Some Studies on Efficient FIR Filter Design using Radix-2 Booth's Algorithm

By
NEELESH BISWAS
Registration No.: 154117 of 2020-2021
Exam Roll No.: M6VLS23007

Under the Guidance of:
Prof. P. VENKATESWARAN

Thesis submitted in partial fulfillment of the requirements
for the award of the Degree of Master of Technology in
VLSI Design and Microelectronics Technology

Department of Electronics and Telecommunication Engineering
Jadavpur University
Kolkata-700032

June 2023

DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMIC ETHICS

I hereby declare that this thesis contains a literature survey and original work by the undersigned candidate, as part of his Master of Technology in VLSI Design and Microelectronics Technology.

All information in this document have been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name of Candidate: NEELESH BISWAS

Exam Roll No: M6VLS23007

Thesis Title: Some Studies on Efficient FIR Filter Design using Radix-2 Booth's Algorithm.

Signature of the Candidate: _____

Date:

FACULTY OF ENGINEERING AND TECHNOLOGY
JADAVPUR UNIVERSITY

CERTIFICATE OF RECOMMENDATION

I hereby recommend that this Thesis prepared by **NEELESH BISWAS** entitled "**Some Studies on Efficient FIR Filter Design using Radix-2 Booth's Algorithm"** under my supervision be accepted for partial fulfillment of the requirements for the award of the degree of Master of Technology in VLSI Design and Microelectronics Technology.

Dr. P. Venkateswaran
Professor
Dept. of Electronics and Tele-Communication Engineering
Jadavpur University, Kolkata-700032

Dr. Manotosh Biswas
Professor & Head
Dept. of Electronics and Tele-Communication Engineering
Jadavpur University, Kolkata-700032

Dean, Faculty of Engineering & Technology
Jadavpur University, Kolkata-700032

FACULTY OF ENGINEERING AND TECHNOLOGY
JADAVPUR UNIVERSITY

CERTIFICATE OF APPROVAL

The foregoing Thesis is hereby accepted as a credible study of an engineering subject carried out and presented in a manner that is satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by the approval, the undersigned don't necessarily endorse or approve any statement made, opinion expressed, or conclusion drawn therein but approve the Thesis only for the purpose for which it is submitted.

Committee on Final Examination for

Evaluation of the Thesis of: NEELESH BISWAS

Exam Roll No: M6VLS23007

_____

_____

Signature of Examiners

ACKNOWLEDGEMENT

I express my deep sense of gratitude and indebtedness to my Thesis Supervisor, Prof. P. Venkateswaran, Department of Electronics & Tele-communication Engineering (ETCE), Jadavpur University (JU), Kolkata, for providing me the opportunity to carry out the Thesis work. I am grateful to him for the valuable insights and suggestions that he gave me throughout my M.Tech Thesis work.

Further, I express my special thanks to Dr. Supriya Dhabal, Assistant Professor, Dept. of Electronics & Communication Engineering (ECE), Netaji Subhash Engineering College, Garia, Kolkata, for his constant and effortful guidance by providing necessary information whenever required.

I would like to thank, Prof. Sayan Chatterjee, Coordinator of IC Centre and Course Co-ordinator of M.Tech VLSI Design and Microelectronics, Dept. of ETCE, Jadavpur University for his compassionate approach. I would like to express my sincere gratitude to all the teaching and non-teaching staffs of the department for providing the necessary support.

I would also like to thank Prof. Manotosh Biswas, Head, Dept. of ETCE, Jadavpur University for his compassionate approach during this Thesis work. Special Thanks to Dr. Aditi Chatterjee, Mrs. Raina Modak Aich, Mr. Sajal Sarkar, Mr. Sk Babul Akhter, Mrs. Tania Paul Das, Mrs. Roshni Chakrabarti and the Professors of my alma mater, Dept. of ECE, Siliguri Institute of Technology for giving me valuable help and insights to carry out my Thesis work. I am also thankful to Mr. Goutam Barman for his constant support throughout my Thesis work.

Last but not the least, this work would not have been possible without the support and encouragement of my parents, my brother, my classmates, and near and dear ones.

_____

NEELESH BISWAS

Computer Networking Lab (T-3-16A)
Dept. of ETCE, Jadavpur University
Kolkata-700032. West Bengal.

Date:

ABSTRACT

Booth's multiplier is known for its versatility due to the signed bit multiplication. For the design of Finite Impulse Response (FIR) Filters, it plays a central role in increasing speed and reducing the power consumption. The work presented in this Thesis is employing Radix-2 Booth's algorithm for the design of FIR Filters which is primarily classified into two broad categories. The first category deals with the design of a traditional Type-I FIR Filter which is compared with the existing algorithms like Parks-McClellan (PM) and the Improved Sine Cosine Algorithm (ISCA). It is observed that as compared to the PM and ISCA algorithms, the proposed Booth's multiplier based design requires zero memory and zero Digital Signal Processing (DSP) blocks which in turn leads to a reduced cost and better hardware utilization of Field Programmable Gate Array (FPGA) board. In second category, two designs of a Parallel 2-FIR Filter and an Area-Efficient 2-FIR Filter are presented. The hardware complexity and power utilization of these two filters are compared among themselves, and also with an existing work designed via Xilinx 14.2 Spartan 3E Starter Board XC3S500E chips. Based on the simulation results, it is observed that the proposed Area-Efficient FIR Filter does not require RAMB16s when compared with Parallel 2-FIR Filter and other existing works available in the literature. The Parallel 2-FIR Filter was designed using the Filter Design and Analysis (FDA) toolbox of MATLAB Platform followed by the generation of VHDL file while the Area-Efficient Filter was designed completely using VHDL.

CONTENTS

List of Figures

Table no	List of Tables	Page No

Chapter 1: Introduction

1.1 Preamble

We use various kinds of filters in our daily lives, homes, offices, and schools like the water filter, air filters, etc. This point to the fact that filters are important for us to live a good standard of living. Similarly, in the field of electronics, filters are very essential elements. The term 'filtering' usually refers to the quality or process of blocking something out while allowing some other things to pass through. Electronic filters refer to the devices that upon receiving input signals, allow only the components within a certain range of frequencies to transmit and attenuate the rest of the components outside that frequency range. Electronic filters have been categorized in the literature based on various criteria. Depending upon the type of components used in construction, filters are classified as — active and passive. Based on rating frequency ranges filters may be classified as Audio Frequency (AF) or Radio Frequency (RF). According to the nature of impulse response, filters can be categorized as— Finite Impulse Response (FIR) or Infinite Impulse Response (IIR) [1-3]. As opposed to IIR filters, FIR filters are non-recursive and the response due to impulse input settles down to zero in a finite amount of time. FIR [2] filters are preferred over IIR filters in a wide array of engineering applications because of the linear phase and greater stability offered by the former. Again, depending on the nature of the filtering operation, filters can be classified as— analog and digital. Digital filter [4-8] circuits have to sample the analog input signal, convert the sampled signal into a set of binary numbers, store the numbers in memory, and process them through a processing unit before digitally manipulating them to yield the final output. None of these steps are required in an analog filter. Despite the higher complexities of digital filters, they are more widely preferred over analog filters these days because of the numerous advantages it offers like linear phase response, adaptability, repeatability, data storage ability, etc. Section 1.2 of this chapter presents a literature survey in the field of optimization techniques. Section 1.3 reveals the motivation behind taking up this work and finally the Thesis outline is presented in Section 1.4.

1.2 Literature Survey

In recent days, the application of various optimization algorithms has found importance in solving problems in various fields. Traditional optimization methods require cost and time, so all optimization problems do not solve all problems, they are applicable only to a specific field. The main objective of optimization algorithms is to minimize or reduce the error between the ideal response and approximated response as much as possible. These algorithms

have the least probability of getting trapped in local minima, known as the “local optima entrapment” problem. Algorithms like particle swarm optimization algorithm, genetic algorithm, artificial bee colony algorithm, sine cosine algorithm, cuckoo search algorithm, etc. are significant innovations in optimization algorithms. Many theories and studies have published that no researchers use a single algorithm because according to **the No Free Lunch Theorem,** there is no such algorithm that can solve all optimization problems. This theorem implies that an algorithm can successfully solve a specific set of problems.

Booth’s algorithm on the other hand is known for optimizing a specific operation which is the multiplication of two binary numbers in a digital FIR filter. Basically, a Booth’s multiplier is performed using signed bit representation. Multiplication is crucial when comes to several Digital Signal Processing (DSP) applications involving convolution, Fast Fourier Transform (FFT), and in the Arithmetic and Logic Unit (ALU) of microprocessors. Several Very Large-Scale Integration (VLSI) design criteria like the area, power dissipation, speed, and cost depend on the performance of the multipliers that execute the multiplication operation. That is where Booth’s multiplier comes as it performs well when used for making digital filters Area-Efficient. This factor becomes useful for better pipelining and parallel processing of FIR filters.

1.3 Motivation

The authors in [9] designed both basic and Area-Efficient FIR filters on XC3S500E which is an FPGA (Field Programmable Gate Array) board that presumably later became outdated and the same design is simulated on IMAGE system, a software designed by Powailabs Pvt. Ltd, India. Since a lot of FPGA boards came after that are having improved features including the one being performed and will be used in this work, this motivated us to implement the same design on our FPGA boards. On the other hand, Booth’s algorithm enhances multiplication faster than the traditional multipliers. However, this algorithm has been heavily used to design FIR filters to optimize the multiplication operation in the FIR filters. So, a filter design is proposed in which the algorithm plays a central role as a multiplier. After designing, the proposed filter is compared with algorithms like Parks-McClellan (PM) [17] and the Improved Sine-Cosine Algorithm (ISCA) [18].

1.4 Thesis Outline

The Thesis work has been systematically arranged and efforts have been taken to cover the theoretical aspects in detail about the practical work performed in the laboratory.

Chapter 1 presents the introduction of the Thesis work as well as the advantages and disadvantages of the FIR filter. A literature survey and the motivation behind this work have been also added in this chapter. The chapter ends with the Thesis outline.

Chapter 2 provides a description of Booth's algorithm followed by a description of FPGA hardware that has been used to design the filters described in chapters 3 and 4.

Chapter 3 presents the types of FIR filters followed by the proposed design of the Type-I filter along with simulation results.

Chapter 4 presents the parallel processing of the FIR filter and is followed by the comparison of the design of a basic as well as Area-Efficient 2-Parallel FIR filters with the help of simulation results.

Chapter 5 gives the conclusion of the complete Thesis work. The future scope of the filter design is also discussed in this chapter.

Chapter 2: Design Methodologies and Hardware Tools

2.1 Introduction

Studies have found that out of all multipliers, Booth's Multiplier is highly preferred due to the reduction of partial products [10-15]. It further increases speed and reduces power consumption [10]. The important feature of Booth's algorithm is the signed bit representation which is the multiplication that can be done for both positive and negative integers; both of which are performed in 2's complement in binary. Booth's algorithm can be done in both synchronous and asynchronous ways [11]. Hence, becoming the central component for designing the FIR filters in the next two chapters, the simulation part has been done in XC6SLX16-2FTG256 FPGA hardware. FPGA can be programmed according to our desire and also provide greater flexibility. XC6SLX16-2FTG256 belongs to the Spartan-6 FPGA family, which is available in various speed grades. The AC and DC components for this FPGA are specified for commercial, industrial, and expanded temperature ranges.

The chapter begins with the different types of binary multipliers starting from 2.2; Booth's algorithm is further elaborated in Section 2.3, followed by a Pseudo Code in Section 2.4. Section 2.5 discusses the features of FPGA in which the filters will be implemented as described in chapters 3 and 4. Section 2.6 describes the design of FIR filters using the FDA tool of MATLAB. The chapter is finally summarized in Section 2.7.

2.2 Binary Multiplier

A binary multiplier is an electronic circuit that is built using binary adders. The multiplication is executed using the traditional method which is a sequence of shifting followed by accumulating and then adding the partial products. Figure 2.1 and Fig. 2.2 describe the multiplication of two numbers, 12 and 11 taken as an example, in both decimal and binary format respectively.

```
   12          // Multiplicand
x  11          // Multiplier
   12          //12 x 1 which is the ones digit of 11
  +12          //12 x 1, shifted 1 position to left
  132          // final result obtained after adding the partial products.
```

Fig. 2.1: Multiplication of two numbers in decimal format.

```
    1100       // this is 12 in binary format
x   1011       // this is 11 in binary format
    1100       // this is 1100 x 1, the is the ones digit of 1011
   +1100       // this is 1100 x 1, shifted 1 position to left
  +0000        // this is 1100 x 0, shifted 2 positions to left
  +1100        // this is 1100 x 1, shifted 3 positions to left
10000100       // this is 132 in binary system obtained after
               adding all the four partial products.
```

Fig. 2.2: Multiplication of two numbers in binary format.

From above, the multiplication process involves adding the partial products that might further involve the addition of carry to the next set of partial products. The multiplication in a binary system might be easier than decimal multiplication, but the former will take computations much longer than the latter. Other difficulties with the traditional multiplication style are the digital processing units including the sign of the input numbers within the number itself using the 2's complement technique. These factors complicate the process and often require adjustments to the processor to accept and handle such inputs.

To generalize, a binary multiplier is an electronic circuit built using binary adders. The multiplication operation is executed using a sequence of shifting, accumulating, and adding the partial products. For an n-bit multiplier and m-bit multiplicand, the resultant product is n + m bits. The generation of n partial products requires n*m two input AND gates whereas the product is a result of n + m bits, and may require at least n adders.

2.2.1 Multiplier Algorithms

To overcome the problem of traditional multiplication, various multiplier algorithms have been invented and are popularly used in various signal and image processing applications. Some of them are mentioned below:

a) **Sequential Multiplier:**

This multiplier consists of a sequential circuit using a single n-bit adder to compute the product of two binary numbers, X and Y of n-bit and m-bit length respectively. This sequential circuit processes the partial products one at a time and repeats the process m times. In each step, partial products generated will be added to an accumulated partial sum and the resulting sum will be shifted to align the accumulated sum with a partial product of the next step. Each step of a sequential multiplication consists of three operations, i.e. generating partial products, adding the generated partial products to the accumulated partial sum, and shifting the partial sum.

b) **Combinational Multiplier:**

These are used to perform the multiplication of two unsigned or signed binary numbers. For two n-bit inputs X and Y, the 2n-bit product in terms of a combinational function P = X.Y can be expressed. Such multipliers use the technique of partial product accumulation. Each bit of the multiplier is multiplied against the multiplicand, the product is associated according to the position of the bit within the multiplier, and the resulting products are then added to form the result. If the multiplier bit is 1, the product is a shifted copy of the multiplicand whereas if the multiplier bit is 0, the partial product is 0.

c) **Array Multiplier:**

This algorithm is much similar to the traditional multiplication process based on the add-and-shift technique followed in any number system. It builds an array of full and half adders for the computation of the product. The process involves multiplying bit-by-bit of the multiplier with the entire input multiplicand. Such individual multiplications result in multiple partial products obtained by sequential shifting and eventually adding all the partial products to obtain the result.

d) Booth's Multiplier:

This algorithm is a very powerful and efficient algorithm to compute the multiplication of two signed binary numbers in 2's complement notation. This algorithm examines adjacent pairs of bits in the N-bit multiplier Q, in signed 2's complement representation, including an implicit bit below the least significant bit, $N-1 = 0$.

Since Booth's algorithm for multiplier will be the central part of this Thesis work, the next section will be discussed about the algorithm elaborately.

2.3 Radix-2 Booth's Algorithm

Booth's algorithm got its name from the inventor Andrew Donald Booth in 1950. It is an algorithm that multiplies two binary numbers in two's complement format. Any multiplier works as the basic building block of an FIR filter. Out of all multipliers, parallel multipliers are preferred for their high speed and Booth's Multiplier is a type of parallel multiplier. Furthermore, the speed of Booth's multiplier can be increased by reducing the terms of multiplicand and multiplier. Section 2.4 shows the pseudo-code of Booth's algorithm.

2.4 Pseudo Code of Radix-2 Booth's Algorithm

Initialize M to the multiplicand and A to 0
Initialize Q to the multiplier
Set n to the number of bits in the multiplier
Set Q_{-1} to 0
Repeat n times:
    if Q[n]= Q_{-1} then
        shift Q_{-1} right by one bit and set the leftmost bit to the original value of Q_{-1}
    else
        if Q[n]=0 and Q_{-1}=1, then
            A← A+M
            shift Q_{-1} right by one bit and set the leftmost bit to the original value of Q_{-1}
        else if Q[n]=1 and Q_{-1}=0, then
            A← A-M
            shift Q_{-1} right by one bit and set the leftmost bit to the original value of Q_{-1}
        end if
    end if
end Repeat
The final result is the concatenation of A and Q

The algorithm starts with initializing A to 0, along with assigning M and Q to the multiplicand and multiplier respectively. Also, n is set to the number of bits (Note that n should be of radix-2, which means n should not have any number other than 2 as a factor) in the multiplier, and Q_{-1} is set to 0. The iteration begins by checking the rightmost bit of Q i.e., Q[n] and Q_{-1}. If both Q[n] and Q_{-1} are equal, i.e., if the bits are 00 and 11, then the Arithmetic Right Shift (ASR) of AQQ_{-1} begins. In this process, Q_{-1} is shifted right by one bit and the leftmost bit of AQQ_{-1} is set to be the original value of Q_{-1}. If Q[n] and Q_{-1} are 01, then A and M are added and stored to M, and then the ASR of AQQ_{-1} begins in the same way. If Q[n] and Q_{-1} are 10, then A and M are subtracted by 2's complement and stored to M and then ASR of AQQ_{-1} begins in the same way. The iteration continues with n no. of times and after the iteration, the results of A and Q are concatenated to get the final result. Figure 2.3 shows the flowchart of Booth's algorithm.

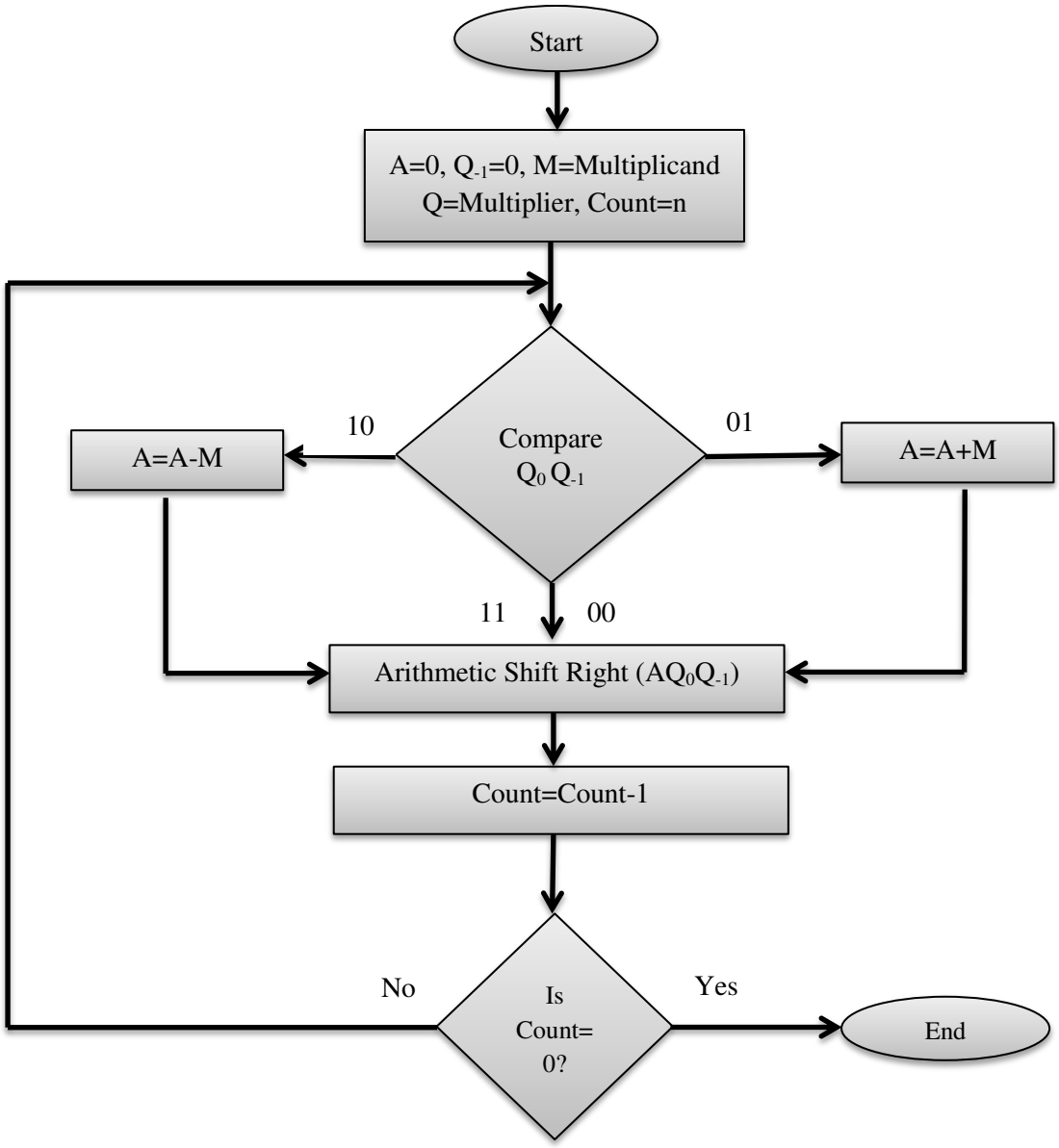

Fig. 2.3: Flowchart of Booth's algorithm.

To understand Booth's algorithm more clearly, an example is shown in Table 1. Two numbers, -7 and 3 are multiplied. For Booth's multiplication, both the numbers are converted into binary. The binary numbers of -7 and 3 are 1001 and 0011 respectively. Since -7 is a negative number, the MSB of A has been retained in each step followed by Arithmetic Shift Right of AQQ_{-1}. The final result obtained is AQ in step 4. As the result is negative, 2's complement of AQ is performed to find the actual answer.

Table 2.1: Example of Booth's algorithm using signed numbers.

<table>
<tr><td rowspan="5">A=0000

Q=3=0011

Q_{-1} = 0

N=4

M= -7= -(0111)=1001</td><td>Step</td><td>A</td><td>Q</td><td>Q_{-1}</td><td>Operation</td><td>N</td><td rowspan="5">Result = AQ = 1110 1011

Since the result is a negative number, we'll have to find its 2's complement

2's Complement of
1110 1011
↓ 1's Complement
0001 0100
↓ Adding 1
0001 0101

Which is the binary of 21

∴ (-7) x 3 = -21</td></tr>
<tr><td>1</td><td>0000
0111
0011</td><td>0011
0011
1001</td><td>0
0
1</td><td>A=A-M
ASR</td><td>N=4
↓
N=3</td></tr>
<tr><td>2</td><td>0011
0001</td><td>1001
1100</td><td>1
1</td><td>ASR</td><td>N=3
↓
N=2</td></tr>
<tr><td rowspan="2">NOTE:

ASR=Arithmetic Shift Right</td><td>3</td><td>0001
1010
1101</td><td>1100
1100
0110</td><td>1
1
0</td><td>A=A+M
ASR</td><td>N=2
↓
N=1</td></tr>
<tr><td>4</td><td>1101
1110</td><td>0110
1011</td><td>0
0</td><td>ASR</td><td>N=1
↓
N=0</td></tr>
</table>

2.5 Field Programmable Gate Array (FPGA)

2.5.1 Introduction

FPGA is an abbreviation for Field Programmable Gate Array. FPGA [28-41] is an array of interconnected digital sub-circuits that implement common logic functions while having high levels of flexibility [30-36] and they can be programmed as required by the user anytime which is an advantage. In earlier days, microprocessors were used to serve this purpose but since they were costly as well as programs in a microcontroller are executed sequentially, the process is sequentially constrained and hence, it is impossible to achieve such multiple tasks simultaneously. To overcome this problem, FPGA [42] has been introduced.

2.5.2 Hardware Description Language (HDL)

FPGA programming is implemented using Hardware Description Language (HDL). The two most commonly known HDLs are VHDL (VHSIC Hardware Description Language) and Verilog. HDLs [36-38] are entirely different from mainstream programming languages like C, and Python as a program written in an HDL is a schematic of a circuit diagram. Moreover, HDL programs are written in three modes: behavioural, structural, and dataflow descriptions [33].

i) Behavioural description describes the behaviour of the design in terms of circuit or system using the algorithm.

ii) Structural description describes the structure of the circuit using logic gates and interconnects between the logic gates.

iii) Dataflow description describes the transfer of data from input to output.

2.5.3 FPGA Architecture

The generalized FPGA Structure has three types of Modules: i) Configurable Logic Blocks (CLB), ii) I/O Blocks, and iii) switch matrix or interconnection wires (Fig. 2.4) [39-41]. CLBs include input, output, and some digital logic, whereas I/O pads communicate the external ports via different applications. To implement the user logic between them, the interconnections provide direction.

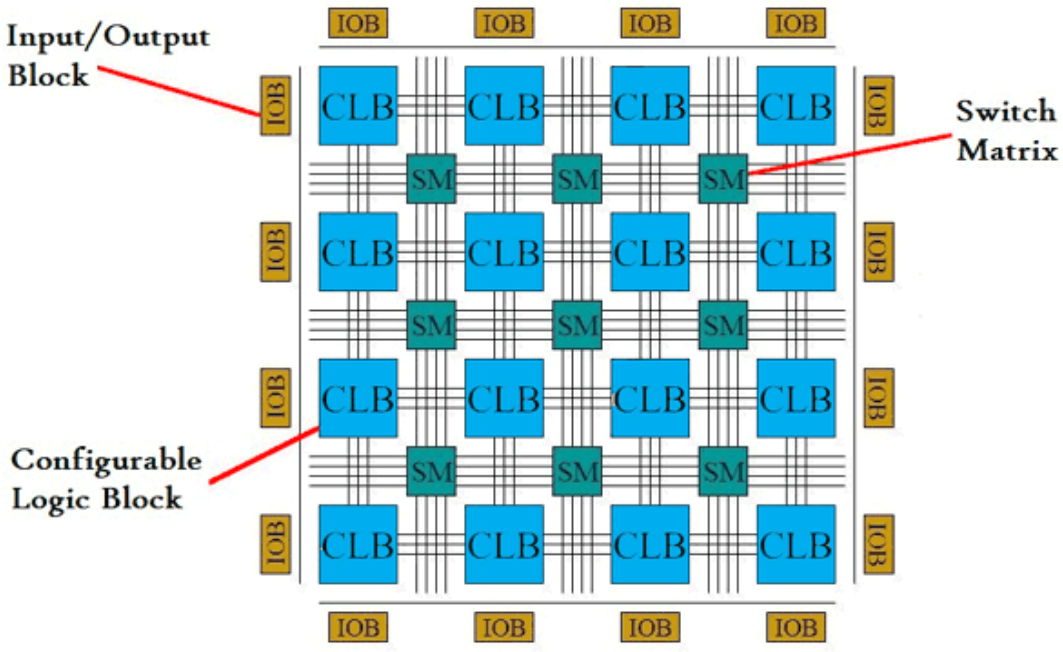

Fig. 2.4: FPGA Architecture.

2.6 FIR Filter Design using MATLAB FDA Tool

MATLAB is an abbreviation for Matrix Laboratory. It is the software well-known for applications in almost all engineering domains. The Filter Design and Analysis tool (FDA tool) is used for designing FIR filters. The design of the filter using the FDA tool has been used to observe the parallel processing of FIR filters, the theory which has been discussed in Chapter 4. FDA tool helps the user to design of digital filter by entering the desired filter specifications. Further, the FDA tool also provides other features like magnitude and phase plots and pole-zero plots. Figure 2.5 shows the design of the MATLAB design of FIR filter using the FDA tool. Given below are the steps for designing FIR filters using the FDA tool:

a) Open MATLAB and type "fdatool" on command window to open The FDA Tool dialog box.
b) Specify the response type as "Bandpass" and design method as "Equiripple" under FIR dropdown menu.
c) Density factor enables the user to control the density of the frequency grid. Although this value is kept 20 by default, increasing this value can design a filter that approximates an ideal equiripple filter. However, more time will be required for increase in the number of computations.
d) Specify the filter order as per the desire of the user by clicking the specify order radio button.
e) Keep the frequency specifications as normalized (0 to 1) under units dropdown menu with the following specifications: Wstop1=0.2, Wpass1=0.3, Wpass2=0.7 and Wstop2=0.8.
f) In the magnitude specifications, Wstop1, Wpass and Wstop2, all are kept as 1.
g) Click on Design filter and the filter gets designed.
h) After designing the filter, click on Targets section and then click on generate HDL with VHDL as the language.
i) In generate HDL section, specify the location of the generated HDL as per the user's convenience and after that under filter architecture section select the following parameters: i) Architecture → Fully parallel, ii) Coefficient Multipliers → Multiplier iii) Mark the "Optimize for HDL" as checked and iv) FIR adder style → Tree.

j) After that on Global Settings section, go to ports and Clear the checkbox buttons of both “Add input register” and “Add output register”.

k) Click on Generate HDL and the file generated is ready for synthesis.

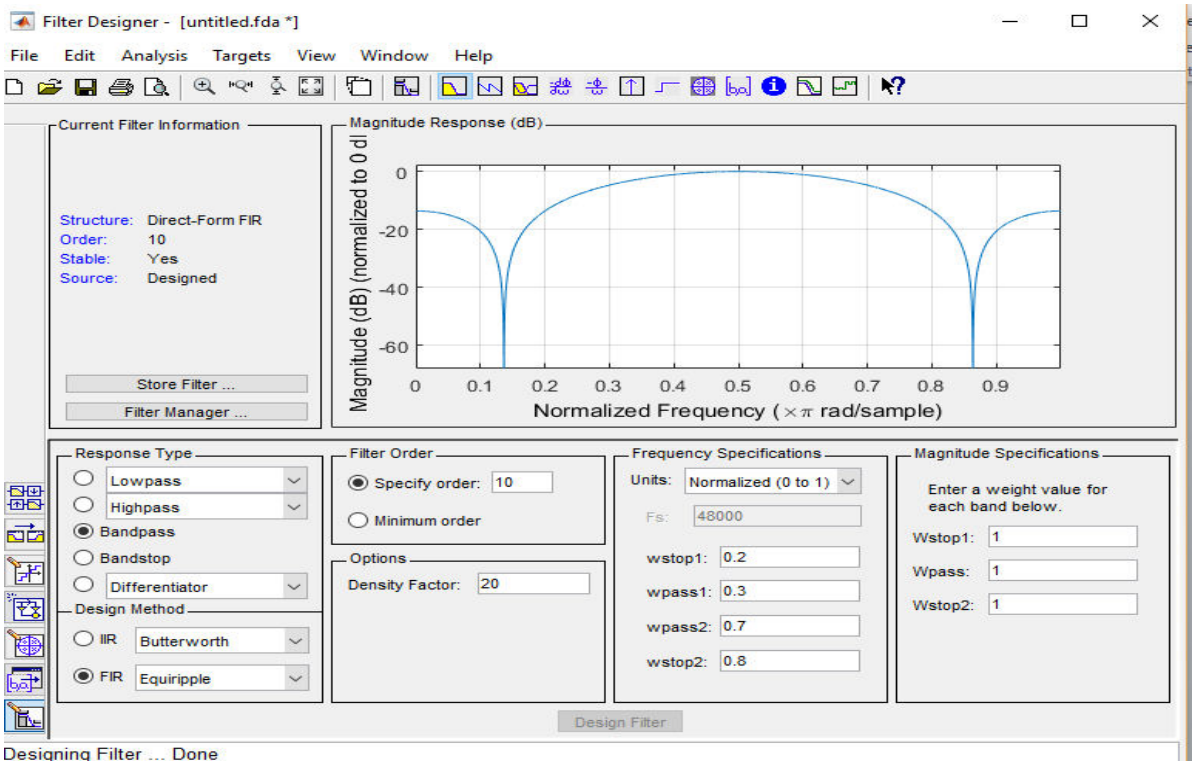

Fig. 2.5: Design of a Band Pass Filter using FDA tool window.

2.7 Summary

In this chapter, the working principle of Booth’s algorithm has been discussed with the help of a flowchart and a pseudo code. Further, Booth’s algorithm has been compared with other traditional multiplier based methods which justify its betterment. This chapter also described the working principle of FPGA followed by the working mechanism of the MATLAB FDA tool. These methods are used for the design of the proposed filters in subsequent chapters.

Chapter 3: Design of Type-I Band Pass FIR Filter using Booth's Multiplier

3.1 Introduction

The Finite Impulse Response (FIR) [2] filters are one of the two sub-categories of digital filters, the other one being Infinite Impulse Response (IIR) filters. The impulse response of FIR filters becomes zero in a finite duration of time. The structure of the FIR filter can be used to implement almost any kind of filter response digitally. Usually, these filters consist of a series of delays, adders, and multipliers. Low pass, Band pass [3], and High pass allow only the components within a certain frequency range and attenuate the rest which is outside the frequency range. Moreover, switching between the stop band and pass band is instantaneous i.e. transition width is zero.

The term optimization refers to the process of selecting the best option from a set of other alternative options. An optimization problem generally consists of inputs that can store as variables. One or more objective function is evaluated for each input to find the best results out of all inputs, thus minimizing the output of the objective function known as the fitness or cost of the solution. Booth's algorithm, on the other hand, is also an optimization algorithm but it is designed to optimize a particular operation, which is the multiplication of two-bit signed binary numbers. Booth's multiplier uses a technique that reduces the number of partial products required to compute the final product.

Section 3.2 shows different types of FIR filters whereas Section 3.3 classifies the filter according to frequencies. Section 3.4 formulates the problem of the Band Pass FIR filter and Section 3.5 describes the proposed FIR filter design. The simulation results of filters are in Section 3.6 and the summary of this chapter is in Section 3.7.

3.2 Types of Digital Filters

Filters are the basic components for all signal processing and telecommunication systems. A filter always rejects unwanted frequencies from the input signal and allows the frequencies that are desirable to the user. Digital filter mainly consists of two types:

a) FIR Filter

b) IIR Filter

a) FIR Filter:

FIR Filter is a digital filter whose impulse response is of finite duration as it settles down to zero after a finite duration of time. FIR filters are generally non-recursive filters. The present output sample depends on the present input samples.

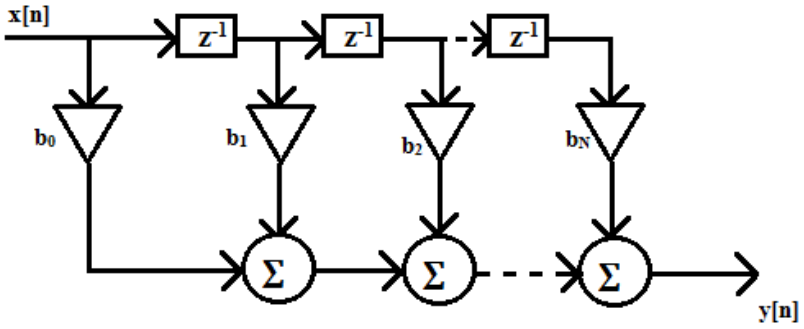

Fig. 3.1: Block diagram of an FIR Filter.

b) IIR Filter:

As the name suggests, the impulse response of an IIR filter is infinite [16], as it doesn't settle down to zero in finite time. The IIR filter is of recursive type as the present output signals depend on both present and past input signals as well as past output signals.

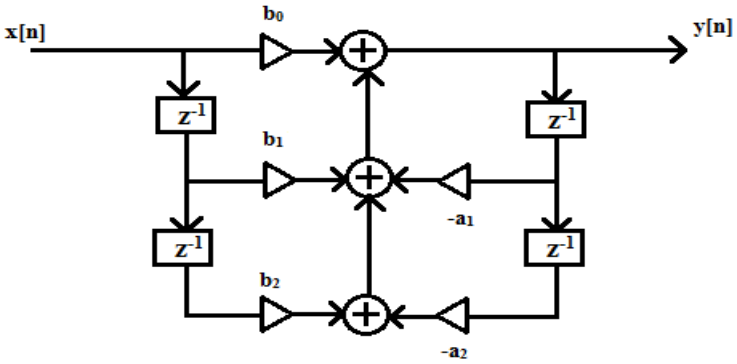

Fig. 3.2: Block diagram of an IIR Filter.

3.3 Frequency Selective Filters

The range of frequencies in which the signal can pass through is known as the pass band whereas the range of frequencies in which the signal cannot pass is called the stop band. Based on pass band and stop band frequencies, frequency selective filters are divided into four categories:

a) Low Pass Filter (LPF)
b) High Pass Filter (HPF)
c) Band Pass Filter (BPF)
d) Band Reject Filter (BRF)

a) Low Pass Filter (LPF): A Low Pass Filter (LPF) [42-45] is a filter that passes signals with a frequency lower than a selected cut-off frequency and attenuates signals with a frequency higher than the cut-off frequency.

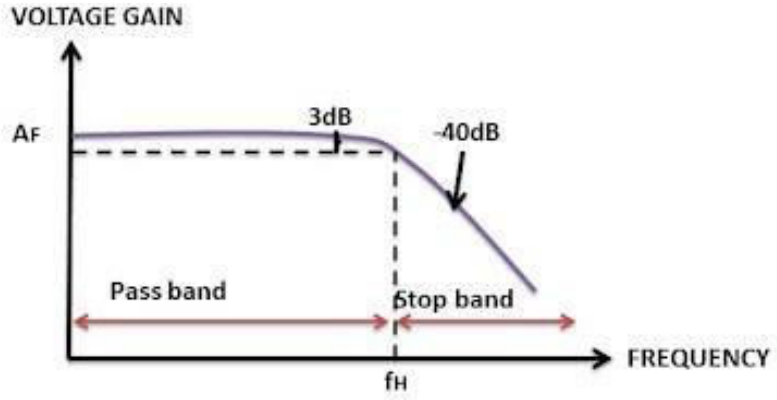

Fig. 3.3: Magnitude response of a LPF.

b) High Pass Filter (HPF): In an HPF [46-50], signals are passed with a frequency higher than the cut-off frequency and attenuates signals lower than the cut-off frequency.

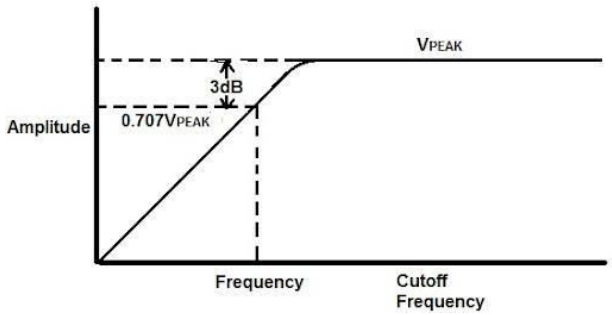

Fig. 3.4: Magnitude response of a HPF.

c) Band Pass Filter (BPF): BPF [51-56] passes frequencies within a certain range and rejects frequencies outside that range.

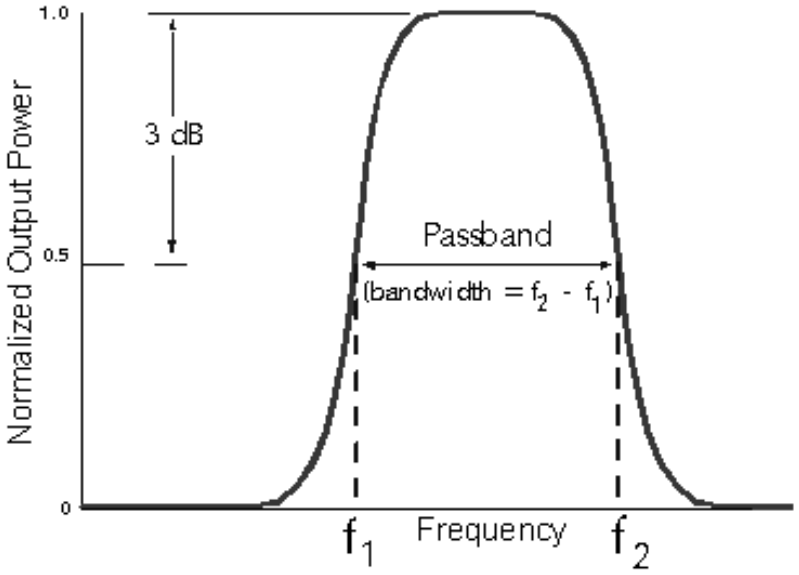

Fig. 3.5: Magnitude response of a BPF.

d) Band Reject Filter (BRF): A Band Stop or Band Reject filter passes most frequencies unaltered, but attenuates those in a specified range to a very low level.

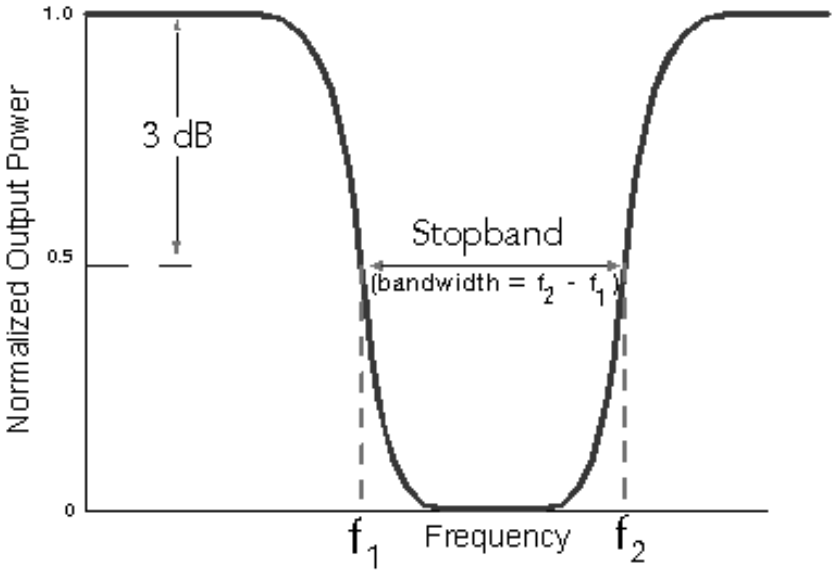

Fig. 3.6: Magnitude response of a BRF.

3.4 Problem Formulation for Band Pass FIR Filter

The transfer function of an FIR filter [2] is given in Eq. (3.1).

$$H(z) = \sum_{n-0}^{M-1} h(n)z^{-n} \quad n = 0,1, \ldots., L-1 \tag{3.1}$$

Equation (3.1) has (M-1) trivial poles and (M-1) trivial zeros located anywhere on the z-plane according to the nature of roots. h(n) denotes the impulse response of the FIR filter. Based on the coefficients of h(n), different types of magnitude responses will be formed by the filter. These filters consist of not only band pass filters but high pass and low pass filters as well.

3.4.1 Error Function Representation

The difference between the pass band and the stop band is known as the error function. The error function can be done practically by minimizing the deviation of the actual response with the ideal response that is defined by Eq. (3.2).

$$E(\omega) = W(\omega)\,[H_d(e^{j\omega}) - H_a\,(e^{j\omega})] \tag{3.2}$$

The error function in Eq. (3.2) is given by Parks-McClellan where $W(\omega)$, $H_d(e^{j\omega})$ and $H_a(e^{j\omega})$ are the weight vector, desired and approximated frequency responses respectively. The minimization of error is modulated by $W(\omega)$. Further in Eq. (3.2), the ratio between peak ripples at pass band (δ_p) and stop band (δ_s) cannot have different values. To reduce flaws in the function, the error function is modified which is represented as:

$$U = \underset{\omega \le \omega_p}{max}\,(|E(\omega)| - \delta_p) + \underset{\omega \ge \omega_s}{max}\,(|E(\omega)| - \delta_s) \tag{3.3}$$

ω_p, ω_s, δ_p and δ_s are the filter specifications in Eq. (3.3).

Equation (3.4) represents the ideal response of a Band Pass filter:

$$H_d\,(e^{j\omega}) = \begin{cases} 0, & 0 \le \omega \le \omega_{s1} \\ 1, & \omega_{p1} \le \omega \le \omega_{p2} \\ 0, & \omega \ge \omega_{s2} \end{cases} \tag{3.4}$$

ω_{s1} and ω_{s2} denote the first and second stop band frequencies respectively whereas ω_{p1} and ω_{p2} denote the first and second pass band frequencies respectively.

3.4.2 Type-I Linear Phase FIR Filter

The frequency response of FIR filter is denoted by,

$$H(e^{j\omega}) = \sum_{n=0}^{M-1} h(n)e^{-j\omega n}, \quad -\pi \leq \omega \leq \pi \tag{3.5}$$

whereas Eq. (3.6) represents the linear phase constraint,

$$\angle H(\omega) = -\tau_{\Phi}\omega, \qquad -\pi \leq \omega \leq \pi \tag{3.6}$$

where τ_{Φ} is a constant phase delay. For Type-I filter, *h(n)* must be symmetrical:

$$h(n) = h(L-1-n), 0 \leq n \leq L-1 \text{ with } \tau_{\Phi} = \frac{(L-1)}{2} \tag{3.7}$$

In Eq. (3.7), *h (n)* shows the symmetry with respect to τ_{Φ} and τ_{Φ} is the index of symmetry. The value of *M* in Eq. (3.5) takes even or odd integer values in case of Type 1 and Type 2 filters. The frequency response of Type 1 filter is:

$$H\left(e^{j\omega}\right) = \left[\sum_{n=0}^{M-1} a(n)\cos\omega n\right]e^{-j\omega(M-1)/2} \tag{3.8}$$

3.4.3 Design of a FIR Filter using Parks McClellan Algorithm

To compare the performance of Booth's algorithm in higher orders, the FIR filter is designed first using Parks-McCllelan (PM) algorithm to find the filter coefficients. PM algorithm is one of the most popular and traditional algorithms for designing FIR filters. To design such filters, 'firpm' and 'fvtool' commands of MATLAB 2017a software are used. Figure 3.7 shows the magnitude response of the BPF for both taps that are designed using the PM algorithm. After designing, the filter coefficients obtained will be used as the inputs for the FIR filter that will be designed using Booth's algorithm in the next section.

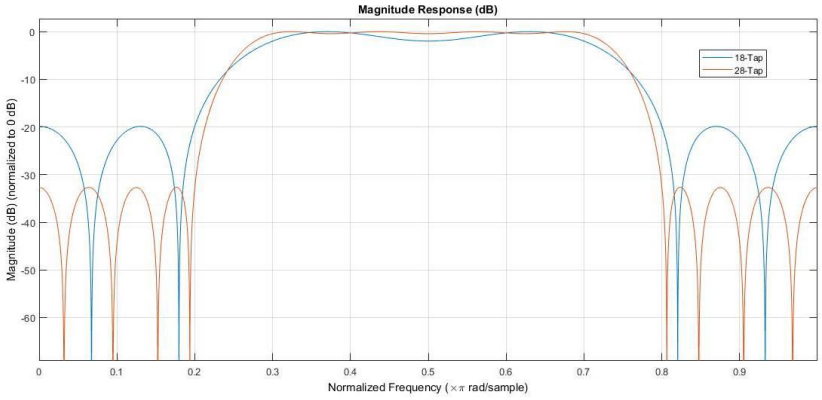

Fig. 3.7: Magnitude response of Filters using PM algorithm.

3.5 Proposed FIR Filter Design

Using Booth's multiplier as the basic building block, two FIR filters of orders 18 and 28 are designed using the filter coefficients obtained in the previous section. Figure 3.8 shows the proposed design of an 18-tap filter. In this design, we take two coefficients of 2 bits each in the ports at a time and those coefficients are given as inputs in each multiplier. The results of those inputs are further fed into another set of Booth's multipliers of 4-bit inputs each, the outputs of which are further fed into a set of 8-bit multipliers followed by 16 and 32-bit input multipliers, leading to the output as 64-bit. The 28-order is also made similarly.

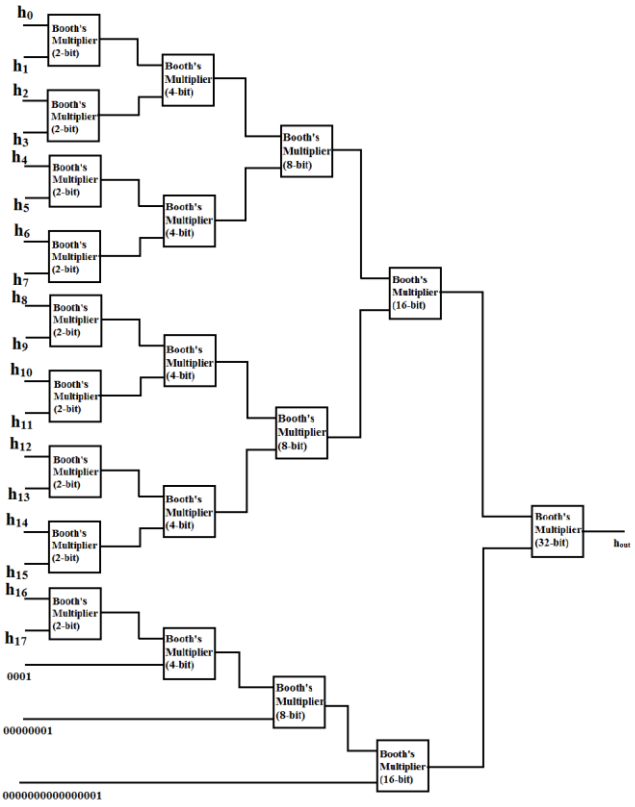

Fig. 3.8: Block diagram of 18-tap FIR Filter.

3.6 Simulation Results

The simulation of the designs has been carried out using Xilinx ISE 14.7 as the simulation software as well as XILINX XC6SLX16-2FTG256C being the FPGA hardware. The implementation of the filter design has been done with the following parameters: $\omega_{s1}=0.2\pi$, $\omega_{s2}=0.3\pi$, $\omega_{p1}=0.7\pi$, and $\omega_{p2}=0.8\pi$. The design work is completely done on the above-mentioned tools along with the 'fdatool' feature of MATLAB 2017a. The simulation results have been further compared with the filters previously implemented using PM [17] and ISCA [18]. Table 3.1 and Table 3.2 show the device and power utilization of the filters respectively.

From Table 3.1, Booth's algorithm consumes most components in all of the three algorithms compared as the number of Bonded IOBs in Booth's algorithm is twice the number of the other two algorithms when compared separately. If this design needs to migrate to a different FPGA device, more IOBs make the process easier. More IOBs in an FPGA not only increase the I/O capability but also the timing and signal integrity of the design are improved. More IOBs can facilitate debugging as well as testing of the filter design. Greater flexibility has been ensured and the performance of the filter can be improved by distributing the logic function across more pins. With more IOBs, more routing resources are available that can help reduce routing congestion and improve the overall performance of the filter. More pins can be distributed on the logic across the FPGA which reduces the length of interconnects, which can help reduce delay and improve timing.

Table 3.1: Device Utilization of the Filters

Component	Available	N=18			N=28		
		PM [17]	ISCA [18]	Proposed	PM [17]	ISCA [18]	Proposed
Slice Registers	18224	195	195	214	307	307	359
Slice LUTs	9112	96	96	1951	192	192	2145
Logic	9112	48	48	738	96	96	1816
Memory	2176	48	48	0	96	96	0
LUT-FF pairs	196	95	95	194	191	191	184
Bonded IOBs	186	51	51	101	51	51	121
BUFG/ BUFGCTRLs	16	1	1	1	1	1	1
DSP48A1	32	4	4	0	8	8	0

Table 3.2: Power Utilization of the Filters

Type of Power	N=18			N=28		
	PM [17]	ISCA [18]	Proposed	PM [17]	ISCA [18]	Proposed
Supply Power (mW)	20.39	20.39	20.44	20.4	20.4	20.48
Dynamic Power (mW)	0.48	0.48	0.52	0.48	0.48	0.57
Static Power (mW)	19.91	19.92	19.92	19.92	19.92	19.92

Also in Table 3.1, memory and DSP48A1 have zero components each as Booth's algorithm is implemented in FPGA. Not having any of these components not only makes the cost of the design less but will also result in low power consumption and better utilization of the FPGA's internal resources. Since filter taps are implemented using adders and registers, the design can be easily modified to meet changing requirements. The design can be easily ported to different FPGA or hardware platforms since there are no dependencies on memory or DSP blocks. Implementing an FIR filter without using dedicated memory or DSP blocks can be

used as a learning aid for students or engineers who want a deeper understanding of DSP and FPGA design. However, in Table 3.2, we can see that Booth's algorithm consumes slightly more power than the others. Figure 3.9 and Fig. 3.10 show the Register-Transfer Level (RTL) of the 18-tap and 28-tap FIR filter respectively whereas Fig. 3.11 shows the timing diagram of the 28-tap filter.

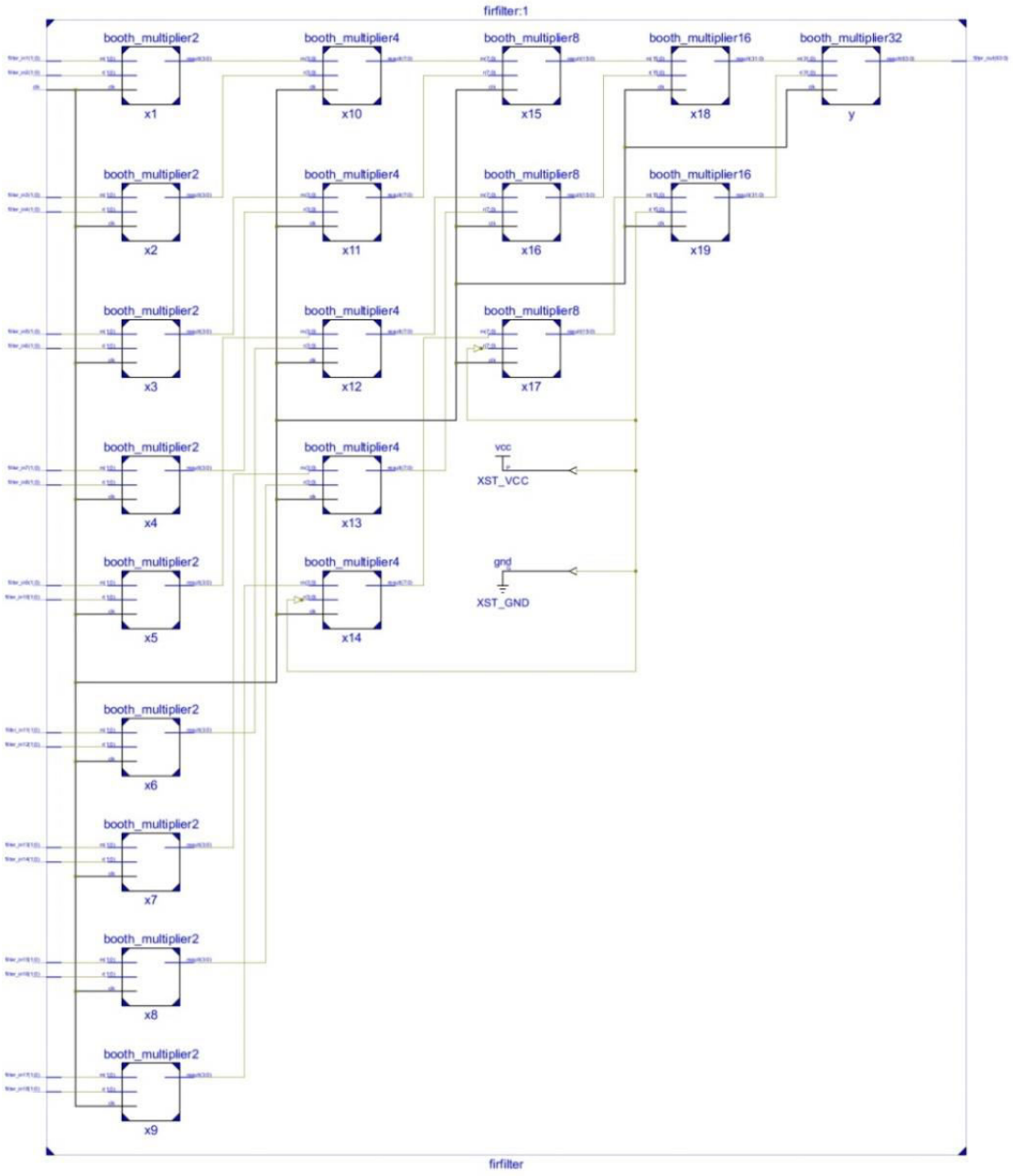

Fig. 3.9: RTL Schematic of 18-tap FIR Filter.

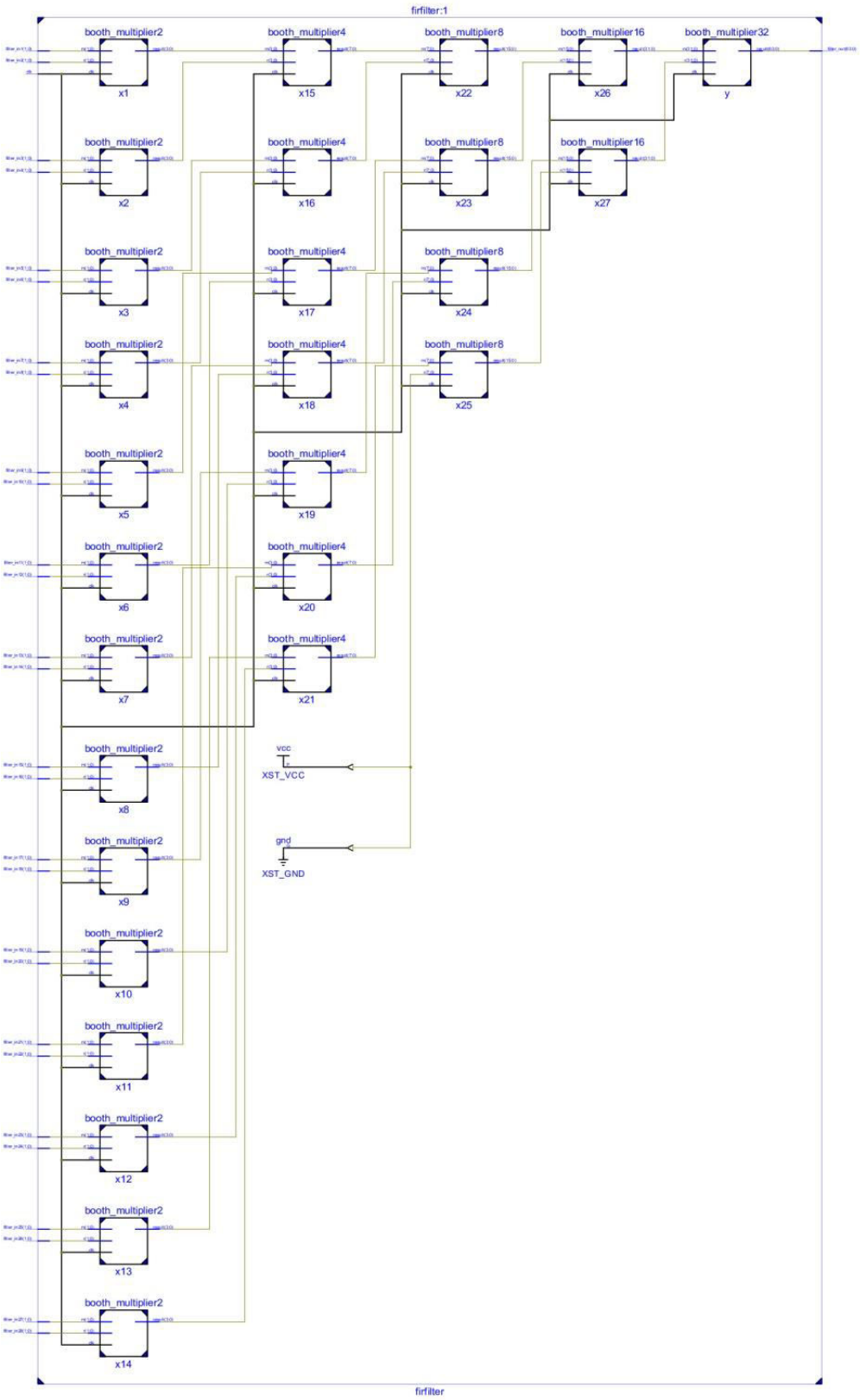

Fig. 3.10: RTL Schematic of 28-tap FIR Filter.

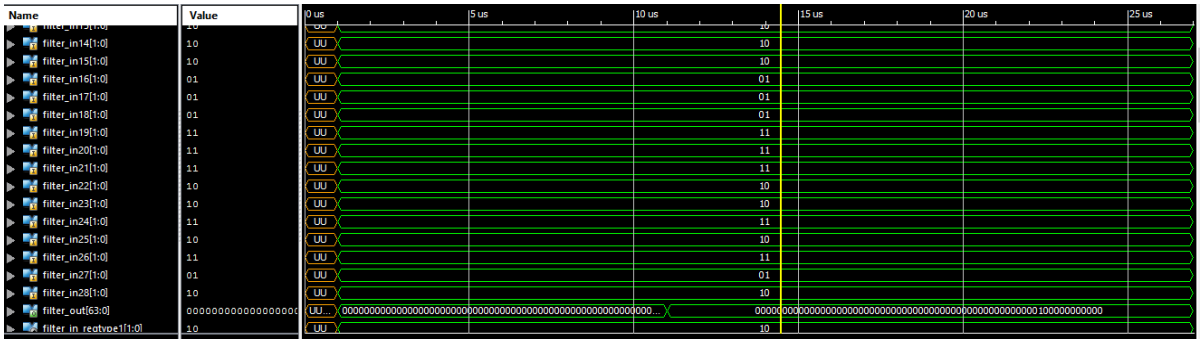

Fig. 3.11: Timing diagram of 28-tap FIR Filter.

3.7 Summary

This chapter described the design of a Type-I Band Pass filter using Radix-2 Booth's algorithm. The simulation results justify that for higher-order FIR filters, Booth's algorithm is preferred when no memory and no DSP blocks of FPGA are utilized to design. Booth's algorithm also performs well when more IOBs are included in an FPGA. For greater flexibility and low power consumption in an FIR filter, this algorithm becomes useful. However, in Table 3.2, we see that Booth's algorithm consumes more dynamic power when compared to others. Hence, dynamic power is a parameter that should be taken care of while designing the filters.

Chapter 4: Parallel and Area – Efficient Design of FIR Filters using Booth's Multipliers

4.1 Introduction

Due to the explosive growth of multimedia applications, the demand for high-performance and low-power digital signal processing systems is increasing every day. FIR filters are one of the most widely used fundamental blocks in DSP applications. Some applications like error correction and detection, video processing, and data compression require the filter to be operated at high frequencies, whereas other applications namely Multiple-Input Multiple-Output (MIMO) systems used in wireless communication require a low-power circuit. The MIMO system requires high throughput FIR filter. The parallel filter structure is a well-known technique for FIR filters in terms of area performance analysis [19-24]. Parallel processing and pipelining are two techniques used in DSP, and both of them can be exploited to reduce power consumption. Pipelining shortens the critical path by alternating pipelining latches along the datapath, at the price of increasing the number of latches and the system latency, whereas parallel processing increases the sampling rate by replicating hardware so that multiple inputs are processed in parallel and multiple outputs are generated at the same time, at the expense of increased area. Both techniques reduce power consumption by lowering the supply voltage, where the sampling speed does not increase.

Section 4.2 to Section 4.4 discusses the formulation of Carry Look Ahead Array whereas Section 4.5 describes the design of basic, and Area-Efficient Parallel 2-FIR filters. Section 4.6 gives the simulation results of the Parallel FIR filters, and Section 4.7 compares the design of filters with another FPGA. Finally, Section 4.8 gives a summary of the chapter.

4.2 Carry Look Ahead Adder

In the field of digital electronics, Carry Look Ahead (CLA) adder is widely used among all available adders [25-27]. In a Central Processing Unit (CPU) of any computer system, there are two main component units: The arithmetic and Logic Unit (ALU) and Control Unit (CU). ALU needs an adder circuit mainly for basic operations like addition, multiplication, etc. Parameters like speed, power, and time are dealt with by a VLSI engineer. A CLA adder is one such type of adder that is used in digital logic. The advantage of CLA adder is it speeds

up the carry bits and reduces area. It provides better speed with the help of their segments, i.e., carry propagate and carry generate, both denoted by P_i and G_i respectively.

To understand CLA adder more in detail, the truth table of full adder is needed to be considered. Table 4.1 shows the truth table of a full adder with carry and Fig. 4.1 shows the overall circuit of a full adder. In this circuit, A, B, and C_{in} are the inputs and C_{out} is the carry output.

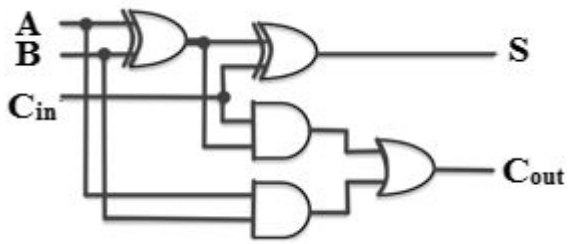

Fig. 4.1: Full Adder circuit.

Table 4.1: Truth table of Full Adder.

A	B	C_{in}	C_{out}
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

In Table 4.1, C_{out} can be calculated using K-Map but for the carry look-ahead adder, the truth table is divided into three parts: out of which, two parts will be needed since the discarded part has C_{out} 0 for both the inputs that are the first two rows. The rows labelled in green are inputs 110 and 111, where C_{out} for both is 1. Between these two inputs, A and B remain the same and C_{in} gets changed. So, AB will be one of the two terms for the expression of C_{out}. For the other term, the other part labelled in red will be needed. In this part, four inputs have been included out of which 010 and 101 have C_{out} as 0 but 011 and 101 have 1. By comparing all four rows, A and B are XOR'ed and the resultant output is AND'ed with C_{in} to get 1. So, the other expression is $(A \oplus B)C_{in}$. Thus, the overall expression of C_{out} is given in Eq. (4.1):

$$C_{out} = AB + (A \oplus B)C_{in} \tag{4.1}$$

Putting $AB = G_i$ and $(A \oplus B) = P_i$ in Eq. (4.1), we get

$$C_{out} = G_i + P_i C_{in} \tag{4.2}$$

Where G_i and P_i are the carry generate and carry propagate respectively. Equation (4.2) forms the base equation of CLA array which will be discussed later on Section 4.4.

4.3 Carry (or Borrow) Look Ahead Subtractor

Similar to the CLA adder, the CLA subtractor also forms a CLA array by forming Eq. (4.2). The major difference between the two is CLA subtractor uses a full subtractor to form C_{out}. Table 4.2 shows the truth table of a full subtractor with borrow and Fig. 4.2 shows the overall circuit of a full subtractor. In this circuit, A, B, and C_{in} are the inputs and C_{out} is the borrow (or carry) output.

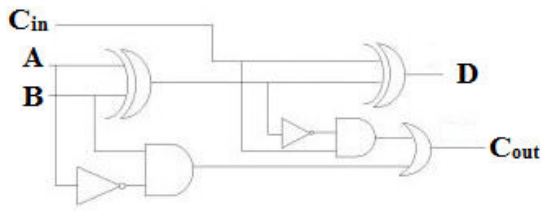

Fig. 4.2: Full Subtractor circuit.

Table 4.2: Truth table of Full Subtractor.

A	B	C_{in}	C_{out}
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

In Table 4.2, all four inputs whose C_{out} is 1 are picked to determine the expression of C_{out} for CLA subtractor; out of which two are labelled green and the other two are red. Inputs 010 and 011 have been labelled green. In these two inputs, A and B remain the same. So, the first expression of C_{out} is B. Now, inputs 001 and 111 are labelled red. On both of these inputs, C_{in} remains 1 and when A and B have the same inputs, the output becomes 1. By comparing both rows, A and B are set in an XNOR gate and the result is set to an AND gate with C_{in}. So, the second expression is $(A \odot B)C_{in}$. The overall expression for C_{out} of CLA subtractor is given in Eq. (4.3):

$$C_{out} = \bar{A}B + (A \odot B)C_{in} \tag{4.3}$$

Putting $\bar{A}B = G_i$ and $(A \odot B) = P_i$ in Eq. (4.3), we get

$$C_{out} = G_i + P_i C_{in} \tag{4.4}$$

G_i and P_i are carry generate and carry propagate terms respectively. Equation (4.4) is identical to Eq. (4.2). This shows that Eq. (4.3) forms the base equation of CLA array for subtractor after identifying the carry generate and carry propagate terms.

4.4 Carry Look Ahead (CLA) Array

The purpose of creating a CLA adder (or subtractor) is to speed up the time required to get the carry bits and reduce the area. For an n-bit CLA array, n stages are formed. As mentioned, Eq. (4.1) and Eq. (4.3) form the base of the CLA array for adder and subtractor respectively. Both equations form the first stage of a CLA array separately for adder and subtractor. To advance to the next stage, C_{in} and C_{out} of Eq. (4.2) are replaced by C_i and C_{i+1} respectively. After replacing, Eq. (4.2) becomes

$$C_{i+1} = G_i + P_i C_i \tag{4.5}$$

For example, in a 4-stage or CLA adder, the value of i is taken from 0 to 3. After the putting of values of i in Eq. (4.5), the CLA array goes like the set of equations described in (4.6). Figure 4.3 shows the diagram of the 16-bit CLA Array where the carry of each 4-bit CLA adder is forwarded to the next stage in the array.

$$C_0 = C_{in}$$

$$C_1 = G_0 + P_0 C_0 = G_0 + P_0 C_{in}$$

$$C_2 = G_1 + P_1 C_1 = G_1 + P_1 G_0 + P_1 P_0 C_{in}$$

$$C_3 = G_2 + P_2 C_2 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{in} \tag{4.6}$$

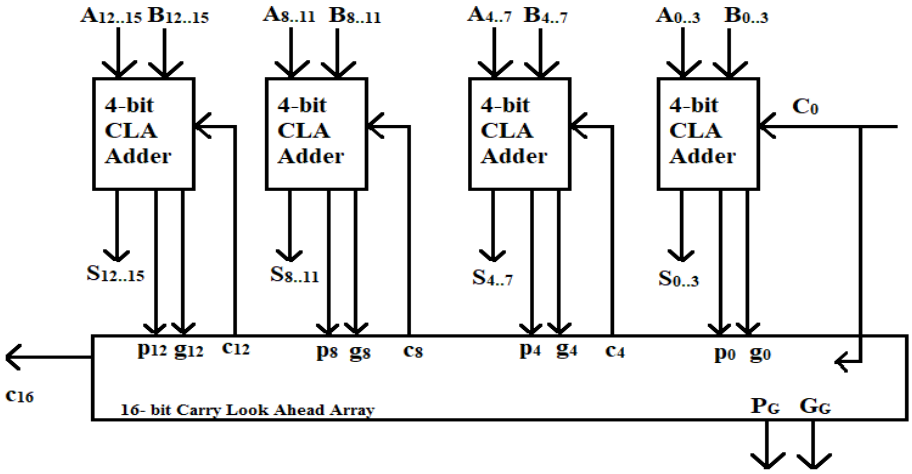

Fig. 4.3: 16-bit Carry Look Ahead Adder.

4.5 Design of a Parallel FIR Filter System

Consider a 3-tap FIR filter system [19] as in Eq. (4.7) i.e.,

$$y(n) = a\,x(n) + b\,x(n-1) + c\,x(n-2) \tag{4.7}$$

The following system in Eq. (4.7) is a SISO (Single Input Single Output) system. However, the delay in this equation is high, so as the power consumption. To achieve less delay, the system will be converted from SISO to MIMO (Multiple Input Multiple Output). Figure 4.4 shows the blocks from conversion of SISO to MIMO. This system conversion is known as parallel processing and the MIMO system is the parallel FIR filters. Eq. (4.8) - (4.10) show the three input parallel system.

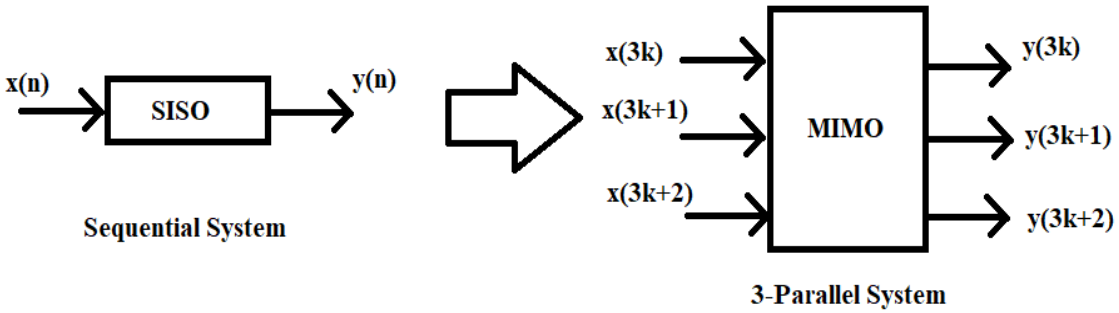

Fig. 4.4: Conversion from SISO to MIMO.

Parallel FIR filters are designed using two methods [20] predominantly: 2x2 parallel filter and 3x3 parallel filter. As the name suggests, 2x2 parallel filter has two filter inputs (x_0, x_1), two filter coefficients (h_0, h_1) and two filter outputs (y_0, y_1), whereas 3x3 filter has three filter inputs (x_0, x_1, x_2), three filter coefficients (h_0, h_1, h_2) and three filter outputs (y_0, y_1,y_2). The Eq. (4.8) - (4.10) represent a 3x3 parallel filter.

$$y(3k) = a\,x(3k) + b\,x(3k-1) + c\,x(3k-2) \tag{4.8}$$

$$y(3k+1) = a\,x(3k+1) + b\,x(3k) + c\,x(3k-1) \tag{4.9}$$

$$y(3k+2) = a\,x(3k+2) + b\,x(3k+1) + c\,x(3k) \tag{4.10}$$

In the example, k is the clock cycle. If all the 3 inputs x(3k), x(3k+1), x(3k+2) are delayed by 1 clock cycle, then the signals will lead to x(3k-3), x(3k-2) and x(3k-1) respectively.

4.5.1 Design of a Basic 2-Parallel FIR Filter

Now, consider a Basic 2-Parallel FIR filter which has two replications of a 4-tap FIR filter (Fig. 4.5). In this FIR filter, D-flip flops are used for designing the delay blocks. The dotted line indicates the critical path which is the longest path with no delay or minimum delay. For simulation, the "fdatool" feature of MATLAB is used that further generates VHDL file. The inputs are taken as 16-bits each whereas outputs as 32-bits each. The equations for the basic 2-Parallel FIR filter are given below:

$$y(2k) = h_0x(2k) + h_1x(2k-1) + h_2x(2k-2) + h_3x(2k-3) \quad (4.11)$$

$$y(2k+1) = h_0x(2k+1) + h_1x(2k) + h_2x(2k-1) + h_3x(2k-2) \quad (4.12)$$

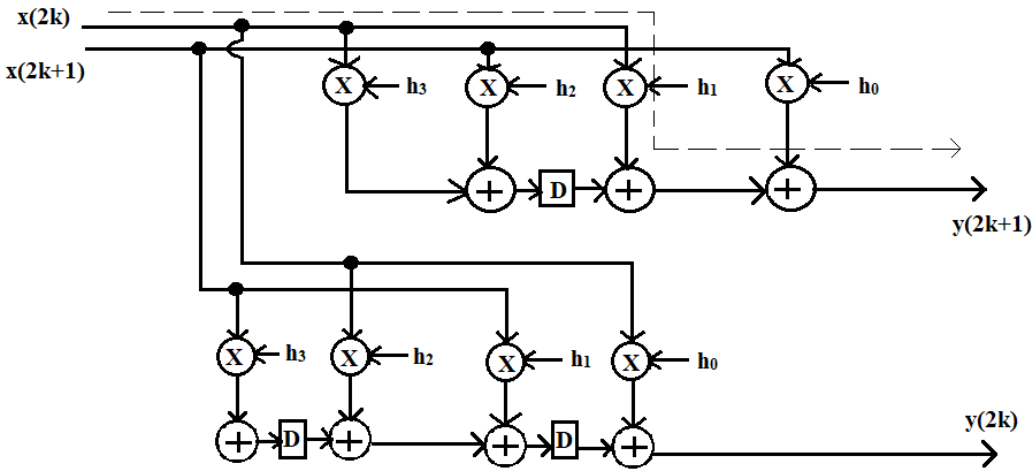

Fig. 4.5: Basic 2-Parallel Filter.

Where h_0, h_1, h_2, h_3 are the filter coefficients, and D indicates the delay of one clock cycle that has to be stored for every one clock cycle.

4.5.2 Area-Efficient 2-FIR Filter

Consider an Area-Efficient 2-FIR filter given in Fig. 4.6. For this filter, booth multiplier, carry look ahead adder and carry look ahead subtractor are used for designing. The dotted line indicates the critical path which is the longest path with no delay or minimum delay. Here also, the inputs and outputs are taken as 16-bit and 32-bit each respectively.

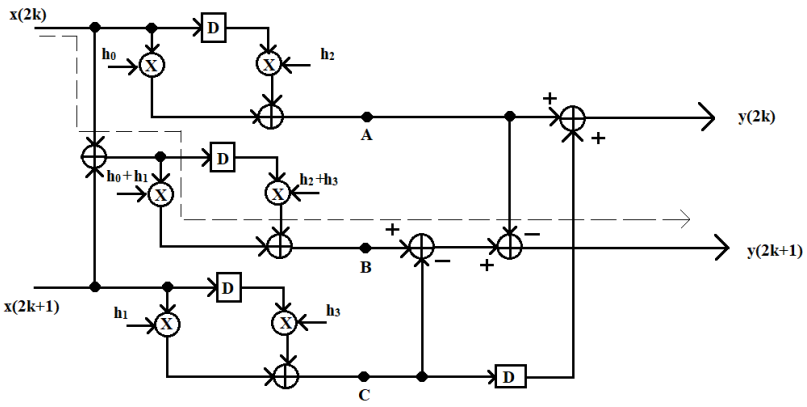

Fig. 4.6: Area-Efficient 2-FIR Filter.

As shown in Fig. 4.6, the formula for basic equation for Area-Efficient 2-Parallel FIR filter is:

$$y(n) = h_0x(n) + h_1x(n-1) + h_2x(n-2) + h_3x(n-3) \tag{4.13}$$

y_A, y_B and y_C indicate the output nodes of A, B and C respectively. Further, we can write the equations as:

$$y_A = h_0x(2k) + h_2x(2k-2) \tag{4.14}$$

$$y_B = (h_0 + h_1)\big(x(2k) + x(2k+1)\big) + (h_2 + h_3)\big(x(2k-2) + x(2k-1)\big) \tag{4.15}$$

$$y_C = h_1x(2k+1) + h_3x(2k-1) \tag{4.16}$$

Using Eq. (4.15) - (4.16) we generate output signals as:

$$y(2k) = y_A + y_C\ [after\ 1\ block\ of\ delay] = h_0x(2k) + h_1x(2k-1) + h_2x(2k-2) + h_3x(2k-3) \tag{4.17}$$

$$y(2k+1) = y_B - y_A - y_C = h_0x(2k+1) + h_1x(2k) + h_2x(2k-1) + h_3x(2k-2) \tag{4.18}$$

Eq. (4.17) and Eq. (4.18) are similar to Eq. (4.11) and Eq. (4.12) respectively.

4.6 Simulation Results

The design works of both the filters are carried out using Xilinx ISE design tool 14.7 and XILINX XC6SLX16-2FTG256C being the FPGA hardware. Table 4.3 and Table 4.4 show the device and power utilization of both the filters respectively. Figure 4.7 and Fig. 4.8 show the RTL schematic for both the filters. Also Figure 4.9 and Fig. 4.10

show the timing diagram of the filters. The Area-Efficient 2-FIR filter consumes more hardware than the Parallel 2-FIR filter; however, there is an exception on the number of bonded IOBs which is less. The fan-out of Area-Efficient filter is also high which makes the filter preferable than the basic Parallel 2-FIR filter.

Table 4.3: Device Utilization of the Filters

Component	Area-Efficient 2-FIR Filter		Parallel 2-FIR Filter	
	Available	Used	Available	Used
Slice Registers	18224	130	18224	48
Slice LUTs	9112	714	9112	20
Logic	9112	668	9112	0
Number of bonded IOBs	186	99	186	101
LUT-FF pairs	714	116	28	20
BUFG/ BUFGMUXs	16	1	16	1
DSP48A1	32	0	32	10
Average Fan-Out	--	4.15	--	1.65

Table 4.4: Power Utilization of the Filters

Type of Power	Area-Efficient 2-FIR Filter	Parallel 2- FIR Filter
Supply Power (mW)	20.38	20.37
Dynamic Power (mW)	0.47	0.46
Static Power (mW)	19.91	19.91

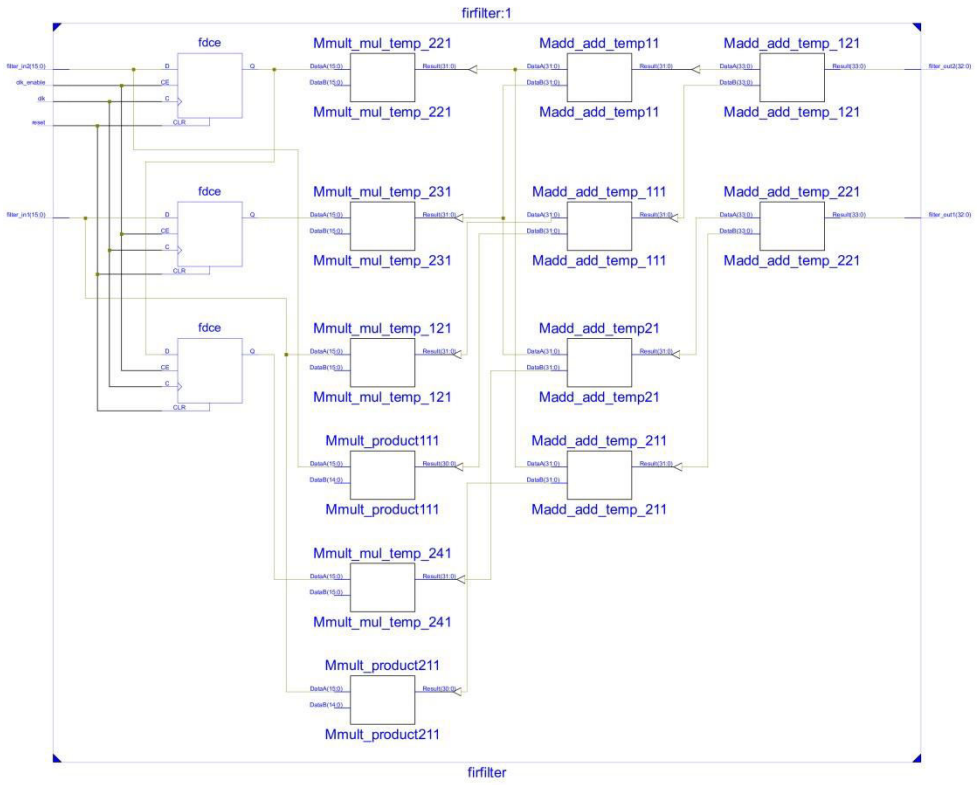

Fig. 4.7: RTL Schematic of basic Parallel 2-FIR Filter.

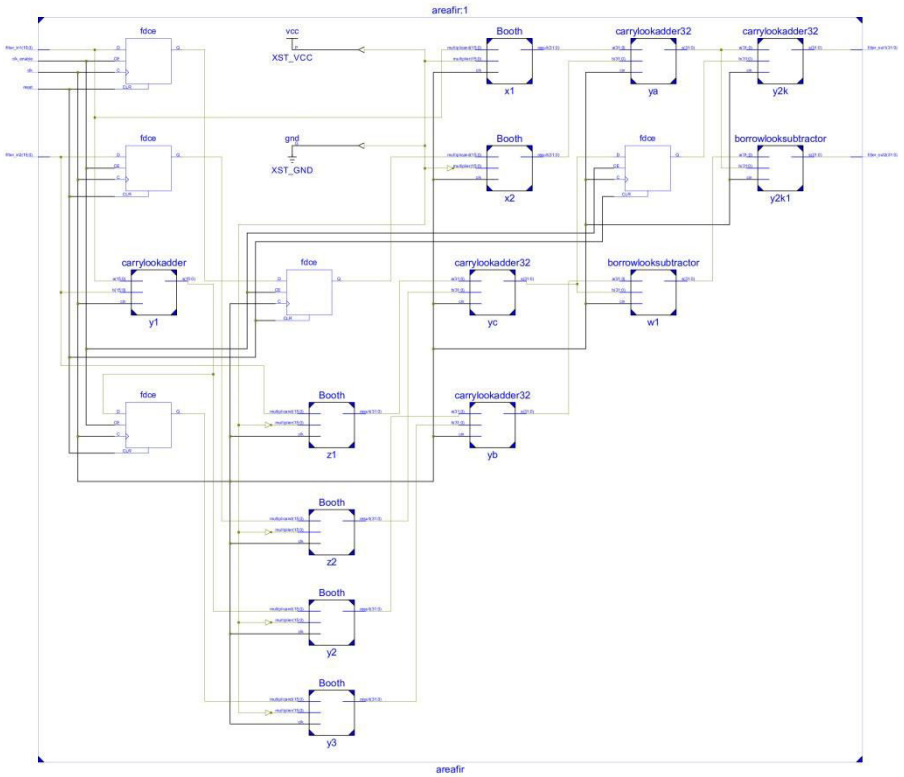

Fig. 4.8: RTL Schematic of Area-Efficient 2-FIR Filter.

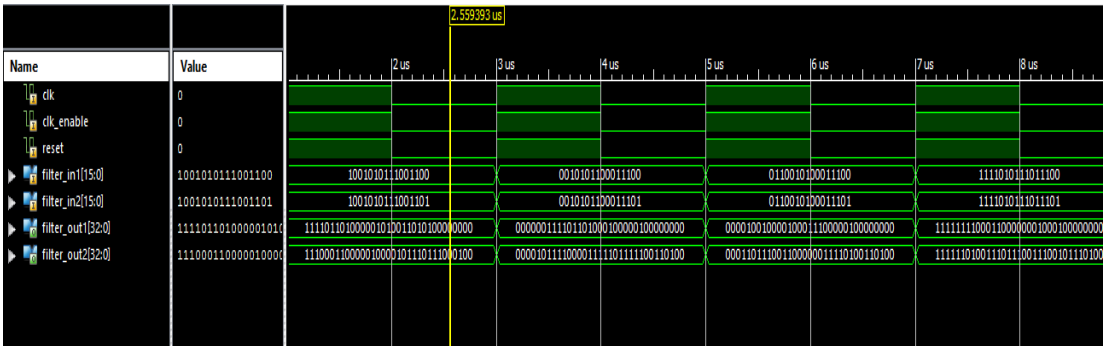

Fig. 4.9: Timing diagram of basic Parallel 2-FIR Filter.

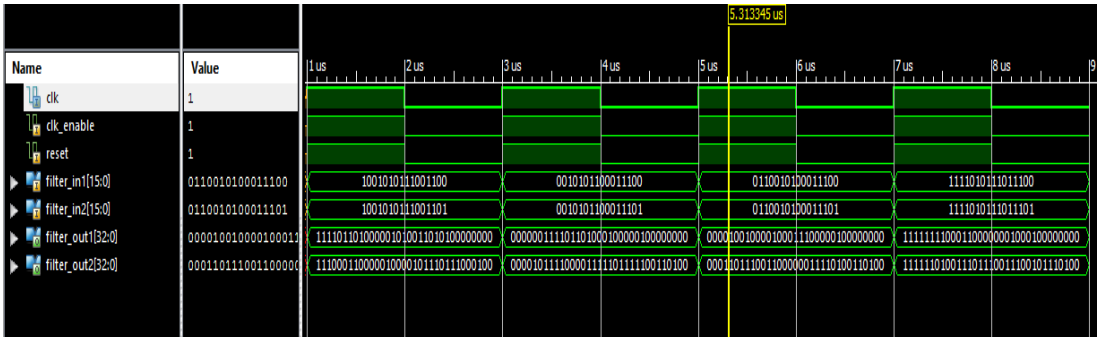

Fig. 4.10: Timing diagram of Area-Efficient 2-FIR Filter.

4.7 Comparison with other FPGA Hardware

The authors of [9] have designed and compared the basic Parallel and Area-Efficient 2-FIR filters using Spartan 3E Starter Board XC3S500E chips being the FPGA Hardware and IMAGE system as the software. Each of the inputs and outputs were 8-bit and 16-bit respectively. Table 4.5 shows the detailed comparison between the parameters of the filters suggested in [9] and the filters designed by proposed method.

Table 4.5: Comparison of the Filters with [9]

Parameters	Phimu and Kumar [9]		Proposed	
	Area-Efficient 2-FIR Filter	Basic Parallel 2-FIR Filter	Area-Efficient 2-FIR Filter	Basic Parallel 2-FIR Filter
Number of external/bonded IOBs	3753 out of 66560 (5%)	3891out of 66560 (5%)	99 out of 186 (53%)	101 out of 186 (54%)
Number of slices	5904 out of 66560 (8%)	6163 out of 66560 (9%)	241 out of 2278 (10%)	7 out of 2278 (1%)
Number of RAMB16s	21 out of 104 (20%)	21 out of 104 (20%)	0 out of 32 (0%)	0 out of 32 (0%)

Table 4.5 shows that the FPGA used in this work shows better performance than the one used in [9]. The Area-Efficient 2-FIR filter requires zero RAMB16s hardware when implemented on XC6SLX16-2FTG256C FPGA. However, the percentage utilization of external/bonded IOBs is much higher than the same when implemented in XC3S500E. This is because XC6SLX16-2FTG256C has less I/O pins.

4.8 Summary

In this chapter, after comparing the Area-Efficient 2-FIR filter with the basic 2-FIR filter we have discussed the working methodology of parallel processing and power consumption required by both the filters. Replacing binary adder, binary subtractor and binary multiplier with CLA adder, CLA subtractor and Booth's multiplier respectively, the delay and speed has been improved. From Table 4.3, it is noticed that the hardware utilization of Area-Efficient 2-FIR filter is much higher than basic Parallel 2-FIR filter. This leads to the fact that the fan-out of the Area-Efficient 2-FIR filter being 2.5 times higher than the basic Parallel 2-FIR filter which is an advantage of the proposed filter. However, looking into Table 4.4, the Area-Efficient 2-FIR filter consumes more dynamic power than the basic Parallel 2-FIR filter, making the overall power a little higher. If the dynamic power is less, then the Area-Efficient filter will be more preferable. Finally, the filters are later compared with the FPGA implementation presented in [9] that shows that the proposed filter require zero RAMB16s hardware in comparison with other works.

Chapter 5: Conclusion

5.1 Conclusion

In this work, two approaches are proposed for the design of FIR filters using Booth's multiplier. The first approach, described in Chapter 3, solves the design problem of a Type-I Band Pass FIR Filter using Booth's multiplier. The results of Type-I Band Pass Filter are compared with existing algorithms like PM and ISCA which demonstrated that Booth's multiplier based design requires zero memory and zero DSP blocks. The second approach, discussed in Chapter 4, deals with the designs of a basic 2-Parallel FIR filter and an Area-Efficient 2-FIR Filter. After comparing these two designs, it is observed that the fan-out of the Area-Efficient filter is higher than the Parallel 2-FIR structure. The proposed designs are also compared with the existing work via Xilinx 14.2 Spartan 3E Starter Board XC3S500E chips and it is observed that the Area-Efficient 2-FIR Filter does not require any RAMB16s hardware.

5.2 Future Work

The primary objective of this work was to optimize the filter design in such a way that it can be implemented by utilizing less area, power, and delay without hampering the quality of the magnitude response. This Thesis work includes the implementation of Type-I FIR filters as well as Parallel and Area-Efficient FIR filters. The same idea can also be adopted for the design of IIR filters. Some strategies may be taken to reduce the power consumption further. One such strategy may include the usage of meta-heuristic algorithms to adjust the coefficients of digital filters properly. The addition of meta-heuristic algorithms can also help in reducing the hardware complexity of digital filters and also minimizing the design error.

REFERENCES

[1] J. G. Proakis and D. G. Manolakis, "Digital signal processing", 3rd ed., Pearson Education, 2007.

[2] X. X. Zheng, J. Yang, S. Y. Yang, W. Chen, L. Y. Huang, and X. Y. Zhang, "Synthesis of Linear-Phase FIR Filters With a Complex Exponential Impulse Response", *in Proc. of IEEE Transactions on Signal Processing*, Vol. 69, pp. 6101-6115, 2021.

[3] R. A. Zitar and A. Al-Dmour, "An Evolutionary FIR Filter Design Method", *in Proc. of Evolutionary Image Analysis and Signal Processing, Studies in Computational Intelligence, Springer*, Vol. 213, pp. 185-200, 2009.

[4] A. Aggarwal, T. Rawat, and D. Upadhyay, "Design of Optimal Digital FIR Filters using Evolutionary and Swarm Optimization Techniques", *in Proc. of AEU -International Journal of Electronics and Communications*, Vol. 70, No. 4, pp. 373-385, 2015.

[5] S. Tsutsumi and K. Suyama, "Design of Digital FIR Filters with discrete coefficients using Ant Colony Optimization", *in Proc. of Electronics and communication in Japan*, Vol. 97, No. 4, pp. 30-37, 2014.

[6] S. Yadav, R. Yadav, A. Kumar, and M. Kumar, "A novel approach for optimal design of digital FIR filter using grasshopper optimization algorithm", *in Proc. of ISA Transactions*, Vol. 108, pp. 196-206, 2021.

[7] R. S. Chauhan and S. K. Arya, "An Optimal Design of FIR Digital Filter Using Genetic Algorithm", *in Proc. of Contemporary Computing. IC3 Communications in Computer and Information Science, Springer*, Vol. 168, pp. 51-56, 2011.

[8] P. U. Kumar, G.R.C.K. Sarma, S. M. Das, and M.A.V. Kamalnath, "Design of Optimal Digital Fir Filter Using Particle Swarm Optimization Algorithm", *in Proc. of Advances in Computational Science, Engineering and Information Technology, Advances in Intelligent Systems and Computing, Springer*, Vol. 225, pp. 187-196, 2013.

[9] L. K. Phimu and M. Kumar, "Design and implementation of area efficient 2-parallel filters on FPGA using image system", *in Proc. of International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, Chennai, India, pp. 1086-1091, 2017.

[10] K. Kamdar, A. Acharya, and P. Kadam, “Low power multiplier design using adiabatic SCRL logic”, *in Proc. of IEEE International Conference on Circuits and Systems (ICCS)*, pp. 255-260, 2017.

[11] N. U. Sadad, A. Afrin, and M. N. I. Mondal, “Synchronous and Asynchronous Implementation of Radix-2 Booth Multiplication Algorithm”, *in Proc. of 3rd International Conference on Electrical & Electronic Engineering (ICEEE)*, Rajshahi, Bangladesh, pp. 93-96, 2021.

[12] A. S. Tariq, R. Amin, M. N. I. Mondal, and M. A. Hossain, “Faster implementation of Booth's algorithm using FPGA”, *in Proc. of 2nd International Conference on Electrical, Computer & Telecommunication Engineering (ICECTE)*, Rajshahi, Bangladesh, pp. 1-4, 2016.

[13] P. Asef, R. B. Perpiñà, M. R. Barzegaran, A. Lapthorn, and D. Mewes, “Multiobjective Design Optimization Using Dual-Level Response Surface Methodology and Booth's Algorithm for Permanent Magnet Synchronous Generators”, *in Proc. of IEEE Transactions on Energy Conversion*, Vol. 33, no. 2, pp. 652-659, 2018.

[14] K. Talawar and P. Hosamani, “Ultra area efficient reversible Quantum Radix-2 booth's recoding multiplier for low power applications”, *in Proc. of IEEE International Conference on Computational Intelligence and Computing Research*, Coimbatore, India, pp. 1-4, 2014.

[15] S. S. Ghoreishi, M. A. Pourmina, H. Bozorgi, and M. Dousti, “High Speed RSA Implementation Based on Modified Booth's Technique and Montgomery's Multiplication for FPGA Platform”, *in Proc. of Second International Conference on Advances in Circuits, Electronics and Micro-electronics*, Sliema, Malta, pp. 86-93, 2009.

[16] S. K. Saha, R. Kar, D. Mandal, and S. P. Ghoshal, “An Efficient Craziness Based Particle Swarm Optimization Technique for Optimal IIR Filter Design”, *in Proc. of Transactions on Computational Science XXI, Lecture Notes in Computer Science, Springer*, Vol. 8160, pp. 230-252, 2013.

[17] N. Dasan, R. Badarinath, K. P. Ray, and A. Vengadarajan, “Sectoral Side-lobe Suppression using Parks-McClellan Algorithm in a Planar Array”, *in Proc. of 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, Bangalore, India, pp. 103-107, 2018.

[18] A. Sikder, P. Bhattacharjee, S. Chowdhury, S. Dhabal, and P. Venkateswaran, "Design of Band-pass FIR Filter using Improved Sine Cosine Algorithm and its Implementation on FPGA", *in Proc. of IEEE Region 10 Symposium (TENSYMP)*, Kolkata, India, pp. 310-315, 2019.

[19] K. K. Parhi, "VLSI Digital Signal Processing Systems – Design and Implementation", John Wiley & Sons, 1st ed., 1999.

[20] D. A. Kumar, M. SaiKumar, and P. Samundiswary, "Design and study of modified parallel FIR filter using fast FIR algorithm and symmetric convolution", *in Proc. of International Conference on Information Communication and Embedded Systems (ICICES2014)*, pp. 1-6, 2014.

[21] A. Nandal, T. Vigneswarn, A. K. Rana, and A. Dhaka, "An Efficient 256-Tap Parallel FIR Digital Filter Implementation Using Distributed Arithmetic Architecture", *Procedia Computer Science*, Vol. 54, pp. 605-611, 2015.

[22] K. A. Rao, A. Kumar, and N. Purohit, "Efficient Implementation for 3-Parallel Linear-Phase FIR Digital Odd Length Filters", *in Proc. of IEEE 4th Conference on Information & Communication Technology (CICT)*, Chennai, India, pp. 1-6, 2020.

[23] N. Kandasamy, N. Telagam, N. Anughna, T. L. Sai, M. R. Reddy, and A. Rupanjali, "Transposed 3 Tap FIR Filter Design Using Consolidation of Pipelining and Parallel Processing Technique", *in Proc. of IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, pp. 1-6, 2020.

[24] S. Annangi and R. Puli, "ASIC implementation of efficient 16-parallel fast FIR algorithm filter structure", *in Proc. of 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, Delhi, India, pp. 1-5, 2017.

[25] B. Koyada, N. Meghana, M. O. Jaleel, and P. R. Jeripotula, "A comparative study on adders", *in Proc. of International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, Chennai, India, pp. 2226-2230, 2017.

[26] S. Nagaraj, G. M. S. Reddy, and S. A. Mastani, "Analysis of different Adders using CMOS, CPL and DPL logic", *in Proc. of 14th IEEE India Council International Conference (INDICON)*, pp. 1-6, 2017.

[27] M. R. Ravula, A. Potharaju, and R. P. Vidyadhar, "Designing Carry Look Ahead Adder to Enrich Performance using One Bit Hybrid Full Adder", *in Proc. of International Conference on Electronics and Renewable Systems (ICEARS)*, Tuticorin, India, pp. 86-89, 2022.

[28] D. Sasidaran, A. Azam, K. E. Nelson, and M. A. Soderstrand, "FPGA implementation of a tunable band-pass filter using the basic heterodyne block", *in Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 1093-1096, Vol. 2, 2001.

[29] C. Charoensak and S. S. Abeysekera, "FPGA implementation of efficient Kalman band-pass sigma-delta filter for application in FM demodulation", *in Proc. of IEEE International SOC Conference*, pp. 137-138, 2004.

[30] C. Guo-wei and W. Feng-ying, "The implementation of FIR low-pass filter based on FPGA and DA", *in Proc. of Fourth International Conference on Intelligent Control and Information Processing (ICICIP)*, pp. 604-608, 2013.

[31] O. KalaiPriya, S. Ramasamy, and D. Ebenezer, "VLSI implementation of nonlinear variable cut-off high pass filter algorithm", *in Proc. of 3rd International Conference on Electronics Computer Technology*, pp. 275-278, 2011.

[32] K. A. Toker, S. Özen, and A. Arsal, "FPGA implementation of a low complexity fading filter for multipath Rayleigh fading simulator", *in Proc. of XXXth URSI General Assembly and Scientific Symposium*, pp. 1-4, 2011.

[33] Y. Zhou and P. Shi, "Distributed Arithmetic for FIR Filter implementation on FPGA", *in Proc. of International Conference on Multimedia Technology*, pp. 294-297, 2011.

[34] A. Das, S. Dash, A. K. Sahoo, and B. C. Babu, "Design and implementation of FPGA based linear all digital phase-locked loop", *in Proc. of Annual IEEE India Conference (INDICON)*, pp. 280-285, 2012.

[35] A. Singhal, A. Goen, and T. Mohapatra, "FPGA implementation and power efficient CORDIC based ADPLL for signal processing and application", *in Proc. of 7th International Conference on Communication Systems and Network Technologies (CSNT)*, pp. 325-329, 2017.

[36] P. K. Meher, S. Chandrasekaran, and A. Amira, "FPGA Realization of FIR Filters by Efficient and Flexible Systolization Using Distributed Arithmetic", *in Proc. of IEEE Transactions on Signal Processing*, Vol. 56, No. 7, pp. 3009-3017, 2008.

[37] T. Dutta, R. M. Aich, S. Dhabal, and P. Venkateswaran, "Finite Impulse Response Filter Design using Grasshopper Optimization Algorithm and Implementation on FPGA", *in Proc. of IEEE Applied Signal Processing Conference (ASPCON)*, pp. 313-317, 2020.

[38] J. B. Evans, "Efficient FIR filter architectures suitable for FPGA implementation", *in Proc. of IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, Vol. 41, no. 7, pp. 490-493, 1994.

[39] M. A. Eshtawie and M. Othman, "FPGA Implementation of an Optimized Coefficients Pulse Shaping FIR Filters", *in Proc. of IEEE International Conference on Semiconductor Electronics*, pp. 454-458, 2006.

[40] G. Jinding, H. Yubao, and S. Long, "Design and FPGA Implementation of Linear FIR Low-pass Filter Based on Kaiser Window Function", *in Proc. of International Conference on Intelligent Computation Technology and Automation*, pp. 496-498, 2011.

[41] R. Chand, P. Tripathi, A. Mathur, and K. C. Ray, "FPGA implementation of fast FIR low pass filter for EMG removal from ECG signal", *in Proc. of International Conference on Power, Control and Embedded Systems*, pp. 1-5, 2010.

[42] L. Chen, M. Liu, J. Yang, J. Wu, and Z. Dai, "Structure Evolution based Optimization algorithm for Low pass IIR Filter Design", *in Proc. of International Journal of Computational Intelligence Systems,* Vol. 10, No. 1, pp. 1036 – 1055, 2017.

[43] Neha and A. P. Singh, "Design of Linear Phase Low Pass FIR Filter using Particle Swarm Optimization Algorithm", *in Proc. of International Journal of Computer Applications*, Vol. 98, No.3, pp. 40-44, July 2014.

[44] P. Gowtham, S. Sowndarya, and N. Pachauri, "Simulated Anneling Optimization Low-Pass FIR Filter for Biomedical Signals", *in Proc. of Advances in Mechanical Engineering, Springer*, pp. 741-748, 2021.

[45] S. K. Sarangi, R. Panda, and A. Abraham, "Design of optimal low-pass filter by a new Levy swallow swarm algorithm", *in Proc. of Soft Computing*, Vol. 24, pp. 18113–18128, 2020.

[46] P. Bertsias and C. Psychalinos, "Differentiator based fractional-order high pass filter designs", *in Proc. of 7th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, pp. 1-4, 2018.

[47] S. Mandal, S. P. Ghoshal, R. Kar, and D. Mandal, "Design of optimal linear phase FIR high pass filter using craziness based particle swarm optimization technique", *in Proc. of Journal of King Saud University - Computer and Information Sciences*, Vol. 24, No. 1, pp. 83-92, 2012.

[48] A. Sarangi, R. Lenka, and S. K. Sarangi, "Design of High Pass Filter Using PSO with Gaussian Mutation", *in Proc. of Swarm, Evolutionary and Memetic Computing, Springer International Publishing*, Vol. 8947, pp. 471-479, 2015.

[49] R. Islam, R. Kar, D. Mandal, and S. P. Ghoshal, "Design and Simulation of FIR High Pass Filter Using Gravitational Search Algorithm", *in Proc. of Swarm, Evolutionary, and Memetic Computing. SEMCCO, Lecture Notes in Computer Science, Springer*, Vol. 8297, pp. 547-557, 2013.

[50] A. Aggarwal, T. K. Rawat, M. Kumar, and D. K. Upadhyay, "Optimal design of FIR high pass filter based on L1 error approximation using real coded genetic algorithm", *in Proc. of Engineering Science and Technology, an International Journal*, Vol. 18, No. 4, pp. 594-602, 2015.

[51] V. Kumar, R. Mehra, and Shallu, "Reconfigurable band-pass filter using Kaiser window for satellite communication", *in Proc. of 2nd International Conference on Next Generation Computing Technologies (NGCT)*, pp. 409-413, 2016.

[52] S. K. Saha, R. Kar, D. Mandal, and S. P. Ghoshal, "Design and simulation of FIR band pass and band stop filters using gravitational search algorithm", *in Proc. of Memetic Computing*, Vol. 5, pp. 311-321, 2013.

[53] S. Mandal, P. Mallick, D. Mandal, R. Kar, and S. P. Ghoshal, "Optimal FIR band-pass filter design using novel particle swarm optimization algorithm", *in Proc. of SHUSER IEEE Symposium on Humanities, Science and Engineering Research*, pp. 141-146, 2012.

[54] N. O. Parchin, A. Ullah, M. Alibakhshikenari, and R. A. Abd-Alhameed, "UWB Band-Pass Filter with Dual Notched Bands Using Electromagnetic Band Gap Structure", *in Proc. of 4th IEEE Middle East and North Africa COMMunications Conference (MENACOMM)*, pp. 248-251, 2022.

[55] D. Wang, X. Chen, and L. Gao, "Comparative experimental research on the LEMP protection effects of RF surge arresters with low-pass and band-pass frequency characteristics", *in Proc. of 7th IEEE International Symposium on Microwave, Antenna, Propagation, and EMC Technologies (MAPE)*, Xi'an, China, pp. 200-202, 2017.

[56] S. Mandal, S. P. Ghosal, R. Kar, and D. Mandal, "Optimal linear phase finite impulse response band pass filter design using craziness based particle swarm optimization algorithm", *in Proc. of J. Shanghai Jiaotong Univ. (Sci.)*, Vol. 16**,** pp. 696–703, 2011.

[57] J. Bae and C. Nguyen, "New dual-band band-pass filter design with enhanced dual-band skirt characteristics", *in Proc. of Asia-Pacific Microwave Conference Proceedings (APMC), Seoul, Korea (South)*, pp. 599-901, 2013.

[58] Z. Quan, S. A. Xuan, S. Han, and W. H. Kwon, "Parallel Implementation of M-Step Kalman FIR Filter for Linear Discrete Time-invariant Systems", *in Proc. of SICE-ICASE International Joint Conference*, Busan, Korea (South), pp. 760-764, 2006.

[59] Q. Tian, Y. Wang, G. Liu, X. Liu, J. Diao, and H. Xu, "Hardware-Efficient Parallel FIR Filter Structure Based on Modified Cook-Toom Algorithm", *in Proc. of IEEE Asia Pacific Conference on Circuits and Systems (APCCAS),* Chengdu, China, 2018, pp. 342-345, 2018.

[60] G. Deepak, P. K. Meher, and A. Sluzek, "Performance Characteristics of Parallel and Pipelined Implementation of FIR Filters in FPGA Platform", *in Proc. of International Symposium on Signals, Circuits and Systems*, Iasi, Romania, pp. 1-4, 2007.

[61] K. Debnath, S. Dhabal, and P. Venkateswaran, "Design of High-Pass FIR Filter using Arithmetic Optimization Algorithm and its FPGA Implementation", *in Proc. of IEEE Region 10 Symposium (TENSYMP)*, Mumbai, India, pp. 1-6, 2022.

[62] X. Zhang, L. Tu, D. Chen, Y. Yuan, K. Huang, and Z. Wang, "Parallel distributed arithmetic FIR filter design based on 4:2 compressors on Xilinx FPGAs", *in Proc. of 4th International Conference on Signal Processing, Computing and Control (ISPCC)*, Solan, India, pp. 43-49, 2017.