

PERFORMANCE ANALYSIS OF AN EFFICIENT SYSTOLIC TRANSFORMATION FIR FILTER FOR VLSI APPLICATIONS

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Electronics and Communication

Engineering

By

38130158– Pavithra B

38130212– Soniya M S



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
SCHOOL OF ELECTRICAL AND ELECTRONICS**

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY (DEEMED TO BE
UNIVERSITY) Accredited with Grade “A” by NAAC I 12B Status by UGC I Approved by

AICTE JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

APRIL – 2022

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY) Accredited with “A” grade by NAAC I 12B Status by UGC I

Approved by AICTE Jeppiaar Nagar, Rajiv Gandhi Salai , Chennai – 600 119 www.sathyabama.ac.in



DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the Bonafide work of PAVITHRA B (Reg.No: 38130158) and SONIYA M S (Reg.No: 38130212) who carried out the project entitled PERFORMANCE ANALYSIS OF AN EFFICIENT SYSTOLIC TRANSFORMATION FIR FILTER FOR VLSI APPLICATIONS under my supervision from November 2021 to April 2022.

Internal Guide Dr .M S Godwin Premi M.E,Ph.D.

Head of the Department: DR. RAVI, M.E,Ph.D.

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

We, **PAVITHRA B**(Reg. No.: 38130158) and **SONIYA M S** (Reg. No.: 38130212) hereby declare that the Project Report entitled **PERFORMANCE ANALYSIS OF AN EFFICIENT SYSTOLIC TRANSFORMATION FIR FILTER FOR VLSI APPLICATIONS** by us under the guidance of **Dr .M S GODWIN PREMI** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **ELECTRONICS AND COMMUNICATION ENGINEERING**.

PAVITHRA B – 38130158

SONIYA M S –38130212

DATE:

PLACE: CHENNAI

SIGNATURE OF CANDIDATES

ACKNOWLEDGEMENT

I am very pleased to acknowledge my sincere thanks to the **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully.

We are grateful to them. I convey our thanks to **Dr. N. M. NANDHITHA, M.E., Ph.D.**, Dean, School of Electrical and Electronics and **Dr. RAVI, M.E., Ph.D.**, Head of the Department, Department of Electronics and Communication Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr .M S GODWIN PREMI** for her valuable guidance, suggestions, and constant encouragement paved way for the successful completion of my project work.

I wish to express our thanks to all Teaching and Non-teaching staff members of the Department of Electronics and Instrumentation Engineering who were helpful in many ways for the completion of the project.

CONTENTS

CHAPTER NUMBER	TITLE	PAGE NO
	ABSTRACT	6
	LIST OF FIGURES	7
1	INTRODUCTION	8
2	LITERATURE SURVEY	19
3	DESIGN METHODOLOGY	21
4	EXISTING SYSTEM	24
	4.1 BLOCK DIAGRAM	
	4.2 XILINX	26
	4.3 MODEL SIM	31
	4.4 OUTPUT SCREEN SHOT	34
	4.5 POWER CONSUMPTION	34
	4.6 SIMULATION	35
5	CONCLUSION	48

ABSTRACT

In digital signal processing and communication systems, FIR filters have important role. One of the main challenges in Very Large Scale Integration(VLSI) signal processing is the FIR filter structure with optimized parameters. These filters made up of many adders and multipliers. In FIR filter, multiplier consumes high amount of power. The most important three areas in VLSI are power, area, and delay. This paper compares different types of fixed point FIR filter architectures and analyze the different performance parameters such as area, hardware performance and delay. Different FIR filter architectures such as the conventional method, systolic architecture, associativity transformation and combination of systolic and associativity transformation architectures are consider. We performance a 4, 8 and 16 tap filters using the above mentioned architectures and compared their performance characteristics.

LIST OF FIGURES

FIGURE NUMBER	FIGURE NAME	PAGE NO
1.1	FIR Filter	9
1.2	Logical Structure of FIR Filter	10
1.3	Frequency Response of an FIR Filter	11
4.1	Systolic Transformation FIR Filter	24
4.2	Block Diagram	25
4.3	Critical Path optimisation	26
4.4	RTL Schamatic	34
4.5	Power Consumption	34
4.6	Stimulation	35
4.7	Device Utilization	37

CHAPTER 1

Introduction

FINITE-IMPULSE response (FIR) digital filter is widely used in several digital signal processing applications, such as speech processing, loud speaker equalization, echo cancellation, adaptive noise cancellation, and various communication applications, including software-defined radio (SDR). Many of these applications require FIR filters of large order to meet the stringent frequency specifications. Very often these filters need to support high sampling rate for high-speed digital communication. The number of multiplications and additions required for each filter output, however, increases linearly with the filter order. Since there is no redundant computation available in the FIR filter algorithm, real-time implementation of a large order FIR filter in a resource constrained environment is a challenging task. Filter coefficients very often remain constant and known a priori in signal processing applications. This feature has been utilized to reduce the complexity of realization of multiplications. Several designs have been suggested by various researchers for efficient realization of FIR filters (having fixed coefficients) using distributed arithmetic (DA) and multiple constant multiplication (MCM) methods. DA-based designs use lookup tables (LUTs) to store pre computed results to reduce the computational complexity. The MCM method on the other hand reduces the number of additions required for the realization of multiplications by common sub expression sharing, when a given input is multiplied with a set of constants. The MCM scheme is more effective, when a common operand is multiplied with more number of constants. Therefore, the MCM scheme is suitable for the implementation of large order FIR filters with fixed coefficients. But, MCM blocks can be formed only in the transpose form configuration of FIR filters. Block-processing method is popularly used to derive high-throughput hardware structures. It not only provides throughput-scalable design but also improves the area-delay efficiency. The derivation of block-based FIR structure is straightforward when direct-form configuration is used, whereas the transpose form configuration does not directly support block processing. But, to take the computational advantage of the MCM, FIR filter is required to be realized by transpose form configuration. Apart from that, transpose form structures are inherently pipelined and supposed to offer higher operating frequency to support higher sampling rate. There are some

applications, such as SDR channelize, where FIR filters need to be implemented in a reconfigurable hardware to support multi standard wireless communication [6]. Several designs have been suggested during the last decade for efficient realization of reconfigurable FIR (RFIR) using general multipliers and constant multiplication schemes. A RFIR filter architecture using computation sharing vector-scaling technique has been proposed have proposed a canonic sign digit (CSD)-based RFIR filter, where the nonzero CSD values are modified to reduce the precision of filter coefficients without significant impact on filter behaviour. But, the reconfiguration overhead is significantly large and does not provide an area-delay efficient structure.

In digital signal processing, an FIR is a filter whose impulse response is of finite period, as a result of it settles to zero in finite time. This is often in distinction to IIR filters, which can have internal feedback and will still respond indefinitely. The impulse response of an Nth order discrete time FIR filter takes precisely N+1 samples before it then settles to zero. FIR filters are most popular kind of filters executed in software and these filters can be continuous time, analog or digital and discrete time. Special types of FIR filters are namely, Boxcar, Hilbert Transformer, Differentiator, Lth-Band and Raised-Cosine.

1.1 What is FIR Filter?

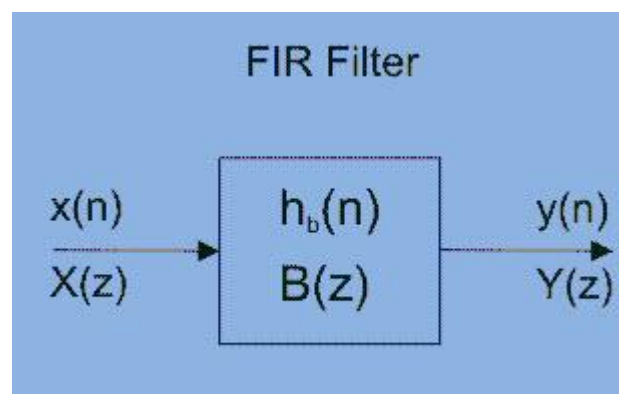


Fig.1.1

The term FIR abbreviation is “Finite Impulse Response” and it is one of two main types of digital filters used in DSP applications. Filters are signal conditioners and function of each filter is, it allows an AC components and blocks DC components. The best example of the filter is a phone line, which acts as a filter. Because, it limits frequencies to a range significantly smaller than the range of human beings can hear frequencies.

1.2 FIR Filters for Digital Signal Processing

There are various kinds of filters, namely LPF, HPF, BPF, BSF. A LPF allows only low frequency signals through to its o/p, so this filter is used to eliminate high frequencies. A LPF is convenient for controlling the highest range of frequencies in an audio signal. An HPF is quite opposite to LPF. Because, it rejects only frequency components below some threshold. The best example of the HPF is, cutting out the 60Hz audible AC power, which can be selected up as noise associated almost any signal in the USA.

The alternative of IR filter is a DSP filter which can also be IIR. IIR filters use feedback, so when you i/p an impulse the o/p theoretically rings forever. The terms used for describing IR filters are Tap, impulse response, MAC (multiply accumulate), delay line, transition band and circular buffer.

1.3 Design Methods of FIR Filter

The design methods of FIR filter are based on approximation of ideal filter. The ensuing filter approaches the perfect characteristic because the order of the filter will increase, so creating the filter and its implementation are additional complicated.

The design process starts with necessities and specifications of the FIR filter. The method used in the design process of the filter depends upon the implementation and specifications. There are many advantages and disadvantages of the design methods. Thus, it is very significant to elect the right method for FIR filter design. Due to efficiency and simplicity of the FIR filter, most commonly window method is used. The other method sampling frequency method is also very simple to use, but there is a small attenuation in the stopband.

1.4 Logical Structure of FIR Filter

A FIR filter is used to implement almost any type of digital frequency response. Usually these filters are designed with a multiplier, adders and a series of delays to create the output of the filter. The following figure shows the basic FIR filter diagram with N length. The result of delays operates on input samples. The values of h_k are the coefficients which are used for multiplication. So that the o/p at a time is the summation of all the delayed samples multiplied by the appropriate coefficients.

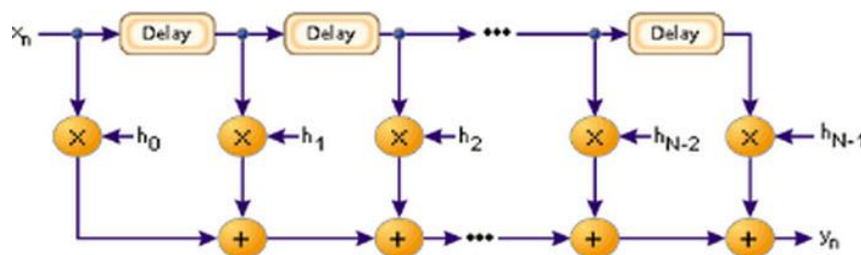


Fig 1.2

Logical Structure of FIR Filter

The filter design can be defined as, it is the process of choosing the length and coefficients of the filter. The intention is to set the parameters so that the required parameters like a stop band and pass band will give the result from running the filter. Most of the engineers use MATLAB software to design the filter.

Usually, filters are defined by their responses to the separate frequency components that found the i/p signal. The responses of the filters are classified into three types based on the frequencies such as stop band, pass band and transition band. The response of the passband is the filter's effect on frequency components that are delivered through mostly unaffected.

Frequencies in a filter's stopband are, by difference, highly reduced. The transition band signifies the frequencies in the middle, which may receive some reduction, but are not detached completely from the o/p signal.

1.5 Frequency Response of an FIR Filter

The frequency response plot of the filter is shown below, where ω_p is the passband ending frequency, ω_s is the stopband beginning frequency, A_s is the amount of attenuation in the stopband. Frequencies b/n ω_p and ω_s drop in the transition band and are reduced to some lesser degree. That confirms that the filter meets the preferred specifications includes transition bandwidth, ripple, filter's length and coefficients. The longer the filter, the more finely the response can be tuned. With the N length and coefficients, float $h[N] = \{\dots\dots\dots\}$, decided upon, the FIR filter implementation is fairly straightforward.

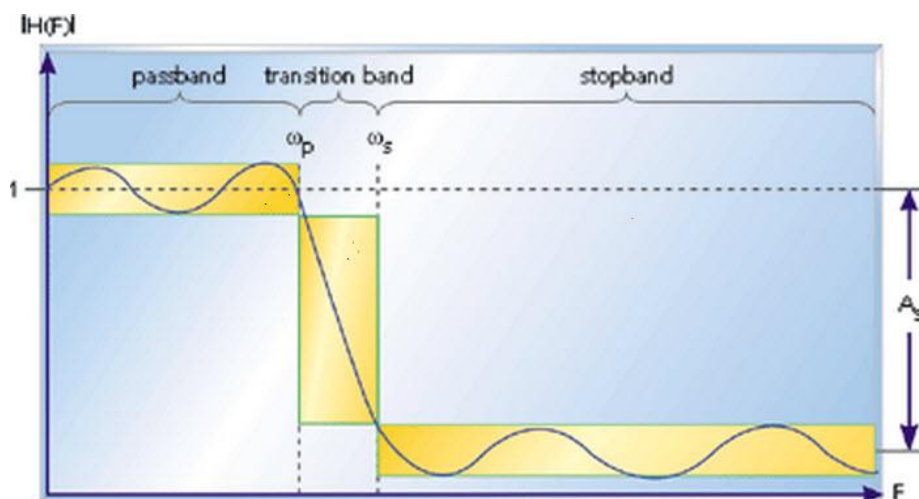


Fig1.3 Frequency Response of an FIR Filter

1.6 Z Transform of an FIR Filter is

For an N-tap FIR filter with $h(k)$ coefficient, then the o/p is defined as $y(n)=h(0)x(n) + h(1)x(n-1) + h(2)x(n-2) + \dots\dots\dots h(N-1)x(n-N-1)$

The Z-transform of the filter is

$H(z)=h(0)z^{-0} + h(1)z^{-1} + h(2)z^{-2} + \dots\dots\dots h(N-1)z^{-(N-1)}$ or

$$H(z) = \sum_{n=0}^{N-1} h(n)z^{-n}$$

Transfer Function of FIR Filter

$$H(z) \stackrel{\text{def}}{=} \sum_{n=-\infty}^{\infty} h[n] \cdot z^{-n}.$$

The Frequency Response Formula for an FIR Filter

$$H(j\omega) = \sum_{n=0}^{N-1} h(n)[\cos(n\omega) - j\sin(n\omega)]$$

DC Gain of an FIR filter is

$$H(j\omega) = \sum_{n=0}^{N-1} h(n)[\cos(n\omega) - j\sin(n\omega)]$$

The applications of FIR filters mainly involve in digital communications in the intermediate frequency stages of the receiver. For instance, a digital radio receives and converts the analog signal to the intermediate frequency and then converts it to digital using with a digital to analog converter. Then uses the finite impulse response to choose the preferred frequency. It is used in software radio, that permits easily adaptable filters with good rejection and without changing hardware.

Thus, this is all about FIR filter, FIR filter design, logical structure and frequency response of FIR filters. We hope that you have got a better understanding of this concept. Furthermore, any queries regarding this topic and applications, please give your suggestions and comments in the comment section below. Here is a question for you, what is the difference between FIR and IIR filter.

1.6 Introduction to VLSI

1.6.1 Very-large-scale integration (VLSI) is the process of creating an integrated circuit (IC) by combining thousands of transistors into a single chip. VLSI began in the 1970s when complex semiconductor and communication technologies were being developed. The microprocessor is a VLSI device. Before the introduction of VLSI technology most ICs had a limited set of functions they could perform. An electronic circuit might consist of a CPU, ROM, RAM and other glue logic. VLSI lets IC designers add all of these into one chip.

1.6.2 Very large-scale integration (VLSI) is the process of integrating or embedding hundreds of thousands of transistors on a single silicon semiconductor microchip. VLSI technology was conceived in the late 1970s when advanced level computer processor microchips were under development .VLSI is a successor to large-scale integration (LSI), medium-scale integration (MSI) and small-scale integration (SSI) technologies.

1.6.3 Technical explains Very Large-Scale Integration (VLSI)

VLSI is one of the most widely used technologies for microchip processors, integrated circuits (IC) and component designing. It was initially designed to support hundreds of thousands of transistor gates on a microchip which, as of 2012, exceeded several billion. All of these transistors are remarkably integrated and embedded within a microchip that has shrunk over time but still has the capacity to hold enormous amounts of transistors.

The first 1 mega byte RAM was built on top of VLSI design principles and included more than one million transistors on its microchip dye.

1.6.4 History

During the mid-1920s, several inventors attempted devices that were intended to control current in solid-state diodes and convert them into triodes. Success had to wait until after WWII, during which the attempt to improve silicon and germanium crystals for use as radar detectors led to improvements in fabrication and in the understanding of quantum mechanical states of carriers in semiconductors. Then scientists who had been diverted to radar

development returned to solid-state device development. With the invention of transistors at Bell Labs in 1947, the field of electronics shifted from vacuum tubes to solid-state devices.

With the small and effective transistor at their hands, electrical engineers of the 1950s saw the possibilities of constructing far more advanced circuits. As the complexity of circuits grew, problems arose.

One problem was the size of the circuit. A complex circuit, like a computer, was dependent on speed. If the components of the computer were too large or the wires interconnecting them too long, the electric signals couldn't travel fast enough through the circuit, thus making the computer too slow to be effective.

Jack Kilby at Texas Instruments found a solution to this problem in 1958. Kilby's idea was to make all the components and the chip out of the same block (monolith) of semiconductor material. Kilby presented his idea to his superiors, and was allowed to build a test version of his circuit. In September 1958, he had his first integrated circuit ready. Although the first integrated circuit was crude and had some problems, the idea was ground breaking. By making all the parts out of the same block of material and adding the metal needed to connect them as a layer on top of it, there was no need for discrete components. No more wires and components had to be assembled manually. The circuits could be made smaller, and the manufacturing process could be automated. From here, the idea of integrating all components on a single silicon wafer came into existence, which led to development in small-scale integration (SSI) in the early 1960s, medium-scale integration (MSI) in the late 1960s, and then large-scale integration (LSI) as well as VLSI in the 1970s and 1980s, with tens of thousands of transistors on a single chip (later hundreds of thousands, then millions, and now billions (10^9)).

1.7 What is VLSI?

VLSI stands for "Very Large Scale Integration". This is the field which involves packing more and more logic devices into smaller and smaller areas. Thanks to VLSI, circuits that would have taken board full of space can now be put into a small space few millimeters across! This has opened up a big opportunity to do things that were not possible before. VLSI circuits are everywhere ... your computer, your car, your brand new state-of-the-art digital

camera, the cell-phones, and what have you. All this involves a lot of expertise on many fronts within the same field, which we will look at in later sections.

VLSI has been around for a long time, there is nothing new about it ... but as a side effect of advances in the world of computers, there has been a dramatic proliferation of tools that can be used to design VLSI circuits. Alongside, obeying Moore's law, the capability of an IC has increased exponentially over the years, in terms of computation power, utilisation of available area, yield. The combined effect of these two advances is that people can now put diverse functionality into the IC's, opening up new frontiers. Examples are embedded systems, where intelligent devices are put inside everyday objects, and ubiquitous computing where small computing devices proliferate to such an extent that even the shoes you wear may actually do something useful like monitoring your heartbeats! These two fields are kind of related, and getting into their description can easily lead to another article.

1.7.1 DEALING WITH VLSI CIRCUITS

Digital VLSI circuits are predominantly CMOS based. The way normal blocks like latches and gates are implemented is different from what students have seen so far, but the behaviour remains the same. All the miniaturisation involves new things to consider. A lot of thought has to go into actual implementations as well as design. Let us look at some of the factors involved ...

1. Circuit Delays. Large complicated circuits running at very high frequencies have one big problem to tackle - the problem of delays in propagation of signals through gates and wires ... even for areas a few micrometers across! The operation speed is so large that as the delays add up, they can actually become comparable to the clock speeds.

2. Power. Another effect of high operation frequencies is increased consumption of power. This has two-fold effect - devices consume batteries faster, and heat dissipation increases. Coupled with the fact that surface areas have decreased, heat poses a major threat to the stability of the circuit itself.

3. Layout. Laying out the circuit components is task common to all branches of electronics. What is so special in our case is that there are many possible ways to do this; there can be multiple layers of different materials on the same silicon, there can be different arrangements of the smaller parts for the same component and so on.

The power dissipation and speed in a circuit present a trade-off; if we try to optimise on one, the other is affected. The choice between the two is determined by the way we chose the layout the circuit components. Layout can also affect the fabrication of VLSI chips, making it either easy or difficult to implement the components on the silicon.

1.7.2 THE VLSI DESIGN PROCESS

A typical digital design flow is as follows:

Specification

Architecture

RTL-Coding

RTL-Verification

Synthesis

Backend

Tape Out to Foundry to get end product a wafer with repeated number of identical Ics.

All modern digital designs start with a designer writing a hardware description of the IC (using HDL or Hardware Description Language) in Verilog/VHDL. A Verilog or VHDL program essentially describes the hardware (logic gates, Flip-Flops, counters etc) and the interconnect of the circuit blocks and the functionality. Various CAD tools are available to synthesize a circuit based on the HDL. The most widely used synthesis tools come from two CAD companies. Synposys and Cadence.

Without going into details, we can say that the VHDL, can be called as the "C" of the VLSI industry. VHDL stands for "VHSIC Hardware Definition Language", where VHSIC stands

for "Very High Speed Integrated Circuit". This language is used to design the circuits at a high-level, in two ways. It can either be a behavioural description, which describes what the circuit is supposed to do, or a structural description, which describes what the circuit is made of. There are other languages for describing circuits, such as Verilog, which work in a similar fashion.

Both forms of description are then used to generate a very low-level description that actually spells out how all this is to be fabricated on the silicon chips. This will result in the manufacture of the intended IC.

A typical analog design flow is as follows:

In case of analog design, the flow changes somewhat.

Specifications

Architecture

Circuit Design

SPICE Simulation

Layout

Parametric Extraction / Back Annotation

Final Design

Tape Out to foundry.

While digital design is highly automated now, very small portion of analog design can be automated. There is a hardware description language called AHDL but is not widely used as it does not accurately give us the behavioural model of the circuit because of the complexity of the effects of parasitic on the analog behaviour of the circuit. Many analog chips are what are termed as "flat" or non-hierarchical designs. This is true for small transistor count chips such as an operational amplifier, or a filter or a power management chip. For more complex analog chips such as data converters, the design is done at a transistor level, building up to a cell level, then a block level and then integrated at a chip level. Not many CAD tools are available for analog design even today and thus analog design remains a difficult art. SPICE remains

the most useful simulation tool for analog as well as digital design.

1.7.3 MOST OF TODAY'S VLSI DESIGNS ARE CLASSIFIED INTO THREE CATEGORIES:

1. Analog:
Small transistor count precision circuits such as Amplifiers, Data converters, filters, Phase Locked Loops, Sensors etc.

2. ASICS or Application Specific Integrated Circuits:
Progress in the fabrication of IC's has enabled us to create fast and powerful circuits in smaller and smaller devices. This also means that we can pack a lot more of functionality into the same area. The biggest application of this ability is found in the design of ASIC's. These are IC's that are created for specific purposes - each device is created to do a particular job, and do it well. The most common application area for this is DSP - signal filters, image compression, etc. To go to extremes, consider the fact that the digital wristwatch normally consists of a single IC doing all the time-keeping jobs as well as extra features like games, calendar, etc.

3. SoC or Systems on a chip:
These are highly complex mixed signal circuits (digital and analog all on the same chip). A network processor chip or a wireless radio chip is an example of an SoC

CHAPTER 2

Literature survey

It is indeed true FIR filter plays an major role in analog and digital applications. Below there are some articles on it.

J.Huang et al., had designed and implemented of a forty-order FIR filter for an IF GPS signal emulator using three algorithms: multiply and accumulate (MAC), add-and-shift strategy with CSD encoding (CSD), and novel common sub-expression approach (CSE) are shown in this study. Each method is examined item by item, including design and optimization procedures, in order to determine the best one with the least amount of hardware and power consumption. The FIR filter is written in Verilog HDL and implemented using a Virtex5 FPGA and Design Compiler based on SMIC 0.18um engineering and a Xilinx Virtex5 FPGA. In comparison to MAC and CSD, CSE consumes the least amount of complete occupied piece, with a 63 percent and 20% reduction in consumption, respectively. CSE in ASIC also shows a 66 percent and 13 percent decline in the entire bit nation, as well as a 36 percent decrease in the entire bit country [1].

Chia-Yu Yao et al., proposed a method for combining the design and execution of fixed-point FIR filter coefficients into a single design flow in this study. In an infinite spread outing sub-expression, the proposed technique constructs the fixed-point coefficients. The execution cost is calculated as good during the design phase and fed back to the design modus operandi so that the algorithm can repeatedly redesign the fixed-point coefficients. In many cases, design

examples indicate that we can produce better hardware-cost-effective FIR filters than the results given by other researchers [2].

SongYu et al. described the rules and construction of the FIR filter in this study in order to plan the FIR filter. Filter design and signal spectrum analysis tools in MATLAB were used to plan and analyse a 16-order FIR filter, as well as calculate the filter coefficients. Finally, Verilog HDL linguistic communication was utilized to code, and the Quartus II package was used to mimic. The simulation's result reveals that the effects of pulsation matching are incorporated into demand design [3].

Qiu-zhong Wu et al., presented an integrated computing machine-aided design (CAD) tool for ASIC execution of multiplier-less FIR digital filters with common sub-expression riddance (CSE) optimization in this paper. This tool combines the main mappings in the design flow of FIR filters for specific applications, such as coefficient computation and quantisation, common sub-expression optimisation, and HDL codification auto-generation. They developed an applied intermedial representation (IR) to express the circuit design resulting from the application of CSE approach, which is the key to merging CSE optimisation and HDL codification auto-generation. The use of this technique in the ASIC execution of multiplier less FIR filters can detect design mechanisation and drastically reduce design time [4].

CHAPTER 3

Design Methodology

3.1 FUNCTIONAL VERIFICATION AND LOGICAL SYNTHESIS

The above design is verified by performing the following steps

- RTL Simulation using Verilog-XL simulator
- Logic Synthesis using Synopsis Design Compiler
- Place and Route using Encounter
- Formality Equivalence checking

3.1.1 RTL SIMULATION

We must first verify the structural Verilog code before proceeding to any kind of circuit implementation. This is done using the Verilog – XL simulator. For this purpose, the command “Verilog <stimulus_file.v> <Verilog_file.v>” is executed in the terminal. This command checks the Verilog file for its correctness against the test bench file. The test bench file holds information about the inputs and displays the output from the simulated module in the terminal, for the given clock frequency

3.1.2 LOGIC SYNTHESIS USING SYNOPSIS DESIGN COMPILER

After the logical functionality of the Verilog code is verified, the second part of the design flow is to synthesize the RTL Verilog code into standard cells. A gate-level netlist file, which contains the interconnections, is obtained in this step. This is done using the Synopsis Design Compiler with the templates from OSU’s free standard cell library. To include the

correct Verilog file, the file template 'compile_dc.tcl' is modified before compiling. With this step, the top module for the Verilog file and also the correct clock frequency is also included.

Now we use the command 'dc_shell -f compile_dc.tcl' to synthesize the circuit and to convert the Verilog code into standard cells. This generates a gate-level netlist with the file extension '.vh'. The files 'timing.rep', 'cell.rep' and 'power.rep' are also generated in this step. These files hold information about the top module area, data arrival time and power consumed respectively.

3.1.3 Post Synthesis Simulation

The post synthesis simulation is performed using the command 'verilog gscl45nm.v <test_bench.v> <new_verilog.vh>'. This command includes the Verilog models of the standard cells present in 'gscl45nm.v'. Information about the number of cells and the total area of the module are stored in the file 'cell.rep'. Likewise, as the name suggests, the file 'timing.rep' contains information about the timing parameters such as the critical path, the delay in each path and the total delay in the design. In this module, the delay is measured from the parameter named 'data arrival time'.

PLACE & ROUTE USING ENCOUNTER

The next step in the design is to place and route all the interconnections of the standard cells. This is done using the layout in Cadence Encounter. For this step, we use the templates from OSU standard cell library to perform the place and route task. The necessary code for this task is predefined in the template file 'encounter.tcl'. The file name of the top level module needs to be changed for the variable 'my_toplevel' in the code.

The command 'encounter -init encounter.tcl' is now executed which runs the Encounter program and displays errors had they occurred. This step generates a Verilog file 'final.v' with all the interconnects. This file is same as that generated in the previous logic Synthesis step although it contains additional buffers to reduce the interconnect delay and to increase the clock robustness.

Post P&R Simulation

In the logic synthesis part, we had executed the gate-level netlist file ‘verilog_netlist.vh’. In the post place and route part the netlist file generated using encounter ‘final.v’ is used. The command used here is ‘verilog gscl45nm.v <testbench.v> final.v’. The values pertaining to the number of cells, area and timing are tabulated in table 5.2. By this, we can verify the design in RTL, logic synthesis and Place and route levels.

The area can be measured using the command ‘reportGatecount –limit 0’ where it displays the area required by the design. The delay and other time related parameters are obtained from the file ‘timing.rep.5.final’. This file contains 10 critical paths for the current design, each of which has information about their respective data arrival time. In the above table, only the worst case value is shown, which is the maximum delay.

3.1.4 FORMALITY EQUIVALENCE CHECKING

This is the final and last stage in the standard cell based ASIC design flow. The task in this step is to verify the functionality of the Verilog file ‘final.v’ against the Verilog file initially created in the RTL stage. This can be done by using the command ‘fsm_shell –gui’.

The RTL Verilog file is used as the reference here. The file ‘final.v’ which was generated by encounter is implemented here. The technology file ‘gscl45nm.v’ is also added to the implementation stage. After the ports are matched, the final verification is done.

Chapter 4

Existing System

Many of the FIR filters applications require of large order to meet the stringent frequency specifications. Very often these filters need to support high sampling rate for high-speed digital communication. The number of multiplications and additions required for each filter output, however, increases linearly with the filter order. Since there is no redundant computation available in the FIR filter algorithm, real-time implementation of a large order FIR filter in a resource constrained environment is a challenging task.

A digital filter uses a finite number of bits to describe signals and coefficients. Structures are required which can somehow retain the target filter specifications even after quantizing the coefficients. Conventional FIR filters require blocks such as adders, multipliers, and delay for constructing it. In the case of 4 Tap filter three adder blocks, three multiplier blocks and four filter coefficients are needed to construct it.

4.1 Block Diagram:

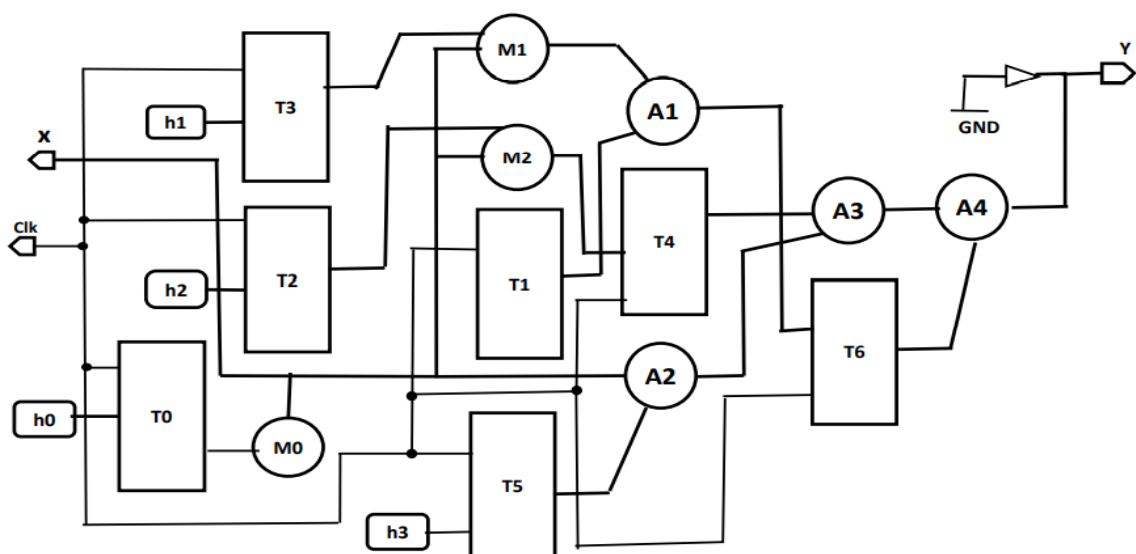


Fig 4.1 Systolic Transformation FIR Filter

- **4.2 Proposed System:**

- The systolic decomposition scheme provides a versatile choice of the look-up-tables (LUT) address length for DA based computation to decide on the correct area-time trade- off. One way systolic pipeline array is possible to reduce the memory size by using smaller address lengths for DA-based computing units but on the other hand this compromises on complexity and delay.
- In order to over come the above issues in conventional, systolic, associative and systolic associative method we are going to implement symmetric hybrid form of FIR filter. The block diagram is shown below.

As shown in Fig. below the critical data path in the proposed SHF FIR design contains of one multiplier and three adders. To further shorten the critical path for performance improvement, the corresponding addition order can be changed as that shown in below figure show the corresponding critical path before and after the addition order adjustment, respectively. It can be observed from the figure that, through the addition order adjustment, the critical data path in the proposed SHF can be

Block Diagram:

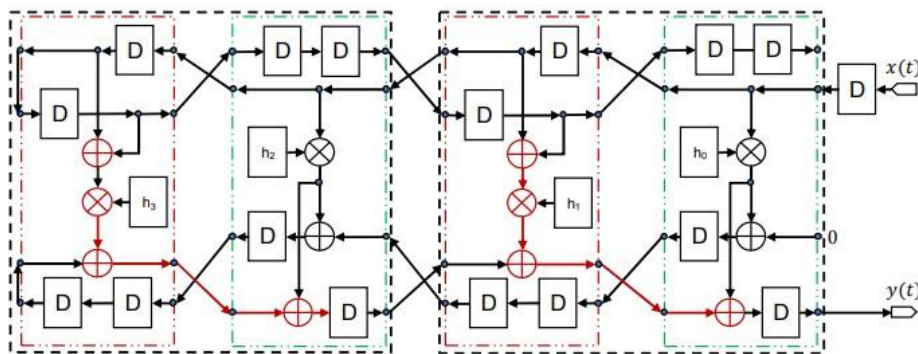


Fig 4.2

Critical Path optimisation

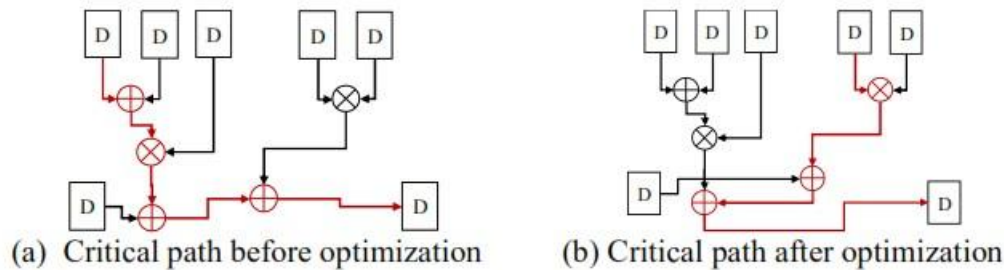


Fig 4.3

Languages used

- Verilog
- Xilinx

4.2 Xilinx

Xilinx

Xilinx, Inc. (/ˈzaɪlɪŋks/ *ZY-lings*) is an American technology company, primarily a supplier of programmable logic devices. It is known for inventing the field programmable gate array (FPGA) and as the first semiconductor company with a fabless manufacturing model. Founded in Silicon Valley in 1984, the company is headquartered in San Jose, California, with additional offices in Longmont, Colorado; Dublin, Ireland; Singapore; Hyderabad, India; Beijing, China; Shanghai, China; Brisbane, Australia and Tokyo, Japan.

Major FPGA product families include Virtex (high-performance), Kintex (mid-range) and Artix (low-cost), and the retired Spartan (low-cost) series. Major computer software includes Xilinx ISE and Vivado Design Suite.

Current family lines

Prior to 2010, Xilinx offered two main FPGA families: the high-performance Virtex series and the high-volume Spartan series, with a cheaper Easy Path option for ramping to volume production. The company also provides two CPLD lines: the Cool Runner and the 9500 series. Each model series has been released in multiple generations since its launch. With the introduction of its 28 nm FPGAs in June 2010, Xilinx replaced the high-volume Spartan family with the Kintex family and the low-cost Artix family.

In newer FPGA products, Xilinx minimizes total power consumption by the adoption of a high-K metal gate (HKMG) process, which allows for low static power consumption. At the 28 nm node, static power is a significant portion of the total power dissipation of a device and in some cases is the dominant factor. Through the use of a HKMG process, Xilinx has reduced power use while increasing logic capacity. Virtex-6 and Spartan-6 FPGA families are said to consume 50 percent less power, and have up to twice the logic capacity compared to the previous generation of Xilinx FPGAs. In June, 2010 Xilinx introduced the Xilinx 7 series: the Virtex-7, Kintex-7, and Artix-7 families, promising improvements in system power, performance, capacity, and price. These new FPGA families are manufactured using TSMC's 28 nm HKMG process. The 28 nm series 7 devices feature a 50 percent power reduction compared to the company's 40 nm devices and offer capacity of up to 2 million logic cells. Less than one year after announcing the 7 series 28 nm FPGAs, Xilinx shipped the world's first 28 nm FPGA device, the Kintex-7, making this the programmable industry's fastest product rollout. In March 2011, Xilinx introduced the Zynq-7000 family, which integrates a complete ARM Cortex-A9 MP Core processor-based system on a 28 nm FPGA for system architects and embedded software developers.

In Dec, 2013, Xilinx introduced the Ultra Scale series: Virtex Ultra Scale and Kintex UltraScale families. These new FPGA families are manufactured by TSMC in its 20 nm planar process. At the same time it announced an Ultra Scale SoC architecture in TSMC 16 nm Fin FET process.

Virtex family

The Virtex series of FPGAs have integrated features that include FIFO and ECC logic, DSP blocks, PCI-Express controllers, Ethernet MAC blocks, and high-speed transceivers. In

addition to FPGA logic, the Virtex series includes embedded fixed function hardware for commonly used functions such as multipliers, memories, serial transceivers and microprocessor cores. These capabilities are used in applications such as wired and wireless infrastructure equipment, advanced medical equipment, test and measurement, and defense systems.

Xilinx's most recently announced Virtex, the Virtex 7 family, is based on a 28 nm design and is reported to deliver a two-fold system performance improvement at 50 percent lower power compared to previous generation Virtex-6 devices. In addition, Virtex-7 doubles the memory bandwidth compared to previous generation Virtex FPGAs with 1866 Mbit/s memory interfacing performance and over two million logic cells. In 2011, Xilinx began shipping sample quantities of the Virtex-7 2000T "3D FPGA", which combines four smaller FPGAs into a single package by placing them on a special silicon interconnection pad (called an interposer) to deliver 6.8 billion transistors in a single large chip. The interposer provides 10,000 data pathways between the individual FPGAs – roughly 10 to 100 times more than would usually be available on a board – to create a single FPGA. In 2012, using the same 3D technology, Xilinx introduced initial shipments of their Virtex-7 H580T FPGA, a heterogeneous device, so called because it comprises two FPGA dies and one 8-channel 28Gbit/s transceiver die in the same package. The Virtex-6 family is built on a 40 nm process for compute-intensive electronic systems, and the company claims it consumes 15 percent less power and has 15 percent improved performance over competing 40 nm FPGAs.

The Virtex-5 LX and the LXT are intended for logic-intensive applications, and the Virtex-5 SXT is for DSP applications.^[75] With the Virtex-5, Xilinx changed the logic fabric from four-input LUTs to six-input LUTs. With the increasing complexity of combinational logic functions required by SoC designs, the percentage of combinational paths requiring multiple four-input LUTs had become a performance and routing bottleneck. The new six-input LUT represented a tradeoff between better handling of increasingly complex combinational functions, at the expense of a reduction in the absolute number of LUTs per device. The Virtex-5 series is a 65 nm design fabricated in 1.0 V, triple-oxide process technology.

Legacy Virtex devices (Virtex, Virtex-II, Virtex-II Pro, Virtex 4) are still available, but are not recommended for use in new designs.

Kintex

The Kintex-7 family is the first Xilinx mid-range FPGA family that the company claims delivers Virtex-6 family performance at less than half the price while consuming 50 percent less power. The Kintex family includes high-performance 12.5 Gbit/s or lower-cost optimized 6.5 Gbit/s serial connectivity, memory, and logic performance required for applications such as high volume 10G optical wired communication equipment, and provides a balance of signal processing performance, power consumption and cost to support the deployment of Long Term Evolution (LTE) wireless networks.

Artix

The Artix-7 family delivers 50 percent lower power and 35 percent lower cost compared to the Spartan-6 family and is based on the unified Virtex-series architecture. Xilinx claims that Artix-7 FPGAs deliver the performance required to address cost-sensitive, high-volume markets previously served by ASSPs, ASICs, and low-cost FPGAs. The Artix family is designed to address the small form factor and low-power performance requirements of battery-powered portable ultrasound equipment, commercial digital camera lens control, and military avionics and communications equipment.

Zynq

The Zynq-7000 family of SoCs addresses high-end embedded-system applications, such as video surveillance, automotive-driver assistance, next-generation wireless, and factory automation. Zynq-7000 integrate a complete Cortex-A9-processor-based 28 nm system. The Zynq architecture differs from previous marriages of programmable logic and embedded processors by moving from an FPGA-centric platform to a processor-centric model. For software developers, Zynq-7000 appear the same as a standard, fully featured ARM processor-based system-on-chip (SOC), booting immediately at power-up and capable of running a variety of operating systems independently of the programmable logic. In 2013, Xilinx introduced the Zynq-7100, which integrates digital signal processing (DSP) to meet emerging programmable systems integration requirements of wireless, broadcast, medical and military applications.^[78]

The new Zynq-7000 product family posed a key challenge for system designers, because Xilinx ISE design software had not been developed to handle the capacity and complexity of

designing with an FPGA with an ARM core. Xilinx's new Vivado Design Suite addressed this issue, because the software was developed for higher capacity FPGAs, and it included high level synthesis functionality that allows engineers to compile the co-processors from a C-based description. The AXIOM, the world's first digital cinema camera that is open source hardware, contains a Zynq-7000. a Zynq 7010 or 7020 is included in the adapteva parallella board.

Spartan family

The Spartan series targets applications with a low-power footprint, extreme cost sensitivity and high-volume; e.g. displays, set-top boxes, wireless routers and other applications. The Spartan-6 family is built on a 45-nanometer [nm], 9-metal layer, dual-oxide process technology. The Spartan-6 was marketed in 2009 as a low-cost solution for automotive, wireless communications, flat-panel display and video surveillance applications.

EasyPath

Because EasyPath devices are identical to the FPGAs that customers are already using, the parts can be produced faster and more reliably from the time they are ordered compared to similar competing programs. A 12-week time is guaranteed from receiving the design to mass production, and no redesign or re-qualification are required by the customer. In addition, customers are free to switch production from the non-programmable Easy Path FPGA to the original programmable Virtex FPGA if the need for design changes arises.

Corporate strategy and culture

Xilinx joined the Fortune ranks of the "100 Best Companies to Work For" in 2001 as No. 14, rose to No. 6 in 2002 and rose again to No. 4 in 2003.

In December 2008, GSA named Xilinx the Most Respected Public Semiconductor Company with \$500 million to \$10 billion in annual sales. The award recognizes excellence through success, vision and strategy in the industry.

Origin of the company name[

Having tried to register a number of more conventional company-names and having them rejected as already taken, the founders decided to deliberately create an unusual name in order to ensure that the name was accepted. Several legends explain the origin of the name.

Xilinx Fellow Bill Carter said that the name Xilinx was chosen because "The chemical symbol for silicon is Si. The 'X's at each end represent programmable logic blocks. The "linx" represents programmable links that connect the logic blocks together." Another explanation has it that two of the founders, Ross Freeman and Jim Barnett both attended the University of Illinois at Urbana–Champaign, also called the Fighting Illini. The name allegedly originated because the company founders were two ex-Illini, hence Xilinx (two 'X' Illini).

Ross Freeman

Ross Freeman, credited with the invention of the FPGA, was inducted into the National Inventor's Hall of Fame in 2009, more than 20 years after the invention of the FPGA. Starting in 1992, Xilinx has had an annual tradition of awarding a company employee with the Ross Freeman Award for Technical Innovation in Freeman's memory. Award honorees are based on nominations for innovations that resulted in significant tangible benefit to the company. Finalists are chosen by a committee and winners are selected based on a vote.

Competition

Today, Xilinx customers represent just over half of the entire programmable logic market, at 51%. Altera is Xilinx's strongest competitor with 34% of the market. Other key players in this market are Actel (now Microsemi) and Lattice Semiconductor.

4.3 Model Sim

ModelSim is a multi-language HDL simulation environment by Mentor Graphics, for simulation of hardware description languages such as VHDL, Verilog and SystemC, and includes a built-in C debugger. ModelSim can be used independently, or in conjunction with Altera Quartus or Xilinx ISE. Simulation is performed using the graphical user interface (GUI), or automatically using scripts.

Editions

ModelSim is offered in multiple editions, such as ModelSim PE, ModelSim SE, and ModelSim XE.

ModelSim SE offers high-performance and advanced debugging capabilities, while ModelSim PE is the entry-level simulator for hobbyists and students. ModelSim SE is used in

large multi-million gate designs, and is supported on Microsoft Windows and Linux, in 32-bit and 64-bit architectures.

ModelSim XE stands for *Xilinx Edition*, and is specially designed for integration with Xilinx ISE. ModelSim XE enables testing of HDL programs written for Xilinx Virtex/Spartan series FPGA's without needed physical hardware.

ModelSim can also be used with MATLAB/Simulink, using *Link for ModelSim*. *Link for ModelSim* is a fast bidirectional co-simulation interface between Simulink and ModelSim. For such designs, MATLAB provides a numerical simulation toolset, while ModelSim provides tools to verify the hardware implementation & timing characteristics of the design.

Language support

ModelSim uses a unified kernel for simulation of all supported languages, and the method of debugging embedded C code is the same as VHDL or Verilog.

ModelSim enables simulation, verification and debugging for the following languages:

VHDL

- Verilog
- Verilog 2001
- SystemVerilog
- PSL
- SystemC

References

- 1.R. Mahesh, A.P. Vinod, "A New Common Subexpression Elimination Algorithm for Realizing Low-Complexity High Order Digital Filters," IEEE Trans, on Computer-aided Design of Integrated Circuits and Systems. Vol.27, No.2, pp.217-229, February 2018.
- 2.H. Lo, H. Yoo, D. V. Anderson, "Hardware-efficient Distributed Arithmetic Architecture for High-order Digital Filters," ICASSP, Vol.5, pp.25-128, 2015.
- 3.W. Sen, T. Bin, Z. Jun, "Distributed Arithmetic for FIR Filter Design on FPGA," pp.620 - 623, ICCCAS2017.
4. Yao CY, Hsia WC, Ho YH. Designing hardware-efficient fixed-point FIR filters in an expanding sub expression space. IEEE Transactions Circuits Systems. 2014 Jan;
5. Johansson K, Gustafsson O, Wanhammar L. Bit-level optimization of shift-and-add based FIR filters. IEEE International Conference of Electronics Circuits Systems; 2019 Dec
6. Lou X, Yu YJ, Meher PK. Fine-grained critical path analysis and optimization for area-time efficient realization of multiple constant multiplications. IEEE Transactions Circuits Systems. 2015 Mar;
7. Ye WB, Yu YJ. Bit-level multiplierless FIR filter optimization incorporating sparse filter technique. IEEE Transactions Circuits Systems. 2014 Nov
8. Jinghao Ye, Masao Yanagisawa and Youhua A High-Performance Symmetric Hybrid Form Design for High-Order FIR Filters. 2020 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)

4.4 OUTPUT SCREEN SHOT

RTL Schamatic

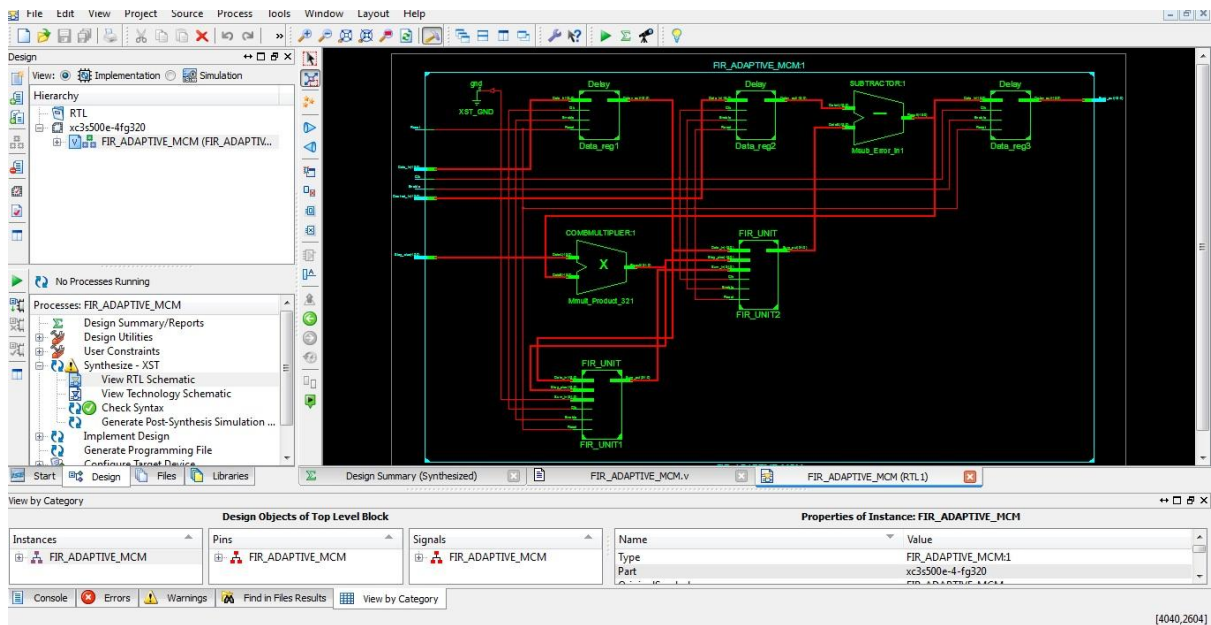


Fig 4.4

4.5 Power Consumption

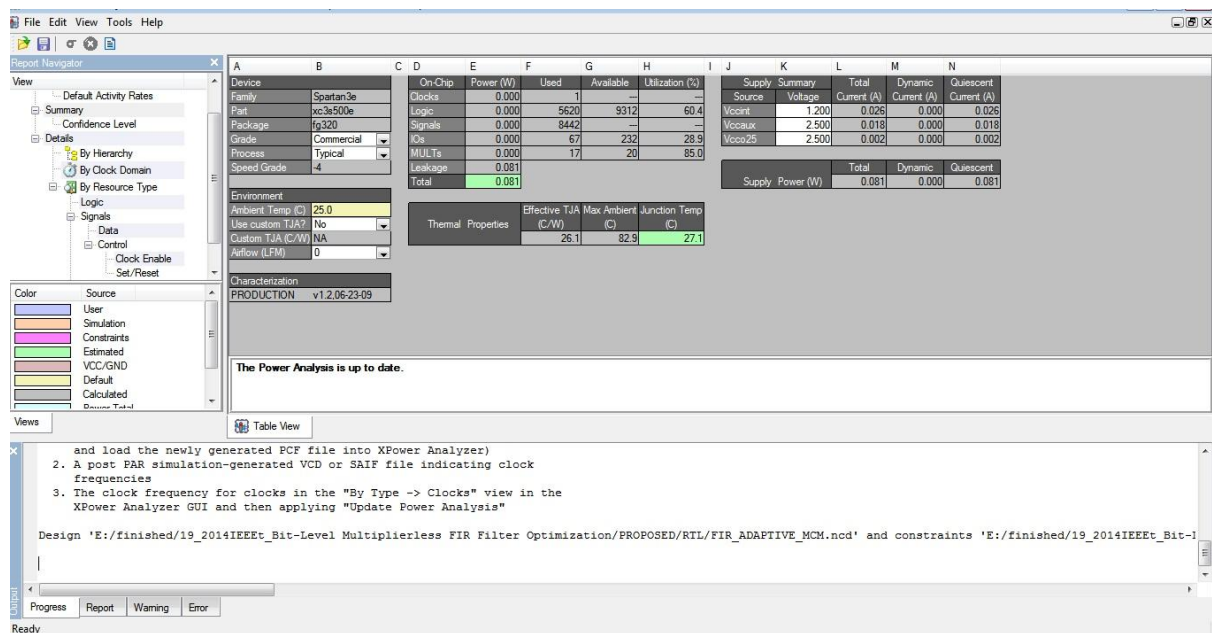


Fig 4.5

4.6 Simulation

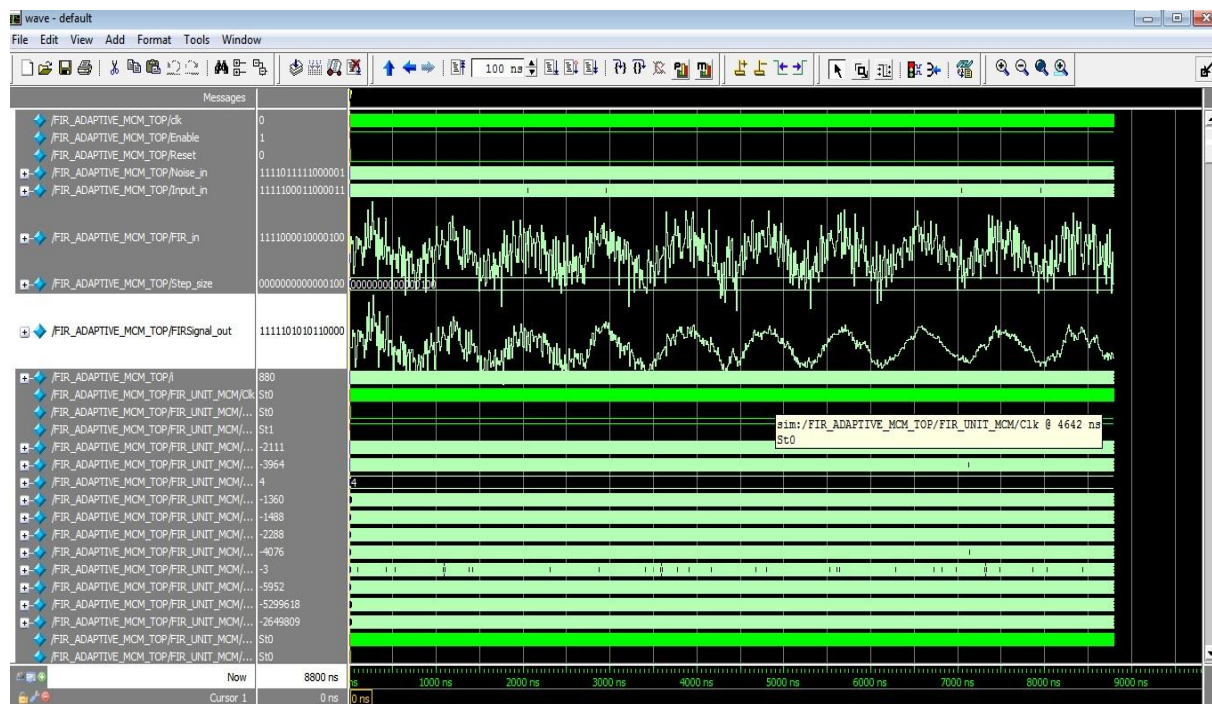


Fig 4.6

Performance Analysis

1. According to base paper

Filter Architecture	4 Tap			8 Tap			16 Tap		
	Power (W)	Hardware utilization	Delay (nS)	Power (W)	Hardware utilization	Delay (nS)	Power (W)	Hardware utilization	Delay (nS)
Conventional	0.223	Slice LUTs : 22 Slice Registers: 46 Bonded IOB : 23	16.034	0.266	Slice LUTs : 23 Slice Registers : 56 Bonded IOB : 23	26.045	0.458	Slice LUTs : 28 Slice Registers : 62 Bonded IOB : 25	30.356
Systolic	0.282	Slice LUTs : 22 Slice Registers : 25 Bonded IOB : 20	12.065	0.324	Slice LUTs : 23 Slice Registers : 28 Bonded IOB : 20	20.066	0.688	Slice LUTs : 26 Slice Registers : 32 Bonded IOB : 20	26.465
Associativity	0.227	Slice LUTs : 17 Slice Registers : 24 Bonded IOB : 20	6.088	0.255	Slice LUTs : 23 Slice Registers : 21 Bonded IOB : 20	10.106	0.496	Slice LUTs : 24 Slice Registers : 23 Bonded IOB : 20	18.015
Systolic with Associativity	0.127	Slice LUTs : 17 Slice Registers : 24 Bonded IOB : 20	5.998	0.240	Slice LUTs : 23 Slice Registers : 21 Bonded IOB : 20	9.885	0.486	Slice LUTs : 24 Slice Registers : 23 Bonded IOB : 20	16.185

Comparison:

	Power(W)	Delay(ns)	Slice
Existing Referred above	0.127	16.034	LUT: 22 Registers:46 IOB: 23
Proposed Attached below	0.081	5.26	LUT: 51 Registers:17 IOB: 51

Delay

```

Cell:in->out      fanout      Gate      Net      Logical Name (Net Name)
-----
LD:G->Q           1      0.472    0.379    result_15 (result_15)
LUT5:I3->O        1      0.097    0.279    Mmux_ArithOut371 (ArithOut_15_0BUF)
OBUF:I->O          1      0.000    0.000    ArithOut_15_0BUF (ArithOut<15>)
-----
Total              1.228ns (0.569ns logic, 0.659ns route)
                  (46.4% logic, 53.6% route)
=====
Timing constraint: Default path analysis
Total number of paths / destination ports: 832 / 16
-----
Delay:              4.046ns (Levels of Logic = 4)
Source:             B<7> (PAD)
Destination:        ArithOut<15> (PAD)

Data Path: B<7> to ArithOut<15>

Cell:in->out      fanout      Gate      Net      Logical Name (Net Name)
-----
IBUF:I->O          3      0.001    0.289    B_7_IBUF (B_7_IBUF)
DSP48E1:A7->P15    1      2.823    0.556    Mmult_A[7]_B[7]_MuLt_0_OUT (A[7]_B[7]_MuLt_0_OUT<15>)
LUT5:I1->O        1      0.097    0.279    Mmux_ArithOut371 (ArithOut_15_0BUF)
OBUF:I->O          1      0.000    0.000    ArithOut_15_0BUF (ArithOut<15>)
-----
Total              4.046ns (2.921ns logic, 1.125ns route)
                  (72.2% logic, 27.8% route)

```

Device Utilization

Device Utilization Summary (estimated values)				[1]
Logic Utilization	Used	Available	Utilization	
Number of Slice Registers	17	126800	0%	
Number of Slice LUTs	51	63400	0%	
Number of fully used LUT-FF pairs	17	51	33%	
Number of bonded IOBs	51	210	24%	
Number of DSP48E1s	1	240	0%	

Fig 4.7

Coding

module

FIR_ADAPTIVE_MCM(Clk,Reset,Enable,Data_in,Desired_in,Step_size>Error_out);

input Clk,Reset,Enable;

input signed [15:0] Data_in;

```

input  signed [15:0] Desired_in;

input  signed [15:0] Step_size;

output signed [15:0] Error_out;


wire signed [15:0] Error_in,Data_in_reg,Desired_in_reg,Product_16;

wire signed [31:0] Product_32,L_sum_out,L_sum_out1;

//initial Error_in= 0;

Delay Data_reg1  (.Clk(Clk),

                  .Reset(Reset),

                  .Enable(Enable),

                  .Data_in(Data_in),

                  .Delay_out(Data_in_reg)

                  );

Delay Data_reg2  (.Clk(Clk),

                  .Reset(Reset),

                  .Enable(Enable),

                  .Data_in(Desired_in),

                  .Delay_out(Desired_in_reg)

                  );

```

```
assign Product_32 = Step_size * Error_in;
```

```
assign Product_16 = Product_32[26:11];
```

```
FIR_UNIT  FIR_UNIT1(.Clk(Clk),  
  
                    .Reset(Reset),  
  
                    .Enable(Enable),  
  
                    .Data_in(Data_in_reg),  
  
                    .Step_size(Product_16),  
  
                    .Sum_in(32'h0000_0000),  
  
                    .Sum_out(L_sum_out1)  
  
                    );
```

```
FIR_UNIT  FIR_UNIT2(.Clk(Clk),  
  
                    .Reset(Reset),  
  
                    .Enable(Enable),  
  
                    .Data_in(Data_in_reg),  
  
                    .Step_size(Product_16),  
  
                    .Sum_in(L_sum_out1),
```

```

        .Sum_out(L_sum_out)

    );

    assign Error_in = Desired_in_reg - L_sum_out[26:11];

    Delay Data_reg3  (.Clk(Clk),

                      .Reset(Reset),

                      .Enable(Enable),

                      .Data_in(Error_in),

                      .Delay_out(Error_out)

    );

endmodule

module FIR_UNIT(Clk,Reset,Enable,Data_in,Step_size,Sum_in,Sum_out);

    input Clk,Reset,Enable;

    input  signed [15:0] Data_in;

    input  signed [15:0] Step_size;

    input  signed [31:0] Sum_in;

```

```

        wire                                signed                                [15:0]
        LMS_tap_1_delay_out,LMS_tap_2_delay_out,LMS_tap_3_delay_out,

        LMS_tap_4_delay_out,LMS_tap_5_delay_out,LMS_tap_6_delay_out,LMS
        _tap_8_delay_out,LMS_tap_7_delay_out;

        wire                                signed                                [31:0]
        LMS_tap_1_out,LMS_tap_2_out,LMS_tap_3_out,LMS_tap_4_out,

        LMS_tap_5_out,LMS_tap_6_out,LMS_tap_7_out,LMS_tap_8_out;

        FIR_UNIT_LOOP        FIR_UNIT_LOOP1    (.Clk(Clk),

                                .Reset(Reset),

                                .Enable(Enable),

                                .Data_in(Data_in),

                                .Step_size(Step_size),

                                .Delay_out(LMS_tap_1_delay_out),

                                .Tap_out(LMS_tap_1_out)

                                );

```

```

FIR_UNIT_LOOP FIR_UNIT_LOOP2  (.Clk(Clk),
                                .Reset(Reset),
                                .Enable(Enable),

                                .Data_in(LMS_tap_1_delay_out),

                                .Step_size(Step_size),

                                .Delay_out(LMS_tap_2_delay_out),

                                .Tap_out(LMS_tap_2_out)
                                );

```

```

FIR_UNIT_LOOP FIR_UNIT_LOOP3  (.Clk(Clk),
                                .Reset(Reset),
                                .Enable(Enable),

                                .Data_in(LMS_tap_2_delay_out),

                                .Step_size(Step_size),

                                .Delay_out(LMS_tap_3_delay_out),

                                .Tap_out(LMS_tap_3_out)
                                );

```

```

FIR_UNIT_LOOP FIR_UNIT_LOOP4  (.Clk(Clk),

```

```

        .Reset(Reset),

        .Enable(Enable),

    .Data_in(LMS_tap_3_delay_out),

        .Step_size(Step_size),

    .Delay_out(LMS_tap_4_delay_out),

        .Tap_out(LMS_tap_4_out)
    );

FIR_UNIT_LOOP FIR_UNIT_LOOP5 (.Clk(Clk),

        .Reset(Reset),

        .Enable(Enable),

    .Data_in(LMS_tap_4_delay_out),

        .Step_size(Step_size),

    .Delay_out(LMS_tap_5_delay_out),

        .Tap_out(LMS_tap_5_out)
    );

FIR_UNIT_LOOP FIR_UNIT_LOOP6 (.Clk(Clk),

        .Reset(Reset),

```

```

        .Enable(Enable),

    .Data_in(LMS_tap_5_delay_out),

        .Step_size(Step_size),

    .Delay_out(LMS_tap_6_delay_out),

        .Tap_out(LMS_tap_6_out)
    );

FIR_UNIT_LOOP FIR_UNIT_LOOP7 (.Clk(Clk),

    .Reset(Reset),

    .Enable(Enable),

    .Data_in(LMS_tap_6_delay_out),

        .Step_size(Step_size),

    .Delay_out(LMS_tap_7_delay_out),

        .Tap_out(LMS_tap_7_out)
    );

FIR_UNIT_LOOP FIR_UNIT_LOOP8 (.Clk(Clk),

    .Reset(Reset),

    .Enable(Enable),

```

```

        .Data_in(LMS_tap_7_delay_out),

                                .Step_size(Step_size),

        .Delay_out(LMS_tap_8_delay_out),

                                .Tap_out(LMS_tap_8_out)

                                );

    assign Sum_out = Sum_in + LMS_tap_1_out + LMS_tap_2_out +

                                LMS_tap_3_out    +    LMS_tap_4_out    +

LMS_tap_5_out +

                                LMS_tap_6_out    +    LMS_tap_7_out    +

LMS_tap_8_out;

endmodule

module
FIR_UNIT_LOOP(Clk,Reset,Enable,Data_in,Step_size,Delay_out,Tap_out
);

    input Clk,Reset,Enable;

    input signed [15:0] Data_in;

    input signed [15:0] Step_size;

```



```
.Enable(Enable),  
.Data_in(Weight_in),  
.Delay_out(Weight_out)  
);
```

```
assign Tap_out = Weight_out * Data_in;
```

```
endmodule
```

```
module Delay(Clk,Reset,Enable,Data_in,Delay_out);
```

```
input Clk,Reset,Enable;
```

```
input [15:0] Data_in;
```

```
output      [15:0] Delay_out;
```

```
reg[15:0] Delay_out;
```

```
always @(posedge Clk or posedge Reset) begin
```

```
    if (Reset ==1'b1) begin
```

```
        Delay_out <= 0;
```

```
    end
```

```
    else begin
```

```

        if (Enable == 1'b1) begin

            Delay_out <= Data_in;

        end

    end

end

endmodule

```

5 CONCLUSION

The report gives the implementation of various sorts of fixed-factor FIR clear out out architectures. The electricity intake has a crucial function in power saving and the growth in speed presents higher device performance. The essential performance parameters along with electricity, hardware utilization, and postpone in 4, 8, and sixteen faucet of numerous fixed-factor FIR clear out out architectures are simulated. The systolic structure with associativity transformation structure presents higher final results in contrast to the opposite architectures. The aforesaid structure gives higher consequences in phrases of electricity intake and postpone.