Speed Maximization of First Order Infinite Impulse Response Filter Based on FPGA

Hatem L. Hamza
Department of Electrical Engineering
University of Technology
Baghdad, Iraq
eee.21.18@grad.uotechnology.edu.iq

Ivan A. Hashim
Department of Electrical Engineering
University of Technology
Baghdad, Iraq
ivan.a.hashim@uotechnology.edu.iq

Najmah A. Habeeb
Depart. of Communicatiom Engineering
University of Technology
Baghdad, Iraq
najmah.a.habeeb@uotechnology.edu.iq

2023 International Conference on Engineering Applied and Nano Sciences (ICEANS) | 979-8-3503-4197-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICEANS58413.2023.10630466

***Abstract*— Digital filters are crucial for navigation, communication, and modern digital signal processing systems. This paper presents a new methodology for hardware implementation of the first-order IIR filter. There are many techniques to optimize the speed of the IIR filter, but this paper uses three techniques: look-ahead pipelining, parallel, and hybrid. The main aim of this work is to increase the operation speed with a low error percentage and minimize resource utilization with low power consumption of the first-order IIR filter. The Spartan 3A-3N FPGA is the platform used to synthesize the proposed designs. Based on the FPGA method, the suggested system achieved the lowest speed of (16.21 nsec); it consumed a considerable amount of area, utilizing 314 LUTs, 193 occupied slices, and 64 flip-flops. In contrast, the proposed 1st order IIR filter design based on the hybrid approach achieved accurate results with small area utilization: 109 look-up tables, 73 occupied slices, and 32 flip-flops. Furthermore, the approach achieves a high speed of (9.85 nsec).**

Keywords*— *first-order IIR filter, Speed optimization techniques, area minimization, low latency.

I. INTRODUCTION:

Digital signal processing establishes enormous applications ranging from speech, audio, image, and biomedical signal processing to robotics and instrumentation [1]. Every modern communication system or architecture has at least one DSP module. The most efficient method to clear out noisy signals is using signal processing technology called filtering. Filtering is an essential aspect of digital signal processing [2]. Filters are electronic systems that allow or transmit the desired frequency band while suppressing the unwanted frequency range. Two main types of filters exist in signal processing: analog and digital. The digital filter is one of the most widely used operations in digital signal and image processing applications. Digital filters include adder, multiplier, and delay unit [3][4]. Digital filters, in turn, can be categorized into two distinct classes based on their response: Infinite Impulse Response (IIR) and Finite Impulse Response (FIR) filters [5][6]. The main objective of this work is to design an IIR filter of high speed and low area utilization with accurate results. Different methods and techniques can increase the speed performance but introduce the same results, such as the look-ahead approach, parallel technique, and bitwise reduction technique. This paper is organized as follows: the second section introduces the literature survey. The third section explains the theoretical background of speed optimization techniques. The fourth section describes the proposed design of the 1st-order IIR filter based on different techniques. The fifth section presents the experimental results. The performance evaluation is introduced in section six. Finally, section seven introduces the conclusion.

II. LITERATURE SURVEY

A survey of previous studies on increasing the speed of the IIR filters has been made, which will be introduced in the next paragraph. Then, various methods and techniques of speed optimization can be used to be applied to the first-order IIR filter design, such as the traditional method, look-ahead pipelining, parallel pipelining, bitwise reduction technique, and the hybrid technique.

The proposed work in [1] discusses the traditional Moving Average (MA) FIR filters, fast MA FIR filters using look-ahead arithmetic, and traditional IIR filters using the Combination of Integrator and Comb sections (CIC) approach. The implementation is achieved with Altera using the Quartus II 13.1 synthesis tool. The authors acquired speed optimization of (25% and 10%) for the 1st order IIR filter when using the look-ahead level 1 and level 2, respectively, with increases in the utilized area by (67% and 37.5%) logic elements. At the same time, they obtained speed optimization of (22% and 11%) for the 2nd order IIR filter based on the look-ahead level 1 and look-ahead level 2, respectively, with increases in the logic elements of (22 % and 61%). The authors in [7] intend to design a digital IIR filter based on FPGA to get faster biomedical signals, especially electroencephalogram (EEG) signals. For this purpose, a high-order IIR filter is introduced to make EEG signals noise-free, less costly, and simple. FPGA load-up is utilized for hardware usage, a mix of various logic gates offering economical and enduring administrations. The authors aim to design and implement a combined parallel processing and pipelining architecture for FIR and IIR filters using Very High-Speed Integrated Circuit Hardware Descriptive Language (VHDL) to reduce the power and delay of the filter. The ALTERA Cyclone II board was utilized for actualizing our proposed design. The total device utilization of the proposed design is as follows: (117 flip-flops, 137 occupied slices, and 79 LUTs). The dissipated power is about 79 mW, and the clock frequency is 140 MHz. D. Datta and H. S. Dutta present an enhanced system design of a reconfigurable IIR filter. A parallel-pipeline-based FIR filter is used to accomplish the suggested IIR architecture. Excellent characteristics of the FIR filters include linear phase, high stability, and fewer finite precision errors. So, regarding signal processing, FIR-based IIR architecture is more attractive and selective. The look-ahead and two-level pipeline IIR filter designs are covered. All the proposed designs have been synthesized and tested on Xilinx Virtex-5. According to the implementation results, the suggested FIR-based IIR architecture reduces hardware utilization (4% of flip-flops and 10% LUTs), higher operating

speed of (14% and 29%), and lower power consumption when compared to the look-ahead, and parallel techniques [8].

III. THEORETICAL BACKGROUND OF THE SPEED OPTIMIZATION TECHNIQUES

Several methods and techniques with different form structures are presented and discussed in the following subsection to increase the speed of the IIR filter system.

A. Distributed Arithmetic (DA)

Distributed arithmetic (DA) is a multiplier-less hardware-efficient method for storing pre-computed values in LUTs of FPGAs since multiplication needs a huge memory space [9] [10].

B. Pipeline Approach

In most DSP systems, pipelining is an implementation approach that increases critical path speed by concurrently executing a stream of instructions by lowering critical route latency. [11]. This approach adds to the original transfer function with an additional canceling pole and zero in the Z-plane. Pipelining does not speed up instruction execution time but does speed up program execution time by increasing the number of instructions completed per the time unit [12][11][13][14]. Different types of pipeline techniques, such as look-ahead and parallel pipeline techniques, will be introduced in the following sections.

C. Look-Ahead (LA) Pipeline Approach

Recursive digital filters can be implemented at high speeds on VLSI using look-ahead (LA) techniques. [15]. There are different types of LA methods [16][17][18], which are Clustered Look-Ahead (CLA) algorithms that group the past output data to achieve pipelined IIR filters. Since CLA cannot ensure stability, a Scattered Look-Ahead (SLA) technique is suggested, which employs evenly separated previous output data and delivers stable pipelined IIR filters with linear increases in hardware. Additionally, to achieve a stable design with minimal hardware complexity, the Distributed Look-Ahead (DLA) algorithm combines the two aforementioned schemes[19]. The majority of LA research focuses on IIR filter hardware reduction and stability. Designing an effective algorithm to locate augmented polynomial coefficients takes less attention. The Look-Ahead pipelining method was chosen in this work because it greatly increases sample rates while using fewer multipliers and adding canceling poles and zeros to the original transfer function.

D. Parallel Pipeline Technique

Multiple outputs are computed concurrently during a clock cycle in a process known as parallel processing. Multiple Input Multiple Output (MIMO) parallel systems are created by duplicating the serial system's hardware in parallel processing. The block diagram for converting a Single Input Single Output (SISO) system to a MIMO system is shown in Figure 1[20] [21].

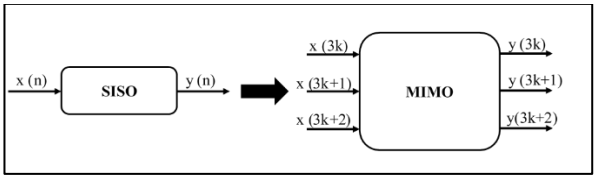

Figure 1: Sequential System to 3-Parallel System [20].

Parallel processing is a reliable approach since it may be used to boost the throughput of a digital filter while decreasing its power consumption [22]. The hardware complexity of the IIR filter in simple parallel processing implementation is L^2 multiply-add operations since L multiply-add operations are required for each output and there are L outputs in total. Hardware complexity can be decreased by using the incremental block processing method. The outputs are progressively commuted sequentially in incremental computation by using the non-recursively calculated intermediate state [23]. On the other hand, the system latency has increased when the hardware complexity has been lowered to ($2L-1$ from L^2) [24]. This approach is employed in this work since it consumes fewer resources than the DA technique.

E. Bitwise (Word Length) Reduction Technique

The bitwise technique is also known as the data word length technique. The primary idea behind this technique is to reduce the data word length of the operand at the software (system) level while not modifying the hardware structures. Decreasing the operand (input) data leads to a reduction of unnecessary switching activities[25]. In this work, the bitwise reduction technique cannot be used independently, whereas the bitwise reduction technique combined with look-ahead techniques to achieve the hybrid approach.

IV. THE PROPOSED DESIGN OF 1ST ORDER IIR FILTER DESIGN

This section shows the implementation of the traditional design and the proposed designs of the 1st-order IIR filter.

A. FPGA Design of IIR Filter

This section introduces the first-order IIR filters based on the FPGA design, which can be regarded as a traditional design that is taken from the related work. This technique depends on reconstructing the suggested filters using the Xilinx System Generator blocksets to implement these systems. The filters equations of these filters are taken from the related works.

The first-order IIR filter is proposed based on FPGA, as shown in Figure 2. This filter has a simple structure that consists of adder, delay, and gain (multiplier) elements. This design is based on the difference equation as follows:

$$\mathrm{y(n)} = \mathrm{ay(n-1)} + \mathrm{x(n)} \quad (1)$$

The coefficient (a) is obtained from [8] by replacing the value of ($a = 0.75$). The equation is expressed as follows:

$$\mathrm{y(n)} = 0.75\mathrm{y(n-1)} + \mathrm{x(n)} \quad (2)$$

The transfer function can be acquired by converting the time domain equation (2) to the z-domain. The transfer functions can be as follows:

$$\mathrm{H(z)} = \frac{1}{1 - 0.75\mathrm{z}^{-1}} \quad (3)$$

From Figure 2, the input signal is an impulse response signal generated by the Discrete Impulse block, which is fed to the GatewayIn block. The Gateway In and the Gateway Out blocks connect between the Simulink block based on the double type and the Xilinx block, which works on a fixed point. The difference equation (2) is represented by two summation terms, and the first term is the $ay(n-1)$ which represents the Add/Sub block with the Delay block where they are used to subtract the current input signal $y(n)$ and the previous input signal $y(n-1)$ and then multiplying with the coefficient (a = 0.75). The Delay block is a D-flip/flop register to store the previous value signal. The second term is the

impulse signal that is summed with the first term. The result will be shown on the Display block, where the To Workspace block stores the output signal's values.

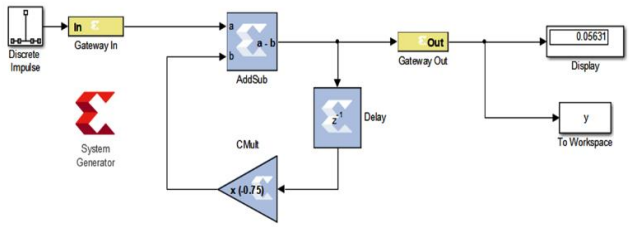

Figure 2: 1st Order IIR Filter Based on the FPGA Method.

B. 1st Order IIR Filter Design Based on LA Pipeline Approach

The Look-Ahead (LA) pipeline approach is used to implement this proposed 1st order IIR filter design, as depicted in Figure 3. This design is based on the main equation (1), where the main idea is to add canceling poles and zeros to the original transfer function to increase the sample rate. In addition, the denominator coefficient related to the transfer function becomes zero.

Consider the same transfer function of equation (1) of the proposed first-order digital IIR filter is shown as follows:

$$\mathrm{H(z)} = \frac{1}{1 - 0.75\mathrm{z}^{-1}} \tag{4}$$

For the number of stages (m) =2, the number of order (N) = 1, and the coefficient (a) = 0.75, the poles and zeros addition equation can be written as follows:

$$\text{Pole \& Zero Addition} = \sum_{\mathrm{i}=0}^{\mathrm{m}-1} \mathrm{r_i z^{-i}} \tag{5}$$

$$\text{Pole \& Zero Addition} = \sum_{\mathrm{i}=0}^{2-1} \mathrm{r_i z^{-i}} = \mathrm{r_0 z^{-0}} + \mathrm{r_1 z^{-1}} \tag{6}$$

According to the equation used to find r_i. There are three possibilities, as depicted in below equation:

$$\mathrm{r_i} = \begin{cases} 0 & \text{for i} < 0 \\ 1 & \text{for i} = 0 \\ \sum_{\mathrm{k}=1}^{\mathrm{N}} \mathrm{a_k r_{(i-k)}} & \text{for i} > 0 \end{cases} \tag{7}$$

$$\mathrm{r_1} = \sum_{\mathrm{k}=1}^{1} \mathrm{a_k r_{(i-k)}} = \mathrm{a_1 r_{(1-1)}} = (0.75).(1) = 0.75 \tag{8}$$

Now, substituting the value of r_0 and the value of r_1 into the equation (6), the resulting equation is:

$$\text{Pole \& Zero Addition} = 1 + 0.75\mathrm{z}^{-1} \tag{9}$$

Finally, the resulting equation (9) is multiplied by both the numerator and denominator of the transfer function equation (4), as shown below:

$$H(z)_{LA} = \frac{1}{(1 - 0.75Z^{-1})} * \frac{(1 + 075z^{-1})}{(1 + 0.75z^{-1})} \tag{10}$$

Now, converting the z-domain to the time domain, the difference equation can be written as follows:

$$y(n) = 0.5625\, y(n-2) + 0.75\, x(n-1) + x(n) \tag{11}$$

Consequently, a pole has been added to the denominator $(1 + 0.75\, z^{-1})$ and a zero has been added to the numerator $(1 + 0.75z^{-1})$ and their magnitudes and directions are identical.

Figure 3 illustrates the implementation of the difference equation (11) for the proposed 1st-order IIR filter based on the LA technique. The input signal is generated using the Discrete Impulse block and fed to the AddSub block. The AddSub block is used to subtract the input signal $(x(n))$ from the feedback signal that multiplied by the coefficient $(a_2 = -0.5625)$ that generated to cancel the zero/pole and delayed by two cycles. Then, the past input signal by one cycle is summed with the other two terms of equation (11) to generate the final results obtained by the Display block.

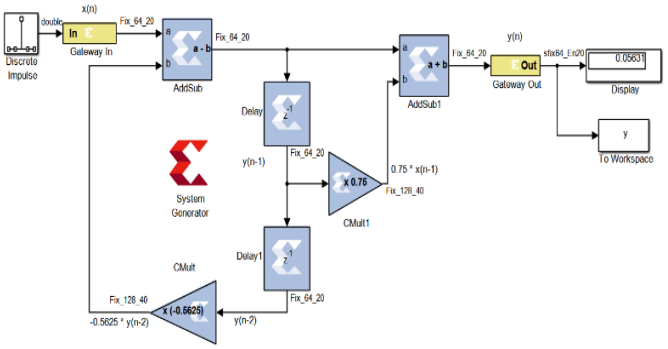

Figure 3: 1st Order IIR Filter Based on Look-Ahead Pipeline Approach.

C. 1st Order IIR Filter Design Based on Parallel Approach

This section uses the parallel technique to implement this suggested design. The essential concept of parallel realization is constructing two parallel filters to increase the execution time. The same equation (3) is used to implement this proposed design, and this equation can be converted into the time domain as follows:

$$y(n) = 0.75y(n-1) + x(n) \tag{12}$$

In order to increase the sample rate by a factor of two, parallel processing is used. It is necessary to design two parallel architectures.

Parallel processing is used to increase the sample rate by a factor of two. Therefore, it is necessary to realize parallel architecture (i.e., the number of parallel paths = L = 2, and the number of stages= M = 2).

Let $n = 2k$, the equation (12) can be written as follows:

$$y(2k) = 0.75y(2k-1) + x(2k) \tag{13}$$

The next signal ($y(2k+1)$) of the equation (13) is expressed as below:

$$y(2k+1) = 0.75y(2k) + x(2k+1) \tag{14}$$

$\because \boldsymbol{L = 2}$, then modifying the equation (14) by subtracting $(\mathbf{2})$ from $(\mathbf{2k+1})$, the resulted equation can be expressed as follows:

$$y(2k-1) = 0.75y(2k-2) + x(2k-1) \tag{15}$$

Substituting equation (15) in equation (13), the obtained equation can be depicted as below:

$$y(2k) = 0.75\big(0.75y(2k-2) + x(2k-1)\big) + x(2k) \tag{16}$$

$$y(2k) = 0.5625y(2k-2) + 0.75x(2k-1) + x(2k) \tag{17}$$

To represent the next state of the signal, let $2k = 2k + 1$

$$y(2k+1) = 0.5625y(2k-1) + 0.75x(2k) + x(2k+1) \tag{18}$$

Figure 4 shows a unique implementation was used to realize the proposed design, where this system is based on the

equations (17) and (18) in which they represent the first parallel filter ($y(2k)$), and the second parallel filter ($y(2k+1)$), respectively. As demonstrated in Figure 4, the implementation of equation (17) begins with the AddSub block and ends with the AddSub1 block. The input signal ($x(2k)$) is directly provided to the AddSub block and subtracted from the feedback delayed signal of two clock cycles by the Delay2 block and then multiplied by the gain (-0. 5625) using the CMult1 block. The second term of equation (17) is represented by the delay signal of one clock cycle and multiplied by the gain = 0.75 using the CMult2 block, and the third term is summed with the other two terms to obtain the final equation. Whereas equation (18) is formed by multiplying equation (17) with the coefficient = 0.75 using the CMult3 block and summed with the next signal ($x(2k+1)$) by the AddSub2 block.

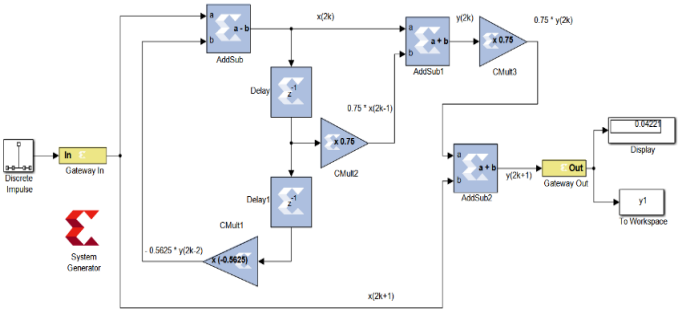

Figure 4: 1st Order IIR Filter Based on Parallel Technique.

D. *1st- Order IIR Filter Design Based on Hybrid Technique*

Figure 5 shows the identical 1st-order IIR filter based on the look-ahead technique designed in section (B); it is also realized in this section. However, this time, the word length reduction (truncation) approach minimizes the area utilized further and accelerates the speed at which this design is executed. This method's fundamental idea is to decrease the binary point bits without modifying the hardware. This technique can be obtained from the GatewayIn block and AddSub blocks configuration window. Basic and Implementation tabs are the two tabs of the GatewayIn window. The Basic tab has an option called the Fixed-point Precession, where the user can set the width of the fixed number and the binary point. Basic, Output, Advanced, and Implementation are the four tabs in the configuration window of the AddSub block. Both Full and User-defined precision options are available on the output tab. Based on the look-ahead, all AddSub blocks in the suggested IIR filter design are set to the full precision option, the default setting., which means that whatever the number of bits for the input feature, these bits are summed with the next block bits, and so on until the end of the system. The multiplication process increases the number of bits that these bits spread across. The option for full precision has been chosen, as shown in Figure 4. The result of the Gateway In block is a fixed number (Fix_64_20) that is 64-bit width and 20-bit binary point, with full precision as the default configuration. The output of the CMult and CMult1 blocks are signed fixed integers (Fix_128_40) with 128-bit and 40-bit binary point widths. A result of the multiplication operation is an increase in the number of bits. After applying the bitwise reduction technique in this design, the user may specify the number of bits with a binary point by changing the settings of all the Gateway In and AddSub blocks to the user-defined precision option. Applying the word length (bitwise) reduction approach to the design is shown in Figure 5.

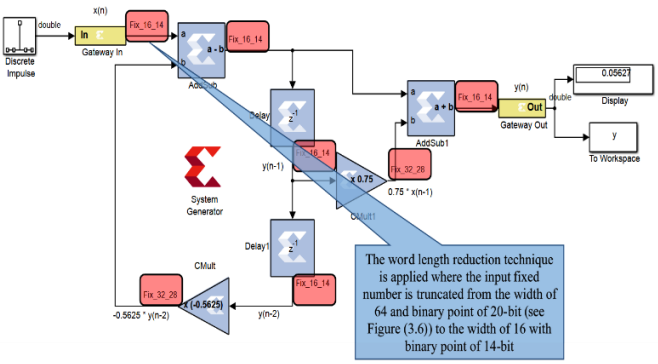

Figure 5: 1st Order IIR Filter Based on Hybrid Technique.

The Gateway In and AddSub blocks have been modified to have a 16-bit width and a 14-bit binary point (Fix_16_14). As can be noticed from the Figure, the number of bits is diminished, and this reduction will propagate through all addition operations. The result of CMult and CMult1 is a fixed-point binary integer with a 32-bit width and a binary point with a 28-bit width (Fix_32_28). When comparing Figure 3 and Figure 5, 24-bit reduction is achieved using this technique.

V. Experimental Results and Discussion:

The suggested design of the 1st order IIR filter was realized utilizing three different techniques. The Xilinx Spartan 3A-3N/XC3S700a/-4/fg484 FPGA platform is used to verify the suggested system design. These designs are implemented using XSG blocks, which can be acquired by configuring ISE 14.7 and MATLAB 2012a. The operation speed increase and resource (area) minimization are targeted in this work. The essential system performance factors are speed, error rate, area, and power dissipation. The speed optimization and error rate percentage will be discussed in detail in this section, and the other two factors (i.e., area utilization and power) will be addressed in the following subsections. The 1st-order IIR filter in [8] is considered as the reference system to compare with the other suggested designs to assess the speed optimization and error rate percentages.

The speed of the traditional design based on FPGA is (16.21) ***nsec,*** whereas the other three proposed 1st-order IIR filters are as follows: (16, 11.8, and 9.85) ***nsec*** that are based on the look-ahead, on the parallel pipelining, and the hybrid techniques, respectively.0shows the speed optimization of the proposed designs. It can be noticed that the hybrid technique achieves the highest speed performance. The highest optimization is due to using two approaches: the bitwise technique and the look-ahead pipeline technique. While the look-ahead method inserts a register between each subunit to lower the critical route delay, the bitwise technique reduces (truncates) the number of bits. As a result, the clock speed or sample speed increases.

TABLE I. Speed Optimization of the 1st Order IIR Filter.

Proposed IIR Filter Design	1st-Order IIR Filter (n sec)	Speed Optimization Percentage
[8]	16.21	-
Based on the Look-Ahead Approach	16	1.30 %
Based on the Parallel Pipeline Technique	11.8	27.21 %
Based on Hybrid Technique	9.85	39.24 %

The second section demonstrates and analyzes the behavior of the four proposed 1st order IIR filter designs that are based on the FPGA method, the look-ahead approach, the parallel pipelining, and the hybrid technique. Figure 6 shows the frequency response (magnitude in dB and phase in degrees) of the four proposed designs which is generated by the built-in function in MATLAB. The error rate percentage is (0) % when comparing the filter based on the FPGA method with the two proposed designs based on look-ahead and parallel pipelining techniques, as shown in Figure 6 (b, c). Whereas, the proposed design based on the hybrid approach has an error rate percentage of (4.3505e-05) %, which can be considered negligible, which means identical matching to the passband, transition band, and stop band, as shown in Figure 6 (d).

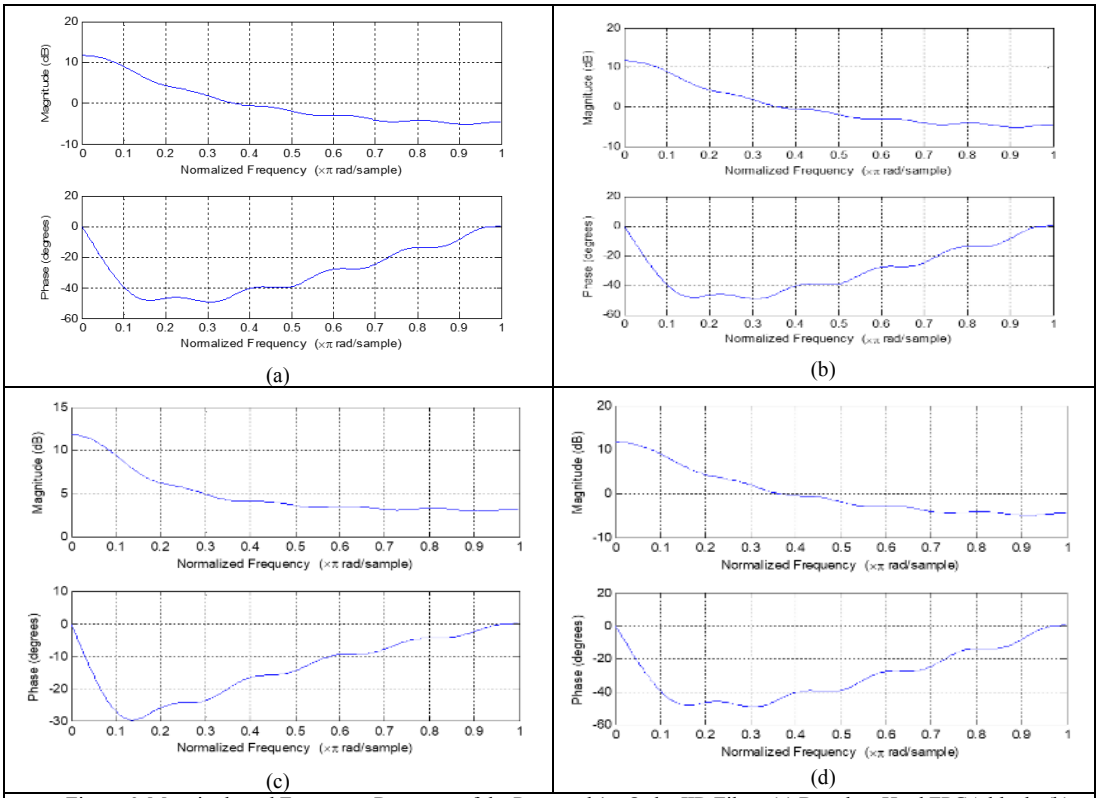

Figure 6: Magnitude and Frequency Response of the Proposed 1st Order IIR Filter: (a) Based on Hard FPGA block, (b) Based on Look-Ahead Approach, (c) Based on Parallel Pipelining Technique, and (d) Based on Hybrid Technique.

Two important subjects will be discussed, studied, and analyzed: the area utilized and the power dissipated in these four proposed filter designs. The first part is resource utilization, shown in TABLE II. illustrates the number of logic elements used by the four suggested 1st-order IIR filter designs. The number of these logic elements can be obtained from the device utilization report generated by the ISE 14.7 package, where the highest number of flip-flops (139) can be found in the Look-Ahead design. In addition, the designs that are based on the Look-Ahead, FPGA method, and parallel techniques have the highest number of LUTs and occupied slices as follows: (541, 363), (314, 193), and (272,197), respectively. In contrast, the hybrid technique has the lowest number of (32) flop-flops, occupied slices of 73, and LUTs of 109, which means that the hybrid technique has minimum area utilization when compared to the other above designs.

The second part discusses the power consumed in the four proposed designs that depicted in Figure 7. From the Figure, the highest power of (119.2, 95.49, and 89.99) mW is consumed by the Look-Ahead technique, design based on the FPGA method, and parallel pipelining designs, respectively. Alternatively, the lowest power dissipated can be obtained by the hybrid design which is (74.02) mW. Power reduction is due to the lowest number of logic elements and flip-flops, which means a small effect of switching activities on the design. Therefore, the hybrid technique achieved the lowest area minimization and power dissipation compared to the other three designs.

TABLE II. RESOURCE UTILIZATION OF THE PROPOSED 1ST ORDER IIR FILTER DESIGNS.

Proposed 1st Order IIR Filter Design	No. of F/F	No. of Slices	No. of LUT
[8]	64	193	314
Based on the Look-Ahead Approach	139	363	541
Based on the Parallel Pipeline Technique	68	179	272
Based on Hybrid Technique	32	73	109

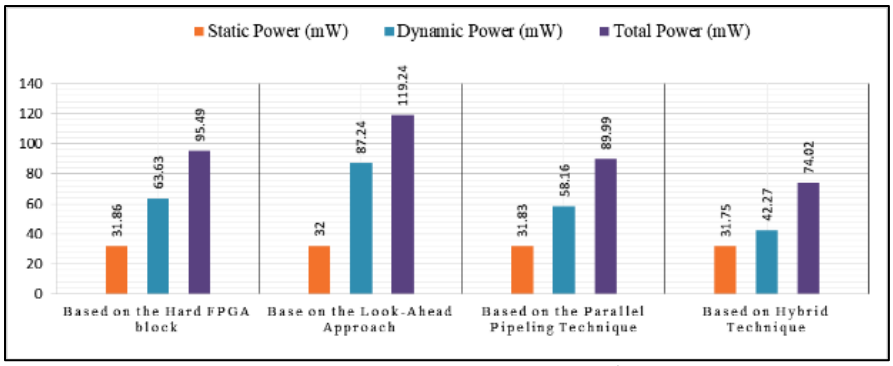

Figure 7: Power Dissipation of the Proposed 1st Order IIR Filters.

VI. CONCLUSION

In this paper, three designs for the first-order IIR filter were proposed. High speed, high accuracy, and area minimization are the essential objectives of this work. Therefore, many techniques are used to achieve these objectives. The proposed designs are based on three techniques: the look-ahead pipelining technique, the parallel technique, and the hybrid technique. It can be concluded that the hybrid technique has the highest speed optimization rate of about (40 %) (low latency) and the highest accuracy, which means it has a low error percentage of (4.3505e-05 %,), and small area utilization in terms of LUTs, occupied slices, and flip-flops which are: (80 %, and 60%) LUTs, and (51%, and 60%) of occupied slices, and (77%, and 53 %) when compared to the other two proposed designs (look-ahead and parallel techniques), respectively. In contrast, the hybrid design has the lowest area utilization of (65 %) of LUTs, (66%) of occupied slices, and (50%) when compared with [8] respectively. The Xilinx Spartan 3A-3N/XC3S700a/-4/fg484 FPGA platform verifies and synthesizes the three suggested designs.

REFERENCES

[1] R. Seshadri and S. Ramakrishnan, "FPGA implementation of fast digital FIR and IIR filters," *Concurr. Comput. Pract. Exp.*, vol. 33, no. 3, p. e5246, 2021.

[2] A. Antoniou, *Digital signal processing*. McGraw-Hill, 2006.

[3] A. Antoniou, *Digital filters: analysis, design, and signal processing applications*. McGraw-Hill Education, 2018.

[4] R. S. H. Istepanian and M. Stojanovic, *Underwater acoustic digital signal processing and communication systems*. Springer, 2002.

[5] R. V Ravi, K. Subramaniam, T. V Roshini, S. P. B. Muthusamy, and G. K. D. Prasanna Venkatesan, "Optimization algorithms, an effective tool for the design of digital filters; a review," *J. Ambient Intell. Humaniz. Comput.*, pp. 1–17, 2019.

[6] K. K. Parhi, "VLSI digital signal processing education," *Conf. Rec. - Asilomar Conf. Signals, Syst. Comput.*, vol. 2, pp. 1303–1308, 1994, doi: 10.1109/ACSSC.1994.471669.

[7] J. Potsangbam and M. Kumar, "Design and implementation of combined pipelining and parallel processing architecture for FIR and IIR filters using VHDL," *Int. J. VLSI Des. Commun. Syst.*, vol. 10, no. 4, pp. 1–16, 2019.

[8] D. Datta and H. S. Dutta, "High performance IIR filter implementation on FPGA," *J. Electr. Syst. Inf. Technol.*, vol. 8, pp. 1–9, 2021.

[9] D. Datta and H. S. Dutta, "Efficient fpga implementation of fir filter using distributed arithmetic," in *Energy Systems, Drives and Automations: Proceedings of ESDA 2019*, 2020, pp. 151–160.

[10] S. H. Abd, I. A. Hashim, and A. S. A. Jalal, "Hardware implementation of deception detection system classifier," *Period. Eng. Nat. Sci.*, vol. 10, no. 1, p. 151, Dec. 2021, doi: 10.21533/pen.v10i1.2594.

[11] A. Sharma and S. Kumar, "VLSI implementation of pipelined FIR filter," *Int. J. Innov. Res. Electr. Electron. Instrum. Control Eng.*, vol. 1, no. 5, 2013.

[12] R. Kaur, A. Raman, H. Singh, and J. Malhotra, "Design and Implementation of High Speed II Rand FIR Filter using Pipelining," *Int. J. Comput. Theory Eng.*, vol. 3, no. 2, pp. 292–295, 2011, doi: 10.7763/ijcte.2011.v3.320.

[13] Y.-C. Tsao and K. Choi, "Area-efficient VLSI implementation for parallel linear-phase FIR digital filters of odd length based on fast FIR algorithm," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 59, no. 6, pp. 371–375, 2012.

[14] M. Hatamian and K. K. Parhi, "An 85-MHz Fourth-Order Programmable IIR Digital Filter Chip," *IEEE J. Solid-State Circuits*, vol. 27, no. 2, pp. 175–183, 1992, doi: 10.1109/4.127340.

[15] Y. Gu and K. K. Parhi, "High-speed architecture design of Tomlinson–Harashima precoders," *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 54, no. 9, pp. 1929–1937, 2007.

[16] K. K. Parhi and D. G. Messerschmitt, "Pipeline interleaving and parallelism in recursive digital filters. I. Pipelining using scattered look-ahead and decomposition," *IEEE Trans. Acoust.*, vol. 37, no. 7, pp. 1099–1117, 1989.

[17] Y. C. Lim, "A new approach for deriving scattered coefficients of pipelined IIR filters," *IEEE Trans. signal Process.*, vol. 43, no. 10, pp. 2405–2406, 1995.

[18] Y.-L. Chen, C.-Y. Chen, K.-Y. Jheng, and A.-Y. Wu, "A universal look-ahead algorithm for pipelining IIR filters," in *2008 IEEE International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, 2008, pp. 259–262.

[19] A. K. Shaw and M. Imtiaz, "A general Look-Ahead algorithm for pipelining IIR filters," in *1996 IEEE International Symposium on Circuits and Systems. Circuits and Systems Connecting the World. ISCAS 96*, 1996, vol. 2, pp. 237–240.

[20] J. Potsangbam and M. Kumar, "Design and implementation of combined pipelining and parallel processing architecture for FIR and IIR filters using VHDL," *Int. J. VLSI Des. Commun. Syst.*, vol. 10, no. 4, pp. 1–16, 2019, doi: 10.5281/zenodo.4545404.

[21] H. T. Assafli, I. A. Hashim, and A. A. Naser, "Advanced Encryption Standard (AES) acceleration and analysis using graphical processing unit (GPU)," *Appl. Nanosci.*, vol. 13, no. 2, pp. 1245–1250, Feb. 2023, doi: 10.1007/s13204-021-01985-3.

[22] S. Balasubramaniam and R. Bharathi, "Performance analysis of parallel FIR digital filter using VHDL," *Int. J. Comput. Appl.*, vol. 39, no. 9, pp. 1–6, 2012.

[23] K. K. Parhi and D. G. Messerschmitt, "Pipeline interleaving and parallelism in recursive digital filters. II. Pipelined incremental block filtering," *IEEE Trans. Acoust.*, vol. 37, no. 7, pp. 1118–1134, 1989.

[24] D. A. Parker and K. K. Parhi, "Low-area/power parallel FIR digital filter implementations," *J. VLSI signal Process. Syst. signal, image video Technol.*, vol. 17, no. 1, pp. 75–92, 1997.

[25] K. Han, B. L. Evans, and E. E. Swartzlander, "Low-power multipliers with data wordlength reduction," in *Proc. Asilomar Conference on Signals, Systems and Computers (ACSSC)*, 2005, pp. 1615–1619.