High-Speed FPGA Realiztion of Infinite Immpulse Response Filter

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

2023 International Conference on Engineering Applied and Nano Sciences (ICEANS) | 979-8-3503-4197-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICEANS58413.2023.10630489

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: 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 the 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:

Numerous applications of digital signal processing have been established, including robotics and instrumentation as well as voice, audio, image, and biological signal processing [1]. There is at least one DSP module in every modern communication system or architecture. Utilizing the signal processing technology known as filtering is the most effective way to remove noisy signals. 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 utilized operations in applications of digital signal and image processing. 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]. This work aims to design an IIR filter with 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.

II. LITERATURE SURVEY

A survey of previous studies on increasing the speed of IIR filters has been made, which will be introduced in the next paragraph. Then, various methods and techniques of speed optimization can be utilized to be applied tfilter design, such as the traditional method, look-ahead pipelining, parallel pipelining, bitwise reduction technique, and the hybrid technique. The suggested work in [1] provides a detailed introduction to traditional Moving Average (MA) FIR filters, fast MA FIR filters using look-ahead arithmetic, conventional IIR filters using the Combination of Integrator and Comb Sections (CIC) method, and fast IIR filters using look-ahead arithmetic. 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 utilizing the level 1 and level 2 look-ahead approaches, respectively, with increases in the utilized area by 67% and 37.5% logic elements. At the same time, they obtained speed optimizations of 22% and 11% for the 2nd-order IIR filter based on look-ahead level 1 and look-ahead level 2, respectively, with increases in the logic elements of 22% and 61%. The authors in [7] suggested a novel Floating Point Processing Element (FPPE) architecture design with fewer logical components and registers. A partially folded arithmetic function architecture is modeled for designing an IIR filter by FPPE and executed on an FPGA with an efficient area. The suggested system design performance is compared with the conventional IIR filter. The author obtained a 22% improvement in speed with increasing resource utilization of 10% of the flip-flops and LUTs. Synthesis results prove that the proposed IIR filter design offers an effective area compared with the present work. The modules are implemented on Xilinx FPGAs and designed in Verilog. The primary goal of R. M. Raj et al. in [8] is to use several filtering methods to eliminate redundant noise (baseline wandering), despite the fact that it is impossible to misinterpret the necessary details more than defined in standard guidance. The baseline wandering effect from an electrocardiogram (ECG) signal is removed by using a DC bias elimination filter with different design structures, such as conventional, pipelined, look-ahead, and clustered look-ahead, to produce better output (clock speed). The filters were implemented on an Altera FPGA device. The resource usage is increased by 35% of flip-flops, 71% of occupied slices, and LUTs), the speed optimization rates are 25% and 17%; and the power optimization is low. All these performances are achieved by comparing the pipeline and look-ahead pipeline with the clustered look-ahead pipeline designs.

The researchers in [9] aimed to solve the problem of the multiplier-less design of 2nd-order IIR filters with the lowest number of adders. When all multiplications are replaced with bit shifts and additions, the authors create a robust direct-form filter

with hardware-aware fixed-point coefficients that produces minimal adders. Synthesis was performed using VIVADO v2019.1 for a Kintex 7 device. Regarding state-of-the-art three-stage filter system design methodologies, the first step delivers a 42% decrease in the number of LUTs and a 21% enhancement in delay. The 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 2nd-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.

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. Lower hardware costs than those attained in the direct and transposed forms.

A. Distributed Arithmetic (DA)

Due to the large memory requirements of multiplication, distributed arithmetic (DA) is a hardware-efficient way of storing pre-computed values in LUTs of FPGAs. [10]-[12]. This technique is not used in this work because the used FPGA kit does not have enough memory, whereas this technique needs a huge amount of memory to store the coefficients.

B. Pipeline Approach

In most DSP systems, pipelining is an implementation approach that increases critical path speed by concurrently executing a stream of instructions while lowering critical route latency [13]. 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 time unit [13]-[16]. 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. [17]. There are different types of LA methods [18-20], 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 [21]. 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

Parallel processing is an implementation technique in which multiple outputs are computed parallelly in a clock period. In parallel processing, the hardware for the original serial system is duplicated, and the resulting system is a Multiple Input Multiple Outputs (MIMO) parallel system. Fig. 1 shows the block diagram for converting a Single Input Single Output (SISO) system to a MIMO system [22].

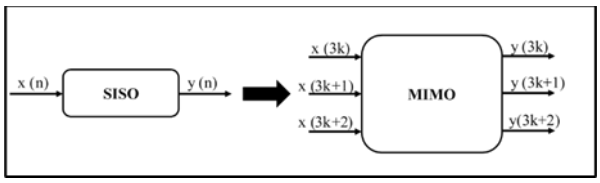

Fig. 1. Sequential System to 3-Parallel System[22].

Parallel processing is a reliable approach since it may be used to boost the throughput of a digital filter while decreasing its power consumption [26],[27]. The hardware complexity of the IIR filter in a 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 [25]. 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 data word length technique is another name for the bitwise algorithm. The fundamental idea behind this technique is to shorten the operand's data word at the software (system) level without altering the hardware structures. Reducing the operand (input) data results in a decrease in unnecessary switching actions. In order to successfully reduce power dissipation without affecting circuit performance, switching operations should be minimized [26]. In this work, the bitwise reduction technique cannot be used independently, whereas the bitwise reduction technique is combined with look-ahead techniques to achieve the hybrid approach.

IV. THE PROPOSED DESIGN OF 2ND ORDER IIR FILTER DESIGN

This section shows the implementation of the traditional design and the proposed designs of the 2nd order IIR filter.

A. Second Order IIR Filter Based on FPGA

The design is based on FPGA with Direct Form II structure according to the " (1)" and "(2)" :

$$w(n) = x(n) - a_1 w(n-1) - a_2 w(n-2) \quad (1)$$

$$y(n) = b_0 w(n) + ab_1 w(n-1) + b_2 w(n-2) \quad (2)$$

The parameters of the proposed filter are obtained from [27], where the sampling frequency (f_s) = 1 MHz, the cut-off frequency (fc) = 50 Hz is utilized, the calculated coefficients are as follows: $a_0 = 1,\ a_1 = -1.9843, a_2 = 0.9843, b_0 = 1, b_1 = 2, b_2 = 1,$ and the gain input = 0.000239.

Therefore, the transfer function of this proposed filter can be shown as follows:

$$H(z)_{LA} = \frac{1}{(1 - 0.75Z^{-1})} * \frac{(1 + 075z^{-1})}{(1 + 0.75z^{-1})} \quad (3)$$

By converting "(3)" from the z-domain to the time domain, the difference equation can be expressed as follows:

$$w(n) = 0.000239 * \big(x(n) + 1.9843\, w(n-1) - 0.9843 w(n-2)\big) \quad (3)$$

$$y(n) = w(n) + 2\, w(n-1) + w(n-2) \quad (4)$$

Fig shows the proposed design which is constructed from six CMult blocks, four AddSub blocks, and two Delay blocks. The CMult block has a gain coefficient of (0.000239) multiplied with the entire difference "(3)" and then subtracted through the AddSub block from the recursive signal $(-1.9843\, w(n-1))$ that by the delay block which is a D-flip flop register. The output of the AddSub block is supplied to the AddSub1 block and subtracted from the feedback signal $(0.9843 * w(n-2))$ that delayed by two cycles through the Delay1 block. The output of the AddSub1 block presents the first difference "(3)". Finally, the outputs of the CMult3 and CMult4 blocks represent the $(w(n))$ and $(2 * w(n-1))$ which they are summed using the AddSub2 block and then the output of CMult block, which represents the $(w(n-2))$ is fed to the AddSub3 block to form the second difference "(3)". The Display block is used to display the value of the output signal $(y(n))$.

B. *Second Order IIR Filter Design Based on Look-Ahead Approach*

Fig. 3 shows the same approach is applied to this design but with a different order, which is 2nd order. The same concept is used in this design that depends on the same main equation that mentioned in section IV and the same transfer function is taken for this proposed design can be written as follows:

$$H(z) = \frac{0.000239(1 + 2z^{-1} + 1z^{-2})}{(1 - 1.9843z^{-1} + 0.9843z^{-2})} \quad (5)$$

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

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

By applying the equation of r_i , the r_1 can be obtained as follows:

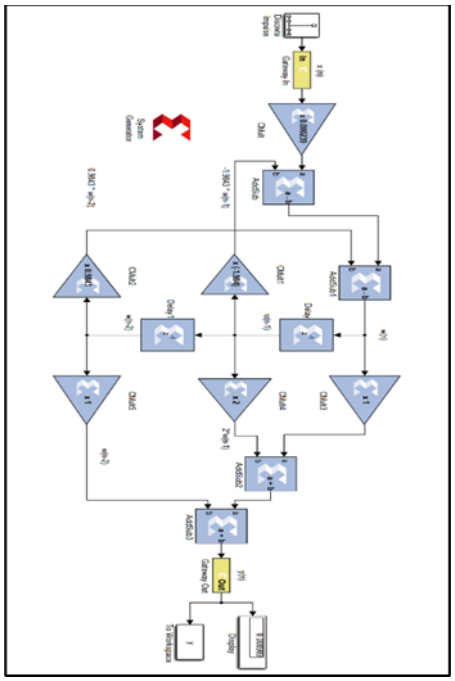

Fig. 2: 2nd Order IIR Filter Based on the FPGA Method.

$$r_1 = \sum_{k=1}^{2} a_k r_{(i-k)} = a_1 r_{(1-1)} + a_2 r_{(1-2)} = (1.9843).(1) + (-0.9843).(0) = 1.9843 \quad (7)$$

Now, substituting the value of r_0 and the value of r_1 into the"(3)", the resulting equation is:

$$\text{Pole \& Zero Addition} = 1 + 1.9843z^{-1} \quad (8)$$

Finally, the resulting "(3)" is multiplied by both the numerator and denominator of the transfer function "(3)", as shown below:

$$H(z)_{LA} = \frac{0.000239(1 + 2z^{-1} + z^{-2})}{(1 - 1.9843z^{-1} + 0.9843z^{-2})} * \frac{(1 + 1.9843z^{-1})}{(1 + 1.9843z^{-1})} \quad (8)$$

The difference equation is obtained by converting the transfer function "(3)" to the time domain as expressed below:

$$y(n) = \big[0.000239\big(x(n) + 2x(n-1) + x(n-2)\big) + 1.9843y(n-1) - 0.9843y(n-2)\big] * [x(n) + 1.9843\, x(n-1) - y(n) - 1.9843y(n-1)] \quad (9)$$

The proposed design is built on the decomposition CLA pipeline method, which factorized the transfer function "**Error! Reference source not found.**" into two sections as d epicted in the transfer function "(8)" and illustrated in Fig. 3. The first section represents the second-order section, which starts from the CMult block and ends with the AddSub3 block. The CMult block is the gain = 0.000239 that multiplied by the numerator of the first section and then subtracted by the AddSub block from the previous delayed signal that multiplied by the gain = 1.9843 of the CMult1. The signal that is delayed by two cycles through the Delay block and gained by (-0.9843) by the CMult2 block is subtracted from the output of the AddSub

block by using the AddSub1 block. The summation AddSub2 and AddSub3 blocks represent the feedforward signals of the numerator of the first section. Consequently, the output of the AddSub3 block is fed to the second section. This section represents the first order of the transfer function "(3)". The denominator of this section is represented by the subtraction of the previous signal that is multiplied by (1.958) through the CMult6 block and delayed one clock cycle through the Delay2 block. In contrast, the numerator is the addition of the current input signal with the delayed input signal that is delayed by two clock cycles and multiplied by the coefficient (1.958) through Cmult 7, and by the Display block shows the final result.

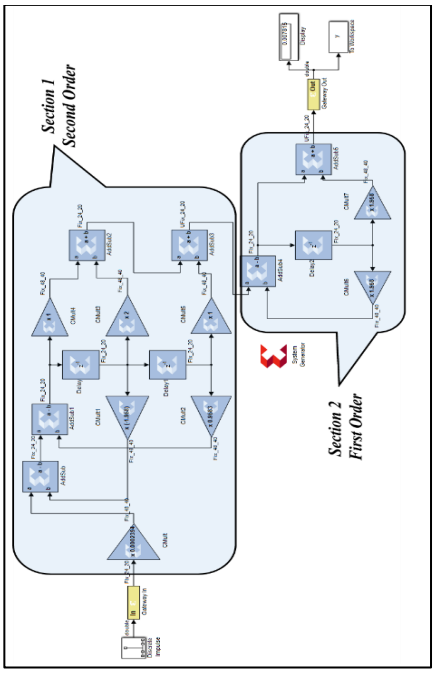

Fig. 3. Proposed 2nd Order IIR Filter Based on Look-Ahead Approach.

C. *Second Order IIR Filter Design Based on Parallel Approach*

The suggested system design is built on the parallel technique, where the primary concept of the parallel realization is to construct two parallel filters to increase the execution time. The same exact "(3)" is used to implement this proposed design and if we let $G = 0.000239, b_1 = a = 2,\ a_1 = b = 1.9843$, and $a_2 = c = 0.9843$, the "(3)" becomes as below:

$$H(z) = \frac{G(1 + az^{-1} + z^{-2})}{(1 - bz^{-1} + cz^{-2})} \qquad (10)$$

The "(10)" can be converted from the z-domain into the time domain as follows:

$$y(n) = G\big(x(n) + ax(n-1) + x(n-2)\big) + by(n-1) - cy(n-2) \qquad (11)$$

Parallel processing increases 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 "(3)" can be written as follows:

$$y(2k) = G\big(x(2k) + ax(2k-1) + x(2k-2)\big) + by(2k-1) - cy(2k-2) \qquad (13)$$

The next state signal ($y(2k+1)$) of the " **Error! Reference source not found.**"

is expressed as below:

$$y(2k+1) = G\big(x(2k+1) + ax(2k) + x(2k-1)\big) + by(2k) - cy(2k-1) \qquad (12)$$

$\because L = 2$, then modifying the "(3)" by subtracting (2) from $(2k+1)$, the resulted equation can be expressed as follows:

$$y(2k-1) = G\big(x(2k-1) + ax(2k-2) + x(2k-3)\big) + by(2k-2) - cy(2k-3) \qquad (15)$$

Now, substituting the "(3)" in the "(13)", the obtained equation can be depicted as below:

$$y(2k) = G\big(x(2k) + ax(2k-1) + x(2k-2)\big) + b\big(G\big(x(2k-1) + ax(2k-2) + x(2k-3)\big) + by(2k-2) - cy(2k-3)\big) - cy(2k-2) \qquad (16)$$

$$y(2k) = G\big(x(2k) + 3.9843x(2k-1) + 4.9686x(2k-2) + 1.9843x(2k-3)\big) + 2.9527y(2k-2) - 1.95413y(2k-3) \qquad (17)$$

To simplify and clear the concept of this proposed design, the difference "(3)" transformed into the z-domain as follows:

$$H(z) = \frac{G(1 + 3.9843z^{-1} + 4.9686\,z^{-2} + 1.9843\,z^{-3})}{(1 - 2.9527z^{-2} + 1.95314z^{-3})} \qquad (18)$$

The "(3)" can be decomposite as follows:

$$H(z) = \frac{G(1 + 2z^{-1} + z^{-3})}{(1 - 1.958z^{-1} + 0.9583z^{-2})} * \frac{1 + 1.958z^{-1}}{1 + 1.958z^{-1}} \qquad (19)$$

Now, let convert the transfer function "(19)" to the difference equation in order to represent the next state of the signal; let $(2k = 2k + 1)$

$$y(2k+1) = G\big(x(2k+1) + ax(2k) + x(2k-1)\big) + b\big(G\big(x(2k) + ax(2k-1) + x(2k-2)\big) + by(2k-1) - cy(2k-2)\big) - cy(2k-1) \qquad (13)$$

The final difference equation can be written as follows:

$$y(2k+1) = G\big(x(2k+1) + 3.984x(2k) + 4.9686x(2k-1) + 1.9843x(2k-2)\big) + 2.9527y(2k-1) - 1.95314y(2k-2) \qquad (21)$$

Figure 4 shows a unique implementation that was used to realize the proposed design, where this system is based on the "(16)" and "(21)" in which they represent the first parallel filter ($y(2k)$), and the second parallel filter ($y(2k+1)$), respectively.

The essential equation to build this proposed design is the transfer function "(19)" , which decomposes the "(18)" . The proposed design of the 2nd order IIR filter based on the parallel is divided into two parts, as illustrated in Fig. . The first part is $\left(\frac{G(1+2z^{-1}+z^{-3})}{(1-1.958z^{-1}+0.9583z^{-2})}\right)$ which is represented by the AddSub block and ends at the AddSub3 block. The input signal is multiplied by the coefficient (0.000239) using the CMult block and then fed to the AddSub block to be subtracted from the recursive signal that is delayed by the Delay block of one clock cycle and multiplied by the gain (-1.958) using CMult block. The output is fed to the AddSub1

block to be subtracted from the feedback signal that is delayed by two clock cycles and multiplied by the coefficient (0.9583) using the CMult2. The feedforward signals x(2k) and x(2k − 1) are represented by the CMult5 block, and Delay with CMult3 blocks, respectively, whereas the third term of the numerator is represented by the Delay1 block with CMult4 block that has been summed with the other two terms of the numerator. On the other hand, the same system is built in parallel to the first part system, which means it is a copy of the first part. The parallel system starts from the AddSub6 block and ends at the AddSub4 block with the same work procedure explained above. The second part of this proposed design is $\left(\frac{1+1.958z^{-1}}{1+1.958z^{-1}}\right)$, which represents the multiplication with the conjugate of the first part of the equation. This part is represented by the output of the AddSub3 block to the AddSub10 block. The output of the AddSub3 block is subtracted from the recursive signal that is delayed by the Dalay2 block of one clock cycle and multiplied by the coefficient (1.958) using the CMult6 block. Whereas the x(2k) signal is represented directly to the AddSub10 block by the coefficient (1.958) using the CMult7 block, both are summed using the AddSub10 block to obtain the final result using the Display block. Otherwise, another part is constructed in parallel to the second part, which starts from the output of the AddSub4 block and ends with the AddSub11 block, which is the copy of the second part system and works with the same procedure mentioned above and the final result is shown using the Display1 block.

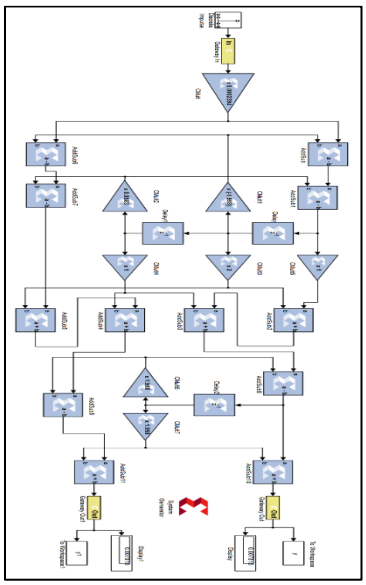

Fig. 4. Proposed 2nd Order IIR Filter Based on Parallel Approach.

V. Experimental Results and Discussion

The suggested design of the 2nd-order IIR filter was realized utilizing three different methods. The suggested design is confirmed to utilize the Xilinx Spartan 3A-3N/XC3S700a/-4/fg484 FPGA board. Utilizing XSG blocks, which can be found in the initial configuration of MATLAB 2012a and ISE 14.7, these designs are executed. The resource (area) minimization is targeted in this research. 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. In order to evaluate the speed optimization and error rate percentages, the 2nd order IIR filter in [27] is considered the reference system to compare with the other suggested system designs.

Table I demonstrates the speed performance and speed optimization performance for the proposed 2nd-order IIR filter designs, in which the proposed designs based on the FPGA method will be compared to the other three designs in terms of speed. The designs based on the FPGA method, look-ahead, and parallel pipeline techniques have approximately the same speed between (22) nsec and (23) nsec. This means they have the lowest speed optimization percentage. At the same time, the hybrid technique achieved the highest speed optimization of 18.13 %. The critical paths are reduced when using the hybrid technique, resulting in high-speed performance.

.

TABLE I. SPEED OPTIMIZATION OF THE 2ND ORDER IIR FILTER

Proposed IIR Filter Design	2nd-Order IIR Filter (n sec)	Speed Enhancement Percentage
[27]	22.5	-
Based on the Look-Ahead Approach	21.36	5.07 %
Based on the Parallel Pipeline Technique	21.76	3.29 %
Based on Hybrid Technique	18.42	18.13 %

Fig illustrates the frequency responses of the four proposed designs of the 2nd-order IIR filter. The evaluation will depend on comparing the proposed filter based on the FPGA method with the other three designs. The comparison shows identical matching in error rate percentage of (0) % when using the look-ahead approach and slightly high error rate percentages of 1.5049 % and 1.5053 % in the passband, transition band, and stop band when compared to the parallel pipeline technique, and the hybrid technique, respectively.

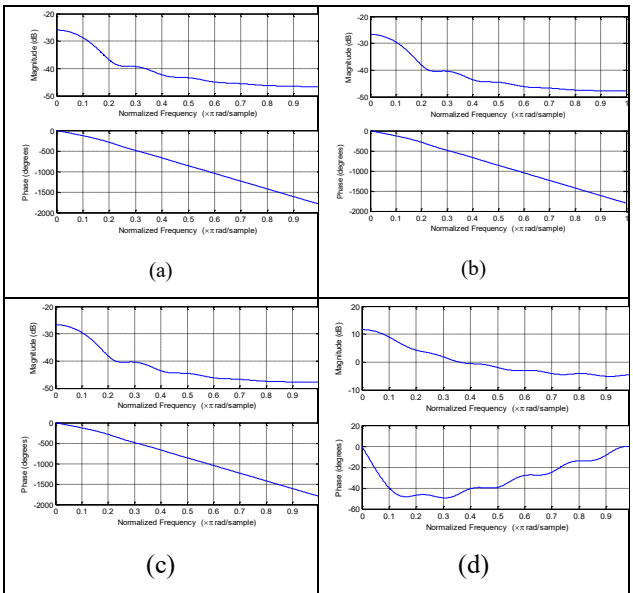

Fig. 5. Magnitude and Frequency Response of the Proposed 2nd Order IIR Filter: (a) Based on Hard FPGA block, (b) Based on Look-Ahead Approach, (c) Based on Parallel Pipelining Technique, and (d) Based on Hybrid Technique.

The first section discusses the resource utilization of each design, but the difference here is that all the proposed designs are 2nd order IIR filters. The device utilization report tracks the resource utilization of the suggested system design. The proposed design built on the FPGA method has the highest number of flip-flops which is (141). In contrast, the other three designs have half the number of flip-flops: (75) flip-flops for the look-ahead, the hybrid techniques for each one, and (72) flip-flops for the parallel technique, as shown in table II. In addition, the highest resource utilization in terms of occupied slices and LUTs can be found in the suggested system designs built on the FPGA method, parallel pipeline, look-ahead, and hybrid techniques, which can be depicted as follows: (2026 of slices and 3264 of LUTs), (777 of slices and 1284 of LUTs), and (687 of slices and 1088 of LUTs), respectively. The hybrid technique uses the least amount of area compared to other designs (only (280) occupied slices and (411) LUTs), hence it is successful in lowering the number of logic elements.

TABLE II. RESOURCE UTILIZATION OF THE PROPOSED 2ND ORDER IIR FILTER DESIGNS.

Proposed 2nd Order IIR Filter Design	No. of F/F	No. of Slices	No. of LUT
[27]	141	2026	3264
Based on the Look-Ahead Approach	75	687	1088
Based on the Parallel Pipeling Technique	72	777	1,284
Based on Hybrid Technique	75	280	411

The second section shows the power consumption of 2nd-order IIR filter designs obtained from the power reports summary of the experimental results, as illustrated in Fi. The highest power dissipation achieved in the designs that are based on the FPGA method, look-ahead, and parallel pipeline techniques by (301, 225.96, and 156) mW, respectively. In comparison, the hybrid technique achieved a power of (112) mW, which is the lowest dissipated power for the above designs. The primary cause is that it consumes a small logic elements, as depicted in table II.

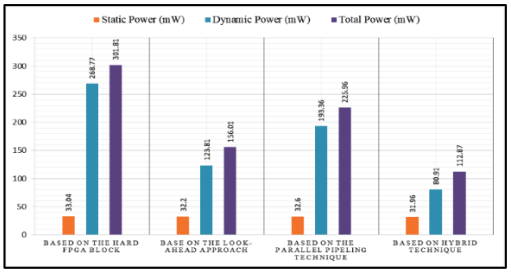

Fig. 6.: Power Dissipation of the Proposed 2nd Order IIR Filters.

VI. CONCLUSIONS

In this paper, three designs for the second-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 19% (low latency) and the highest accuracy, which means it has a low error percentage of 1.5053 %, and small area utilization in terms of LUTs and occupied slices, which are: 62 %, and 68% LUTs and 59%, and 64% of occupied slices, respectively, when compared to the other two proposed designs (look-ahead and parallel techniques). In contrast, the hybrid design has the lowest area utilization of 87.5 % of LUTs, 86% of occupied slices, and 47% of flip-flops when compared with [27] respectively. The three proposed designs are verified and synthesized using the Xilinx Spartan 3A-3N/XC3S700a/-4/fg484 FPGA platform.

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] R. NALLATHAMBI, S. VEERANA, and D. PRABHAKARAN, "Design of area-efficient IIR filter using FPPE," *Turkish J. Electr. Eng. Comput. Sci.*, vol. 27, no. 3, pp. 2321–2330, 2019.

[8] R. M. Raj, V. Rajesh, S. Saravanan, M. S. P. Balaji, and V. Elamaran, "Baseline wandering noise removal using high-speed IIR filters with an FPGA implementation," in *Microelectronic Devices, Circuits and Systems: Second International Conference, ICMDCS 2021, Vellore, India, February 11-13, 2021, Revised Selected Papers 2*, 2021, pp. 55–65.

[9] R. Garcia, A. Volkova, M. Kumm, A. Goldsztejn, and J. Kühle, "Hardware-aware Design of Multiplierless Second-Order IIR Filters with Minimum Adders," *IEEE Trans. Signal Process.*, vol. 70, pp. 1673–1686, 2022.

[10] 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.

[11] S. A. White, "Applications of distributed arithmetic to digital signal processing: A tutorial review," *IEEE Assp Mag.*, vol. 6, no. 3, pp. 4–19, 1989.

[12] S. Akhter, S. Kumar, and D. Bareja, "Design and analysis of distributed arithmetic based FIR filter," in *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, 2018, pp. 721–726.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] J. Potsangbam and M. Kumar, "D ESIGN A ND I MPLEMENTATION O F C OMBINED P IPELINING A ND P ARALLEL P ROCESSING A RCHITECTURE F OR FIR A ND IIR F ILTERS U SING," vol. 10, no. 4, pp. 1–16, 2019.

[23] 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

[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. 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.

[26] 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.

[27] A. Kapić, R. Sarić, S. Lubura, and D. Jokić, "FPGA-based Implementation of IIR Filter for Real-Time Noise Reduction in Signal."