
Number of sliceregisters	12480	35	1%
Flip Flops	12480	35	1%
Number of slice LUTs	12480	159	1%
Number of used as a logic	12480	151	1%
Slices	3120	62	1%
Bonded IOB	172	26	15%

Table. 4 Hardware utilization of proposed FIR in Virtex 6 FPGA

FPGA performances	Total resources	Occupied resources	% of utilization
Number of slice registers	93,120	50	1%
Flip Flops	93,120	43	1%
Number of slice LUTs	46,560	127	1%
Number of used as a logic	46,560	123	1%
Slices	11,640	50	1%
Bonded IOB	240	26	10%

Table. 5 Hardware utilization of proposed FIR in Virtex 7 xc7vx330t FPGA

FPGA performances	Total resources	Occupied resources	% of utilization
Number of slice registers	4,08,000	48	1%
Flip Flops	4,08,000	41	1%
Number of slice LUTs	2,04,000	128	1%
Number of used as a logic	2,04,000	124	1%
Slices	51,000	65	1%
Bonded IOB	600	26	4%

The hardware implementation of the suggested FIR filter in several FPGA platforms such as the Virtex 5, Virtex 6, and Virtex 7 xc7vx330t is shown in Tables 3, 4, and 5. When compared to the other two FPGA platforms, the Virtex 7 FPGA has the most resources (slice registers, FF, LUT, slices, and IOB) available, yet the suggested architecture uses just a small portion of those resources. As a result, Virtex 7 is the best platform for synthesizing and testing the ISPA design's capabilities.

D. Comparative analysis

This section describes the comparison of the proposed ISPA architecture with the present architecture. Existing techniques such as FIR filter with SNP multiplier [13], FIR with Wave Pipelined Vedic multiplier (WPV) [12], and FIR based on Vedic mathematics and RCA [14] are compared to the ISPA architecture. Slices, LUTs, flip flips, frequency, and power are all included in this comparison. Tables 6, 7, and 8 illustrate the results of comparing the proposed ISPA to several current techniques.

Table 6. Comparison of proposed architecture with SNP for Kintex 7 device

Kintex 7		
Parameters	**SNP[13]**	**Proposed**
Gates	8000	141
LUT	13000	149
Delay(ns)	500	4.93
Frequency(MHz)	2	202.7

Table 7. Comparison of proposed architecture with FIR- wave pipelined multiplier for Artix 7 device

Artix 7		
Parameters	**SNP [13]**	**Proposed**
Gates	8000	143
LUT	13000	152
Delay(ns)	500	6.32
Frequency(MHz)	2	150.08

Table 8. Comparison of proposed architecture with FIR-Vedic multiplier and RCA for Virtex- 6 device

Virtex6		
Parameters	**SNP [13]**	**Proposed**
Gates	8000	145
LUT	13000	156
Delay(ns)	500	5.85
Frequency(MHz)	2	170.9

This section describes the comparison of the proposed ISPA architecture with the present architecture. Existing techniques such as FIR filter with SNP multiplier [13], FIR with Wave Pipelined Vedic multiplier (WPV) [12], and FIR based on Vedic mathematics and RCA [14] are compared to the ISPA architecture. Slices, LUTs, flip flips, frequency, and power are all included in this comparison. Tables 6, 7, and 8 illustrate the results of comparing the proposed ISPA to several current techniques.

VI. Conclusion

A high-speed, low-area-delay Interpolated Spectral Parameter Approximation (ISPA) FIR filter is suggested in this work. The use of an optimum adder and multiplier in this filter design minimizes hardware complexity while increasing the speed of the ISPA filter by lowering the area and frequency. As a result, the processing parts' latency and power consumption are decreased throughout the filtering process, resulting in improved filter performance. The suggested architecture is simulated and synthesized in FPGA platforms such as Virtex 5, Virtex 6, and Virtex 7 using MATLAB r2015b (for calculating filter coefficients), Modelsim 10.5, and Xilinx 14.4, and the simulation results The proposed ISPA FIR filter improves performance while using the least amount of hardware (LUT=128 (1%), FF=41 (1%), and IOB=26 (4%)). The filter coefficients are successfully calculated using the parks-McClellan method by regularly decreasing 0.5fs, resulting in a fractional delay of roughly 0.0057ns. As a result, the filter speed is faster than other existing techniques. Different optimum algorithms will be developed in the future to increase FPGA performance.

REFERENCES

[1] Park, Sang Yoon, and Pramod Kumar Meher., Efficient FPGA and ASIC realizations of a DA-based reconfigurable FIR digital filter., IEEE Transactions on Circuits and Systems II: Express Briefs 61(7) (2014) 511-515.

[2] Thakur, Rakhi, and KavitaKhare., High-speed FPGA implementation of FIR filter for DSP applications., International Journal of Modeling and Optimization 3 (1) (2013) 92-94.

[3] Badave, S. M., and A. S. Bhalchandra., Multiplierless fir filter implementation on fpga., International Journal of Information and Electronics Engineering 2(2) (2012) 185.

[4] Ryou, Albert, and Jonathan Simon., Active cancellation of acoustical resonances with an FPGA FIR filter., Review of Scientific Instruments 88, no. 1 (2017): 013101.

[5] Bhattacharjee, Subhankar, SanjibSil, and AmlanChakrabarti., Evaluation of power-efficient FIR filter for FPGA based DSP applications., Procedia Technology 10 (2013) 856-865.

[6] Lehto, Raija, TarjaTaurén, and Olli Vainio., Recursive FIR filter structures on FPGA., Microprocessors and Microsystems 35(7) (2011) 595-602.

[7] Singh, Gurpadam, and Neelam R. Prakash., FPGA Implementation of Higher Order FIR Filter., International Journal of Electrical and Computer Engineering 7(4) (2017) 1874.

[8] Keerthi, M., VasujadeviMidasala, and S. NagakishoreBhavanam., FPGA implementation of distributed arithmetic for fIR filter., International Journal of Engineering Research and Technology 1(9) (2012).

[9] Szadkowski, Zbigniew, D. Głas, C. Timmermans, and T. Wijnen., First results from the FPGA/NIOS adaptive FIR filter using linear prediction implemented in the auger engineering radio array., IEEE Transactions on Nuclear Science 62(3) (2015) 977-984.

[10] Pandey, Bishwajeet, Bhagwan Das, Amanpreet Kaur, Tanesh Kumar, Abdul Moid Khan, DM Akbar Hussain, and Geetam Singh Tomar., Performance evaluation of FIR filter after implementation on different FPGA and SOC and its utilization in communication and network., Wireless Personal Communications 95(2) (2017) 375-389.

[11] Kalaiyarasi, D. and Reddy, T.K., Design and implementation of least mean square adaptive FIR filter using offset binary coding-based distributed arithmetic. Microprocessors and microsystems, 71(2019) 102884.

[12] Prasad, J., Geetha, D.M. and Srinivasan, K., Experimental setup of stretchable arid dry pad sensors for the signal acquisition fir filter design using Vedic approach. Measurement, 141(2019) 209-216.

[13] Diaz, C., Sanchez, G., Avalos, J.G., Sanchez, G., Sanchez, J.C. and Perez, H., Spike-based compact digital neuromorphic architecture for efficient implementation of high order FIR filters. Neurocomputing, 251(2017) 90-98.

[14] Sundar, P.P., Ranjith, D., Karthikeyan, T., Kumar, V.V., and Jeyakumar, B., Low power area-efficient adaptive FIR filter for hearing aids using distributed arithmetic architecture. International Journal of Speech Technology, (2020) 1-10.

[15] Roy, S. and Chandra, A., On the Order Minimization of Interpolated Bandpass Method Based Narrow Transition Band FIR Filter Design. IEEE Transactions on Circuits and Systems I: Regular Papers, 66(11) (2019) 4287-4295.

[16] Samyuktha, S and Chaitanya, D.L., VLSI design of efficient FIR filters using Vedic Mathematics and Ripple Carry Adder, Material today proceedings., (2020).

[17] Bharat Garg and Sujit Kumar Patel., Reconfigurable Carry Look-Ahead Adder Trading Accuracy for Energy Efficiency, Journal of Signal Processing Systems., (2020).

[18] SILVIU-IOAN FLIP., A Robust and Scalable Implementation of the Parks-McClellan Algorithm for Designing FIR Filters, ACM Transactions, (2016) 24.

[19] Blok, M., April. Farrow structure implementation of fractional delay filter optimal in Chebyshev sense. In Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments., 6159(4) (2016) 61594. International Society for Optics and Photonics.

[20] Singh, A., Dhillon, N., and Bains, S.S., FIR Filter Design With Farrow Structure Using Genetic Algorithm.International Journal of Computer & Organization Trends 3(9) (2013).