EFFICIENT 3-PARALLEL POLYPHASE ODD LENGTH FIR FILTER USING BRENT KUNG ADDER AND DADDA MULTIPLIER FOR VLSI APPLICATIONS

A MAJOR PROJECT REPORT

Submitted

By

Batch D-15

V. SIVA NAGAHANUSHA **[21485A0424]**

V. VINAY GURU CHARAN **[20481A04N7]**

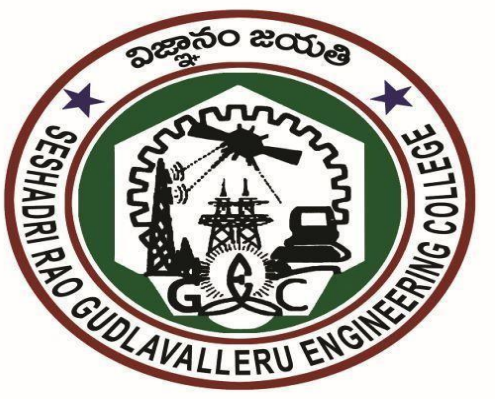
SK. DARIYA VALI **[20481A04L0]**

SYED MOHAMMAD AMEER **[20481A04M2]**

Under Supervision of

Ms. M. Deepika Krishna

Assistant Professor



Department of Electronics and Communication Engineering

SESHADRI RAO GUDLAVALLERU ENGINEERING COLLEGE

(An Autonomous Institute with Permanent Affiliation to JNTUK, Kakinada)

GUDLAVALLERU – 521356

ANDHRA PRADESH

SESHADRI RAO GUDLAVALLERU ENGINEERING COLLEGE
(An Autonomous Institute with Permanent Affiliation to JNTUK)
SESHADRI RAO KNOWLEDGE VILLAGE GUDLAVALLERU-521356
DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that the project report entitled "**EFFICIENT 3-PARALLEL POLYPHASE ODD LENGTH FIR FILTER USING BRENT KUNG ADDER AND DADDA MULTIPLIER FOR VLSI APPLICATIONS** "is a bonafide record of work carried by **V. SIVA NAGA HANUSHA** (21485A0424), **V. VINAY GURU CHARAN** (20481A04N7), **SK. DARIYA VALI** (20481A04L0), **SYED MOHAMMAD AMEER** (20481A04M2) under my guidance and supervision in partial fulfillment of the requirements for award of the degree of Bachelor Of Technology in Electronics and Communication Engineering of Seshadri Rao Gudlavalleru Engineering College, Gudlavalleru**.**

(Ms. M. Deepika Krishna)
PROJECT GUIDE

(Dr. B. Rajasekhar)
HEAD OF THE DEPARTMENT

ACKNOWLEDGEMENT

We are glad to express our deep sense of gratitude to **Ms. M. Deepika Krishna** Assistant Professor, Electronics and Communication Engineering for her guidance and cooperation in completing this project. Through this we want to convey our sincere thanks to her for inspiring assistance during our project. We express our heartfelt gratitude and deep in debtedness to our beloved Head of the Department **Dr. B. RAJASEKHAR**, for his great help and encouragement in doing our project successfully. We also express our gratitude to our principal **Dr. B. Karuna Kumar** for his encouragement and facilities provide during the course of the project.

We express our heartfelt gratitude to our **Faculty members** and **Lab Technicians** for providing great support for us in completing our project. We think one and all who have rendered help us to directly or indirectly, in the completion of this work.

V. Siva Naga Hanusha-21485A0424
V. Vinay Guru Charan-20481A04N7
SK. Dariya Vali-20481A04L0
Syed Md. Ameer-20481A04M2

CONTENTS

LIST OF FIGURES

ABSTRACT

The main requirements for the design and implementation of DSP processors are less area and minimum power consumption. A parallel Finite Impulse Response (FIR) filter reduces the hardware complexity of higher order FIR filter and serves as core in designing a processor. In this work, we will use an improved adder and multiplier in conjunction with a Fast FIR Algorithm (FFA) based 3-parallel polyphase FIR filter in place of a conventional adder and multiplier. Consequently, a three-parallel polyphase odd-length FIR filter based on FFA will be developed, employing improved adders such as—Brent Kung, Carry lookahead, and ripple carry, as well as two multipliers—the Vedic and Booth multipliers. Low power consumption and delay reduction in the Booth multiplier as well as Brent Kung adder make it highly preferable for designing of the parallel FIR filter for low power and small chip area VLSI applications. Three-parallel polyphase FIR filters based on FFA can be further developed using various multipliers and adders. The proposed filters will be implemented using Xilinx.

KEYWORDS:

- VLSI
- Digital filters
- Finite Impulse Response Filter
- Brent-Kung Adder
- Dadda Multiplier
- Verilog HDL

Chapter 1
INTRODUCTION

1.1 Background:

In the realm of Very Large-Scale Integration (VLSI) technology, digital filters stand as indispensable components, providing essential signal processing capabilities. Among these, the Finite Impulse Response (FIR) Filter holds particular significance, owing to its widespread use and critical role in numerous applications. Central to the performance of FIR filters is the speed of their multiplier units, which directly impacts their effectiveness in processing digital signals with minimal delay. Enhancing the efficiency of FIR filters has thus become a key focus within the VLSI community, prompting innovative approaches to address latency and power consumption concerns.

The Dadda multiplier presents itself as a potential remedy for the need for enhanced FIR filter performance. Significant benefits over conventional designs are provided by this innovative multiplier architecture, particularly in terms of power efficiency and latency reduction. Multiplication operations may be performed more quickly by taking use of the Dadda structure's intrinsic parallelism, which helps to reduce the bottleneck that multiplier units in FIR filters are known for. With minimal latency and power conservation being critical factors in VLSI applications, this breakthrough marks a major advancement in digital filter design optimization. The Dadda multiplier's practical benefits may be realized through its implementation, which is made easier by Xilinx tools and Verilog Hardware Description Language (HDL).

Engineers may easily create effective FIR filter designs with Verilog HDL's strong modelling and simulation framework for complicated digital systems. Additionally, the implementation workflow is streamlined by interaction with Xilinx tools, which provide thorough assistance for operations related to synthesis, place-and-route, and verification. Engineers may use the Dadda multiplier to significantly increase the performance of FIR filters, especially in low voltage and low-power VLSI applications, by using this well-organized development environment.

A major development in VLSI technology, the Dadda multiplier's usage in FIR filter design offers improved performance and efficiency for a variety of applications. This novel technique tackles important issues facing digital filter implementations in contemporary VLSI systems by lowering latency and power consumption while retaining high operating frequencies. Engineers may achieve noticeable gains in FIR filter designs by utilizing the combination of Verilog HDL, Xilinx tools, and Dadda multiplier architecture. This will open the door for more resilient and energy-efficient VLSI systems in the digital era.

1.2 Aim of the project:

The main aim of this project is to implement FIR filter using Dadda multiplier to enhance speed in multiplication. It emphasizes low power consumption and high speed in FIR filter styles, acknowledging the trade-off between hardware complexity and space.

- The primary goal of the work is to use a Dadda multiplier architecture to create a Finite Impulse Response filter. The purpose of this architecture is to increase the FIR filter's multiplication operations' speed in order to lower latency and boost overall performance.
- Achieving low power consumption in the FIR filter design is a key focus of the research. The goal is to save energy without sacrificing performance by using the Wallace tree multiplier, which has intrinsic power efficiency advantages over conventional multiplier designs.
- A comparison study will be carried out in order to assess the effectiveness and performance of the suggested system. In this investigation, the Dadda multiplier-implemented FIR filter's performance metrics such as latency, power consumption, and resource utilization will be compared to those of a conventional multiplier-based FIR filter.

1.3 Methodology:

The technical method used in this project involves applying principles of digital signal processing (DSP) and hardware description language (HDL) programming, specifically Verilog, to implement a digital filter with efficient multiplication using the Dadda multiplier and Brent-Kung Adder.

1. Digital Signal Processing (DSP): This involves understanding the theory and principles behind Digital filters, including finite impulse response (FIR) or infinite impulse response (IIR) filter design, filter structures, and algorithms for efficient implementation.

2. Verilog Programming: Verilog is a hardware description language used for modeling and designing digital systems. In this project, Verilog is utilized to describe the behavior and structure of the digital filter, including its various components such as multipliers, adders, accumulators, and control logic.

Sub methodology:

Dadda Multiplier: The **Dadda multiplier** is a hardware binary multiplier design invented by computer scientist Luigi Dadda in 1965.[1] It uses a selection of full and half adders to sum the partial products in stages (the **Dadda tree** or **Dadda reduction**) until two numbers are left. The design is similar to the Wallace multiplier, but the different reduction tree reduces the required number of gates (for all but the smallest operand sizes) and makes it slightly faster (for all operand sizes).

Brent-Kung Adder: The Brent–Kung adder is a parallel prefix adder (PPA) form of carry-lookahead adder (CLA). Proposed by Richard Peirce Brent and Hsiang Te Kung in 1982 it introduced higher regularity to the adder structure and has less wiring congestion leading to better performance and less necessary chip area

to implement compared to the Kogge–Stone adder (KSA). It is also much quicker than ripple-carry adders (RCA).

1.4 Significance of this Work:

FIR filter design plays a vital role in digital signal processing (DSP) for manipulating signals in the frequency domain. The significance of this work is

Versatility: FIR filters can be designed to achieve various frequency responses, allowing you to pass specific frequency bands (low-pass, high-pass, band-pass) while attenuating unwanted ones.

Stability: FIR filters are inherently stable, meaning their output won't grow uncontrollably, unlike some other filter types.

Linear Phase: FIR filters can achieve linear phase response, ensuring no distortion or delay variation across certain frequencies. This is crucial for applications like audio processing.

Ease of Implementation: FIR filters are relatively simple to implement in hardware or software due to their finite impulse response.

1.5 Outline of this Report:

To implement a digital filter using Verilog, you first need to clearly define the filter's specifications, including its type (FIR or IIR), order, cutoff frequency, and other requirements. Once you have a clear understanding of the filter's specifications, you can select the appropriate algorithm, such as FIR or IIR, based on the design requirements. The next step involves writing Verilog code to implement the selected algorithm, including defining the filter's structure with registers, adders, multipliers, and other necessary components. After writing the Verilog code, you need to simulate it using tools like ModelSim to verify its functionality and correctness.

1. Introduction to FIR Filters: Briefly introduce FIR filters, explaining their significance in digital signal processing and their application in various fields such as communications, audio processing, and image processing.

2. Verilog Overview: Provide an overview of Verilog, a hardware description language used for modeling electronic systems. Explain its role in digital design and its syntax for describing digital circuits.

3. FIR Filter Basics: Dive deeper into the fundamentals of FIR filters, covering their structure, mathematical representation, and the concept of finite impulse response.

4. Filter Design Specifications: Discuss the specifications and requirements of the FIR filter to be implemented. This includes parameters such as filter order, cutoff frequency, passband ripple, and stopband attenuation.

5. Verilog Implementation: Detail the process of implementing the FIR filter in Verilog. This involves translating the mathematical description of the filter into Verilog code, specifying the structure of the filter, and describing its functionality.
6. Simulation and Verification: Explain how simulation tools such as ModelSim or Vivado are used to simulate the Verilog code and verify the functionality of the FIR filter design. Discuss testbench design and verification techniques.
7. Performance Evaluation: Evaluate the performance of the implemented FIR filter in terms of its frequency response, filter characteristics, and computational efficiency. Compare the results with the design specifications.

1.6 Conclusion:

In conclusion, the implementation of a Finite Impulse Response (FIR) filter using Verilog offers a robust and efficient solution for digital signal processing applications. By leveraging Verilog's capabilities in software design, we have effectively translated the theoretical underpinnings of FIR filters into executable code, allowing for efficient signal processing tasks in various domains. Through meticulous simulation and verification processes, we have validated the functionality and accuracy of the FIR filter design, ensuring its suitability for real-world applications.

CHAPTER 2
LITERATURE REVIEW

2.1 Literature Survey:

[A. R. Kumar at. al.(2023)] In this article, a reconfigurable FIR filter capable of covering the most common wireless communication standards on software-defined radio applications was implemented using a low-cost field programmable gate array, Proposal of a reconfigurable FIR filter for SDR applications Implemented using a low-cost FPGA with 22% utilization [1].
[V. Thamizharasan and N. Kasthuri, at al (2023)] In this paper, a hybrid FIR filter is proposed to improve the speed of the signal processing system using hybrid adder and hybrid multiplier, and the result shows that the delay of FIR filter using the proposed multiplier is improved by 26.51%, 15.59, 15.83% and 2.79% as compared with CLA, conventional CSA, CSA-BK-BEC and the proposed adder with array multiplier respectively [2].
[N. Chabini and R. Beguenane at al (2022)] In this article, a divide-and-conquer approach is proposed to synthesize digital FIR filters with small coefficients and large data input on modern Field Programmable Gate Arrays (FPGAs)[3].
[VJamuna at al (2018)] the paper mentions that the FIR filter is implemented in Spartan-3E FPGA kit using Xilinx ISE simulator for the analysis of the designed architecture[4].
[N. Udaya Kumar, at al(2018)] The paper mentions that the design of the FIR filter using different MAC techniques is implemented with Xilinx ISE 12.2 software on Spartan 3E kit[5].

2.2 Early Developments:

Using FPGA platforms, A. R. Kumar, B. Sriraj, K. H. L. P. Prasad, and P. R. Rajasri offer a new method for building a systolic finite impulse response (FIR) filter[1] The authors' main goal is to achieve effective filtering operations in FPGA settings by utilizing the single-channel approach. They seek to improve FIR filtering jobs' performance and throughput by utilizing the systolic architectures' inherent parallelism and pipelining capabilities. The design factors, implementation techniques, and experimental findings from their FPGA-based FIR filter prototype are covered in this study. The authors show the efficacy and applicability of their suggested technique for practical applications in cloud computing, data science, and engineering via thorough examination and analysis.

In this article, N. Chabini and R. Beguenane investigate the design and implementation of digital Finite Impulse Response (FIR) filters based on FPGAs that are optimized for situations involving large data input and small coefficients. They also discuss the difficulties these scenarios present and suggest FPGA-specific solutions[3]. The authors explore the complexities of fine-tuning FIR filter designs to effectively handle tiny coefficient values while preserving reliable performance with massive input data streams. Their study provides important insights into the design of FIR filters in particular, which is relevant to the development of FPGAbased digital signal processing systems. The study's conclusions and methodology, which highlight developments in FPGA-based digital signal processing techniques, are applicable to a wide range of computer and communication systems where FIR filtering is used.

In the field of Very Large Scale Integration (VLSI), N. Udaya Kumar, U. Subbalakshmi, B. Surya Priya, and K. Bala Sindhuri explore the complexities of implementing Finite Impulse Response (FIR) filters using different addition and multiplication techniques[5]. They provide a thorough examination of the various

approaches used in the VLSI domain to efficiently realize FIR filters. The authors want to find the best methods by examining several addition and multiplication strategies that strike a balance between efficiency, area use, and power usage. Their work sheds light on the trade-offs involved in selecting particular arithmetic operations in VLSI applications, which is helpful in understanding the design and implementation of FIR filters.

The practical aspects of implementing Finite Impulse Response (FIR) filters using Field-Programmable Gate Arrays (FPGAs) and Verilog Hardware Description Language (HDL) are explored by L. Y. Akshitha V. Ramesh, Apeksha Ravi Kumar, and Amulya S. Iyengar[7]. They offer insights into the process of designing and realizing FIR filters on FPGA platforms, utilizing the capabilities of FPGA for hardware implementation and Verilog HDL for digital circuit modelling. In order to accomplish effective and high-performance filtering operations, the authors examine numerous design considerations, approaches, and optimisation strategies used in the construction of FIR filters. Through the integration of theoretical concepts with real-world implementation specifics, the study provides insightful advice for academics and engineers working on digital signal processing and FPGAbased system design.

The creation of Finite Impulse Response (FIR) filters implemented on Field-Programmable Gate Arrays (FPGAs) is the subject of study by S. Chao, Q. Hui, S. Tong, M. Junzhi, Z. Yongjie, and D. Jianjun. With an emphasis on performance, resource efficiency, and power economy, the authors explore the complexities of creating FIR filters especially for FPGAbased implementations[9]. The goal of the authors' use of FPGA technology is to provide effective and real-time signal processing for a variety of applications in automation control, networking, electronics, and information technology. With its insightful descriptions of design techniques, implementation approaches, and experimental outcomes from their FPGAbased FIR filter design, the work makes a significant addition to the fields of digital signal processing and FPGA-based system design.

CHAPTER 3

WORK TITLE EXPLANATION

3.1 Flow Chart

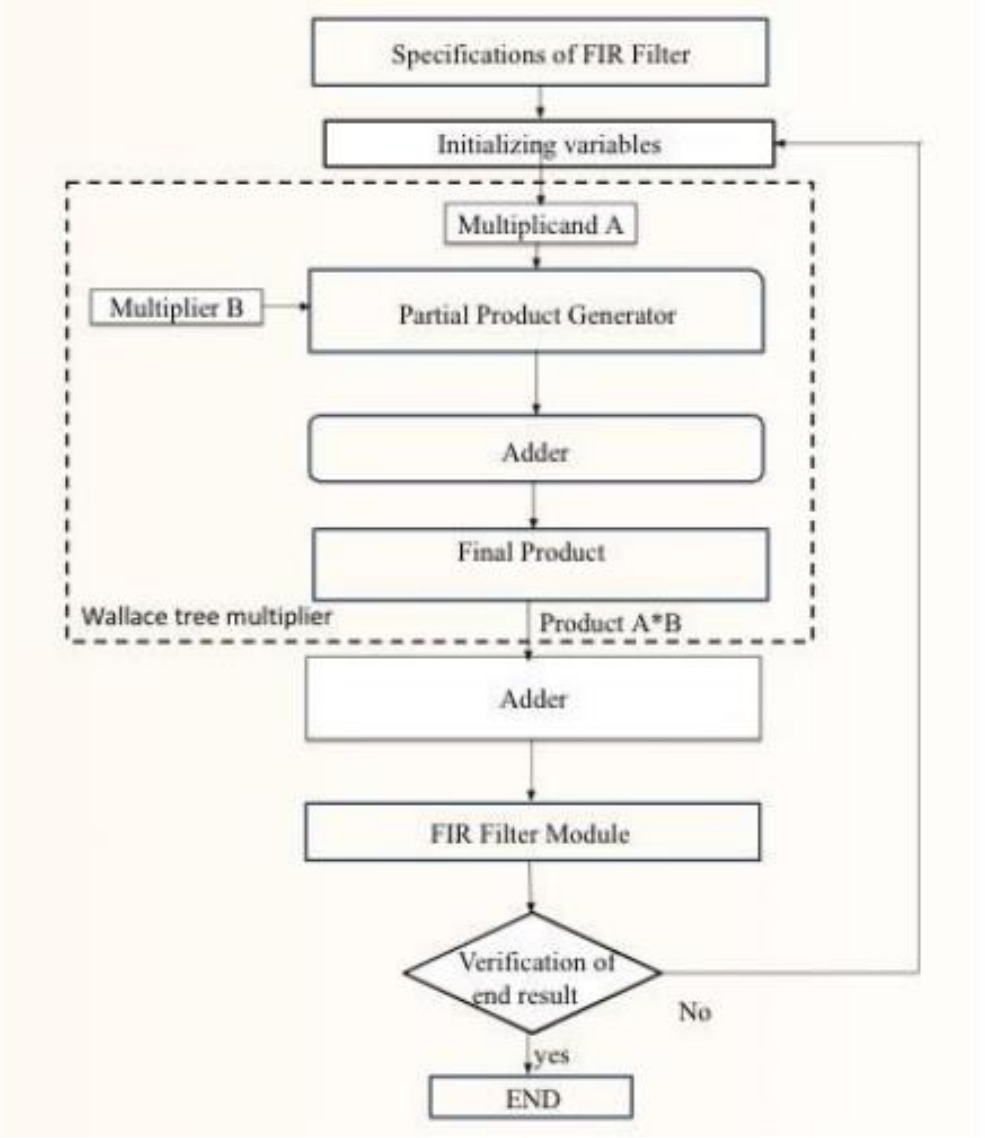


Fig.3.1: Flow chat of FIR Filter Design

3.1.1 Digital Filter:

An example of a digital filter is a limited Impulse Response filter, which has an impulse response with a limited period. Each output sample in a FIR filter is the weighted sum of its input samples, with the weights being established by the filter coefficients. FIR filters are used for applications requiring accurate signal processing because they have a linear phase response and are intrinsically stable, in

contrast to infinite impulse response (IIR) filters. FIR filters are widely used in image processing, biological signal analysis, telecommunications, and audio processing. Because of their finite impulse response feature, which makes them easily implementable with methods like convolution, they are useful tools for a variety of digital signal processing applications. FIR filters provide fine-grained control over the features of frequency response. To satisfy the demands of various signal processing jobs, engineers can create FIR filters with certain frequency response profiles, such as low-pass, high-pass, band-pass, or band-stop. Furthermore, FIR filters maintain the phase connections in the input signal by offering linear phase response throughout the whole frequency range. This feature is very helpful in applications like audio and picture processing where phase distortion must be reduced to the lowest possible level.

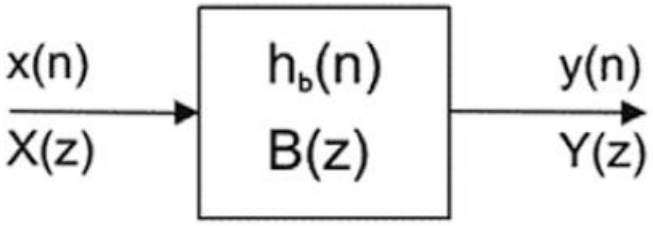


Fig 3.1.1: FIR Filter

1. Design steps for FIR filter:

• Define filter specifications.

• Specify the window functions according to the filter specifications.

• Compute the filter order required for a given set of specifications.

• Compute the window function coefficients.

• Compute the ideal filter coefficients according to the filter order.

• Compute FIR filter coefficients according to the obtained window function and ideal filter coefficients.

• If the resulting filter has too wide or too narrow transition region, it is necessary to change the filter order by increasing or decreasing it according to needs, and after that steps 4, 5 and 6 are iterated as many times as needed.

3.1.2: 4-Tap FIR Filter :

The data flow diagram for N-tap FIR filters. Triangle shape indicates multiplier. The incoming x[n] is multiplied with b0, b1, b2, b3, etc. which are the coefficients. Encircled plus sign symbol shows adders which are used to accumulate the convolution of an x[n] where n = 0,1,2,3 etc…..

Nth order equation of a FIR Filter is given by,

$y(n) = b0x(n) + b1x(n-1) + b2x(n-2) + \cdots + bNx(n-N)$

It can also be expressed as a convolution of the co-efficient sequence bi with the input signal i.e., transfer function of the linear invariant (LTI) FIR filter can be expressed as the following equation:

$$y(n) = \sum_{k=0}^{N-1} bix[n-i]$$

Where, N is the length of the filter, bi is the ith coefficient, and x(n−i) is the input data at time instant (n−i) .

The z transform of the data output is

$Y(z) = H(z).X(z)$

Where, H (z) is the transfer function of the filter, given by

$$H(z) = \sum_{k=0}^{N} bi.z^{-1}$$

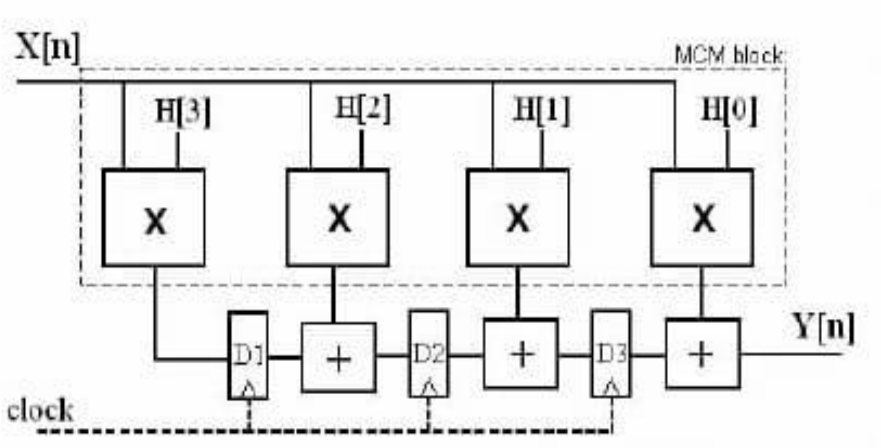


Fig 3.1.2: 4-Tap FIR Filter

3.1.3: DADDA MULTIPLIER:

Dadda multipliers are a similar alternative which don't aim to cover the maximum number of bits at a single step, so they might leave some possibility uncovered

- It's aim is to preserve the same number of steps as the Wallance tree multiplier while reducing the number of full adders and half adders used.
- The algorithm in determining the number of half adders and full adders is a lot more complicated
- It's a lot more complicated to determine how to perform coverage

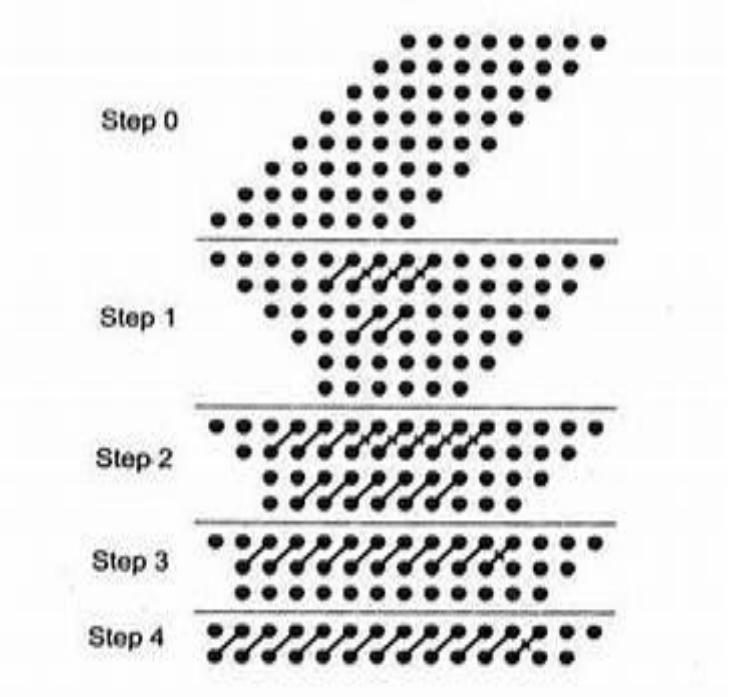


Fig.3.1.3: 8-Bit Dadda multiplier

The Dadda multiplier is a technique used in digital circuit design for multiplying two binary numbers. It's named after Luigi Dadda, an Italian computer scientist who introduced it in the 1960s. The main idea behind the Dadda multiplier is to perform partial products generation and reduction using a parallel prefix tree structure, also known as a carry-save adder tree.

Here's a simplified overview of how it works:

1. **Partial Products Generation**: Multiply each bit of one number (multiplicand) with each bit of the other number (multiplier) to generate partial products. This step results in a matrix of partial products.
2. **Partial Products Reduction**: Group the partial products into groups of three bits each and perform a parallel reduction operation. This reduces the number of bits by two for each group.
3. **Carry Propagation**: Carry bits are propagated through the tree structure to obtain the final result.

The advantage of the Dadda multiplier is its efficient use of hardware, as it minimizes the number of logic gates required compared to traditional multiplication algorithms like the array multiplier. This makes it suitable for applications where hardware resources are limited or where high-speed multiplication is required.

Overall, the Dadda multiplier is a significant technique in digital design for efficient multiplication of binary numbers.

3.1.4 Brent-Kung Adder

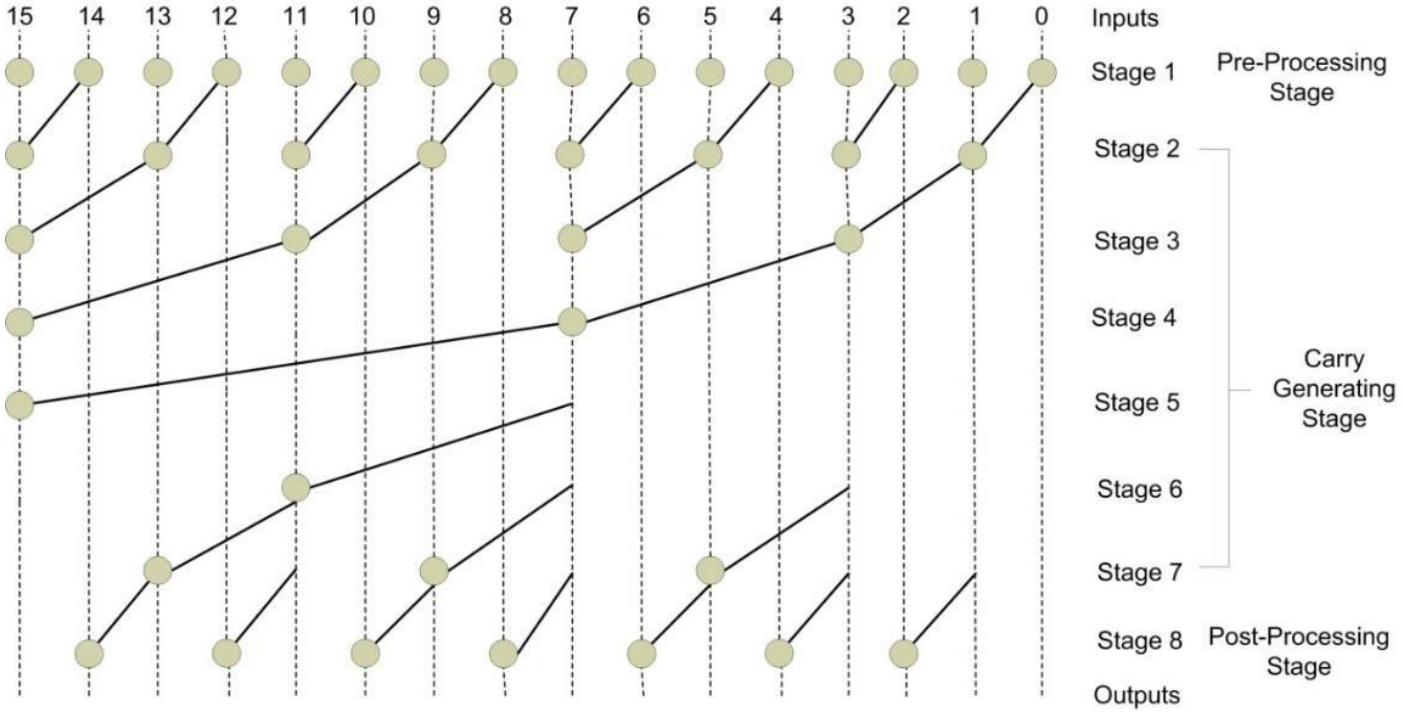


Fig 3.1.4: 16-Bit Brent-Kung Adder

The Brent-Kung adder is a type of parallel-prefix adder, which is designed to perform fast addition of binary numbers. It was introduced by Robert Brent and David Kung in their paper published in 1982.

Parallel-prefix adders are based on the idea of breaking down the carry propagation into smaller stages and computing them in parallel. This allows for faster addition compared to traditional ripple-carry adders, especially for large word sizes.

Here's a simplified overview of how the Brent-Kung adder works:

1. **Prefix Computation**: The input bits are fed into a series of prefix computation stages. Each stage computes partial prefix sums of the input bits.
2. **Carry Generation**: The carry bits are generated in parallel at each stage based on the prefix sums computed in the previous stage.
3. **Carry Propagation**: The generated carry bits are propagated to the next stage, where they are combined with the prefix sums to compute new prefix sums and carry bits.
4. **Final Sum Generation**: Once all the carry bits have been computed, the final sum bits are generated by combining the input bits with the carry bits using XOR gates.

The key advantage of the Brent-Kung adder is its efficient use of hardware resources and its ability to perform addition in parallel, leading to improved performance for large word sizes. Overall, the Brent-Kung adder is an important advancement in digital circuit design, providing a faster alternative to traditional ripple-carry adders for addition operations.

CHAPTER 4

SOFTWARE IMPLEMENTATION

4.1 Software implementation details

4.1.1 Xilinx version 14.7:

Xilinx ISE 14.7 was a comprehensive design suite for FPGA design and implementation, offering tools for design entry, synthesis, implementation, verification, and programming. However, it has been largely superseded by Vivado Design Suite, which provides enhanced features, performance, and support for the latest FPGA technologies.

1. **Design Flow:** Xilinx ISE 14.7 provided a comprehensive design flow for FPGA design, encompassing tasks such as design entry, synthesis, implementation, and verification. It supported various design entry methods including schematic capture, Hardware Description Languages (HDLs) such as VHDL and Verilog, as well as higher-level design entry tools like System Generator for DSP.
2. **Synthesis and Optimization:** The suite offered synthesis tools that transformed RTL (Register Transfer Level) code into a gate-level netlist, optimizing for area, speed, or power depending on user preferences. It provided optimizations such as technology mapping, logic restructuring, and timing-driven synthesis to meet design constraints.
3. **Implementation:** Xilinx ISE 14.7 facilitated the process of placing and routing the synthesized design onto the target FPGA device. It included algorithms for automatic placement and routing, floor planning, and physical optimization to achieve timing closure and meet performance goals.
4. **Verification and Simulation:** The suite supported simulation of the design using Xilinx's ISim simulator or third-party simulators. It allowed users to verify functional correctness, debug designs, and analyze timing behavior before synthesis and implementation.
5. **Device Support:** Xilinx ISE 14.7 supported a wide range of FPGA families from Xilinx, including Spartan, Virtex, and Cool Runner devices. It provided device-specific constraints and optimizations tailored to each FPGA family.
6. **Programming and Configuration:** The suite included utilities for programming the configured bitstream onto the target FPGA device. It supported various programming methods such as JTAG, SelectMAP, and PROM programming for configuration memory devices.
7. **Legacy Status:** Xilinx ISE 14.7 is considered a legacy tool, as Xilinx has transitioned to Vivado

Design Suite as its primary development environment for FPGA design. While still functional, users are encouraged to migrate to Vivado for new designs and take advantage of its improved capabilities and support for newer FPGA families.

CHAPTER 5

RESULTS AND DISCUSSION

A Finite Impulse Response filter result's "black box" depiction usually captures the filter's operation without disclosing any of its internal workings. It acts as an abstraction so that users may only interact with the filter based on its input-output behavior, without having to comprehend the internal workings or specifics of its implementation. The input signal and the filtered output signal are represented by the input and output ports of the FIR filter, which are shown in this illustration as a functional block. An eight-tap Finite Impulse Response (FIR) filter is represented by the "fir8t4" module. The process generates a 16-bit output signal y expressed as y[15:0] from an 8-bit input signal x represented as x[7:0]. Clock and reset inputs are also included in the module to regulate the timing and startup of the filter operation. The module encapsulates the functionality of the FIR filter as a black box by abstracting its coefficients and internal structure. messages helped in enhancing ease of use and usability.

5.1 Top view of FIR filter:

The implementation specifics of the "fir8t4" moduleestablish the precise coefficients and filtering properties of the filter, which creates the filtered output signal ‘y’ by applying a convolution operation between the input signal ‘x’ and its internal coefficients

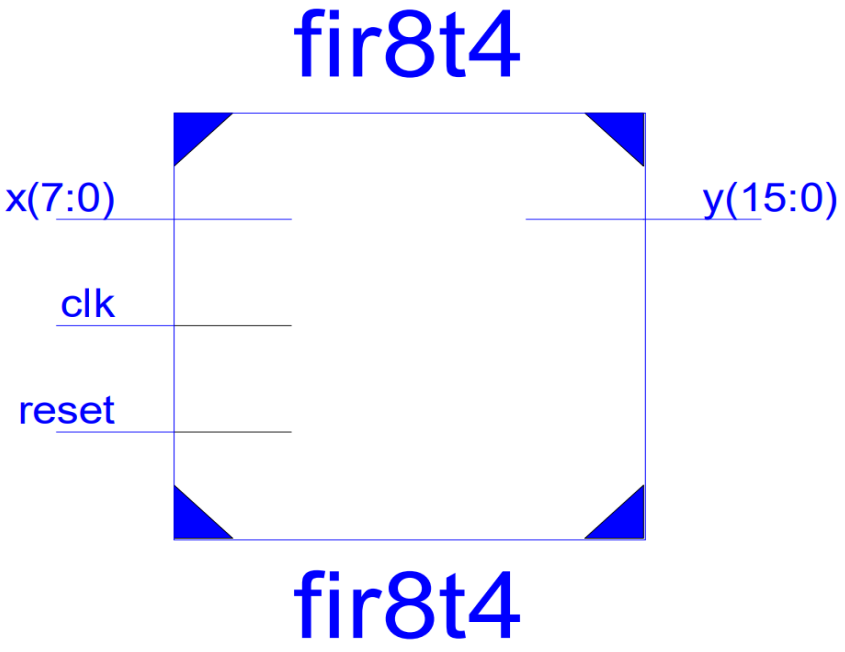


Fig 5.1: Black box representation of FIR Filter

5.2 RTL Schematic:

An FIR filter's underlying structure and data flow are depicted in its Register-Transfer Level (RTL) abstracted RTL graphic depiction. The signal processing activities of the filter are commonly displayed as sequential registers for storing input samples, multipliers for coefficient multiplication, adders for accumulation, and registers for output storage.

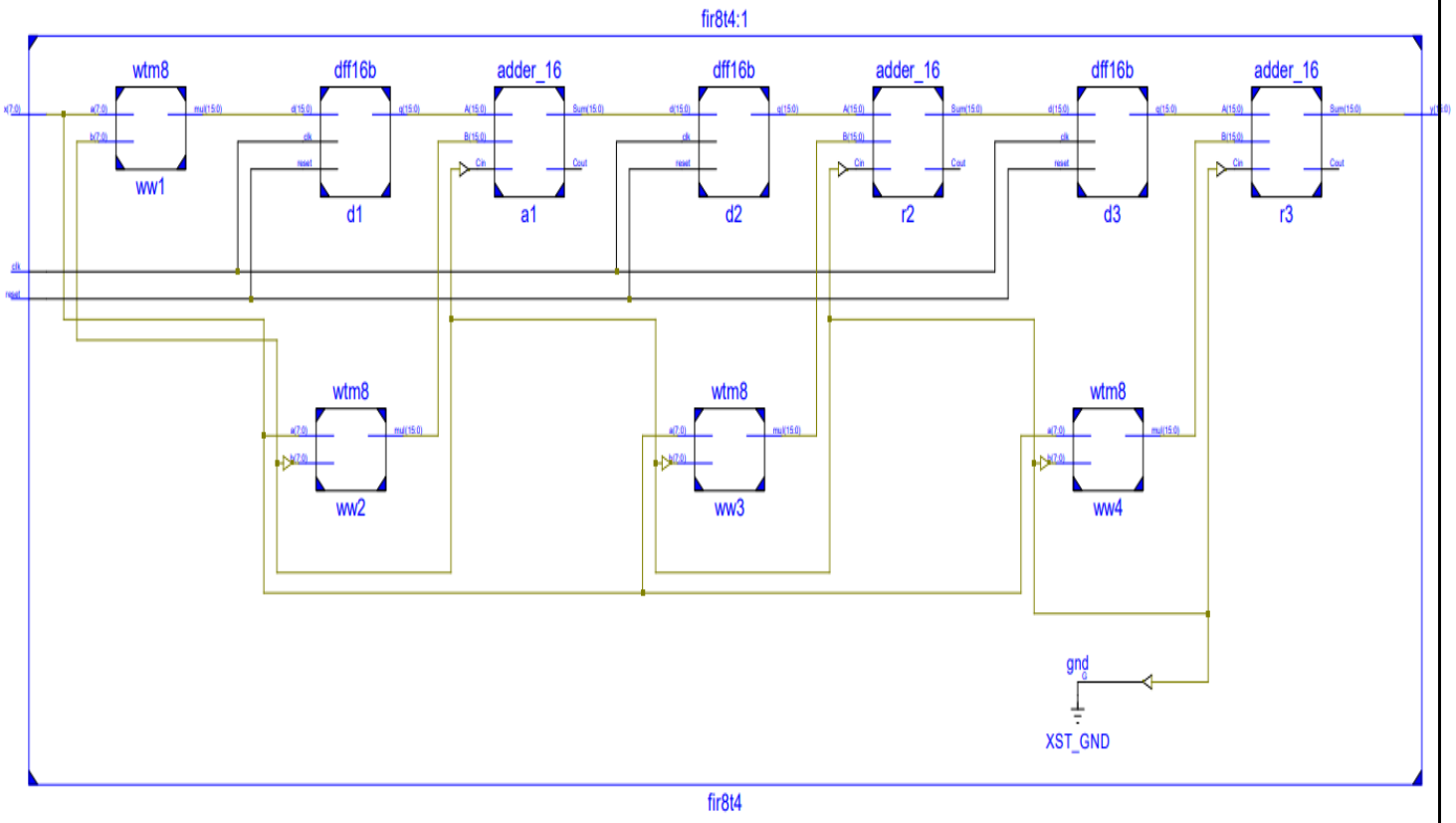


Fig 5.2: RTL Schematic view of FIR Filter

RTL (Register Transfer Level) schematic view provides a representation of digital circuits at a level of abstraction that describes how data is transferred and manipulated between registers. It's a high-level view of the circuit's functionality, showing the interconnection of logical elements and the flow of data between them. RTL schematic view provides a behavioral description of the circuit's functionality, describing how data is transferred and manipulated between registers. It abstracts away the details of the underlying hardware implementation, focusing on the logical operation of the circuit.

5.3Technology Schematic:

An FIR filter's technology diagram illustrates how its constituent parts are physically implemented within an integrated circuit. It illustrates how transistors, gates, and other electronic components are arranged and connected on the chip, describing how the silicon-level functioning of the filter is achieved. This representation offers information on the physical dimensions, signal routing, and general structure of the circuit, facilitating study of variables like power consumption, signal integrity, and space utilization.

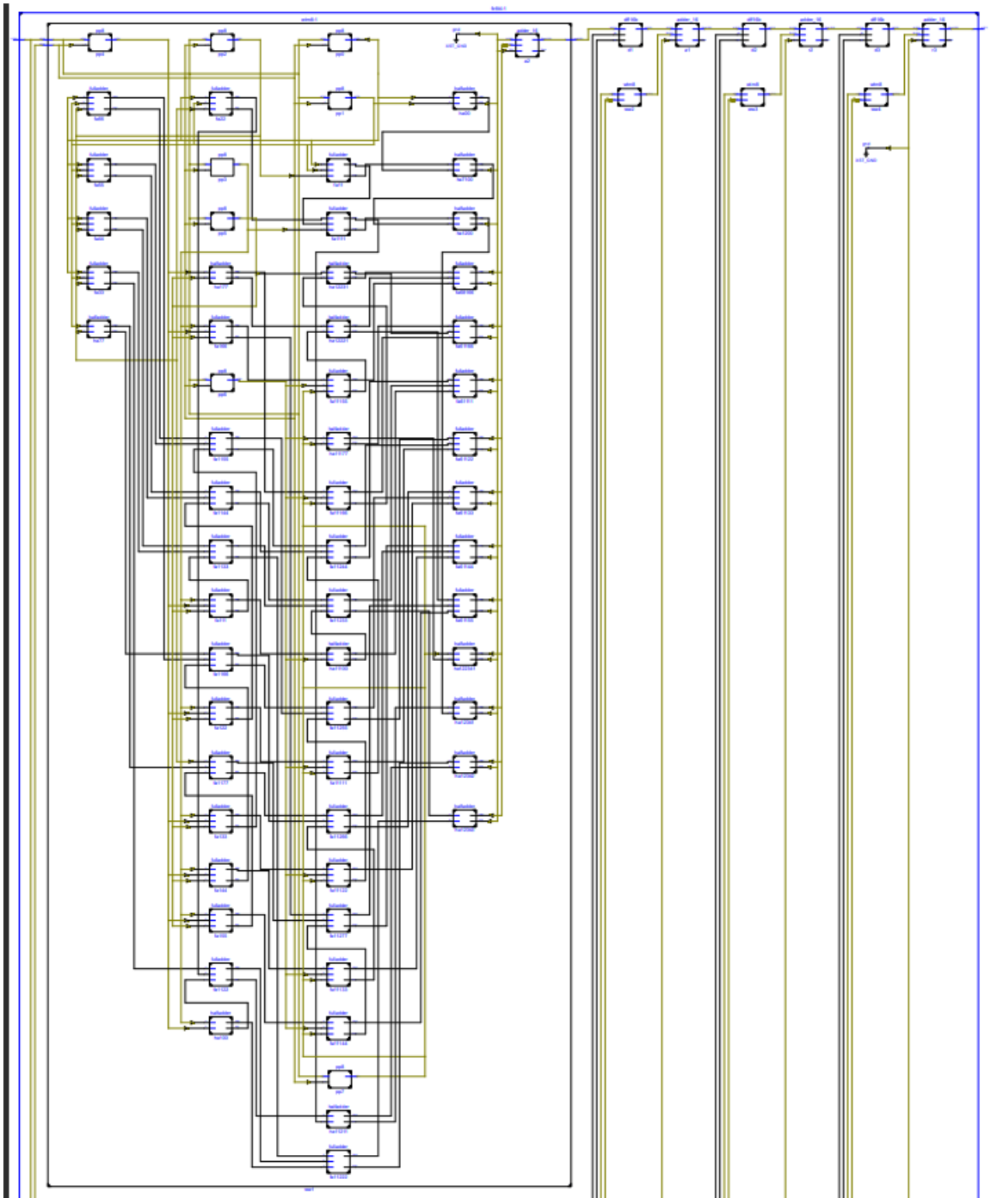


Fig 5.3: Technology Schematic view of FIR Filter

5.4 Simulation Waveforms:

The FIR filter's reaction to input stimuli is seen in the simulation output. Waveform traces showing the input signal, the filtered output signal, and several internal signals within the filter are usually shown. The filtering process is captured by the simulation, demonstrating how the filter responds to variations in the input signal over time. Engineers may assess the performance of the filter, including features like frequency response, transient behavior, and signal distortion, by examining the waveform traces. When implementing the FIR filter design in hardware, this simulation result is a useful tool for confirming its operation and accuracy

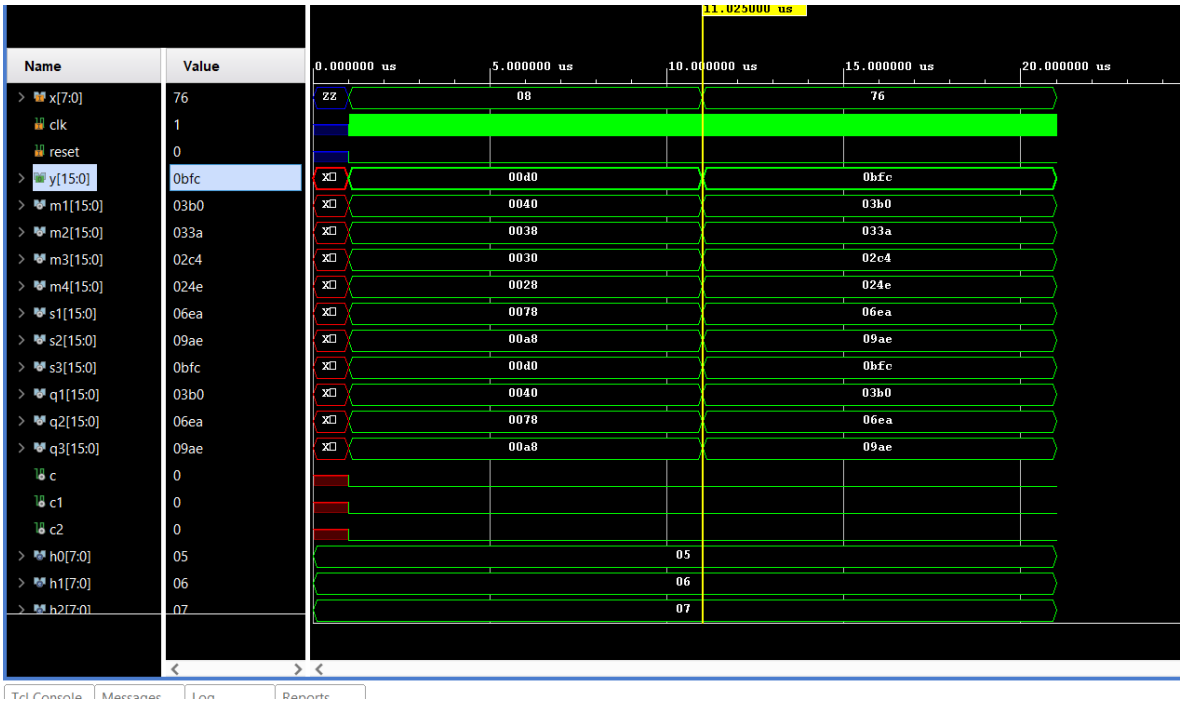


Fig 5.4: Simulation Waveforms of FIR filter

5.5 Timing summary:

The FIR filter's timing summary provides information about its performance metrics and operational properties. The filter reaches a minimum period of 3.542ns, or a maximum operating frequency of 282.318MHz, with a speed grade of -3. While output signals must fulfill a setup time requirement of 10.380ns after the clock edge, input signals must arrive at least 7.997ns before the clock edge. The filter's maximum combinational path delay, which represents the largest delay experienced along its critical route, is 12.149 ns. Important information about the filter's speed, timing limitations, and overall performance in practical applications may be found in these timing requirements.

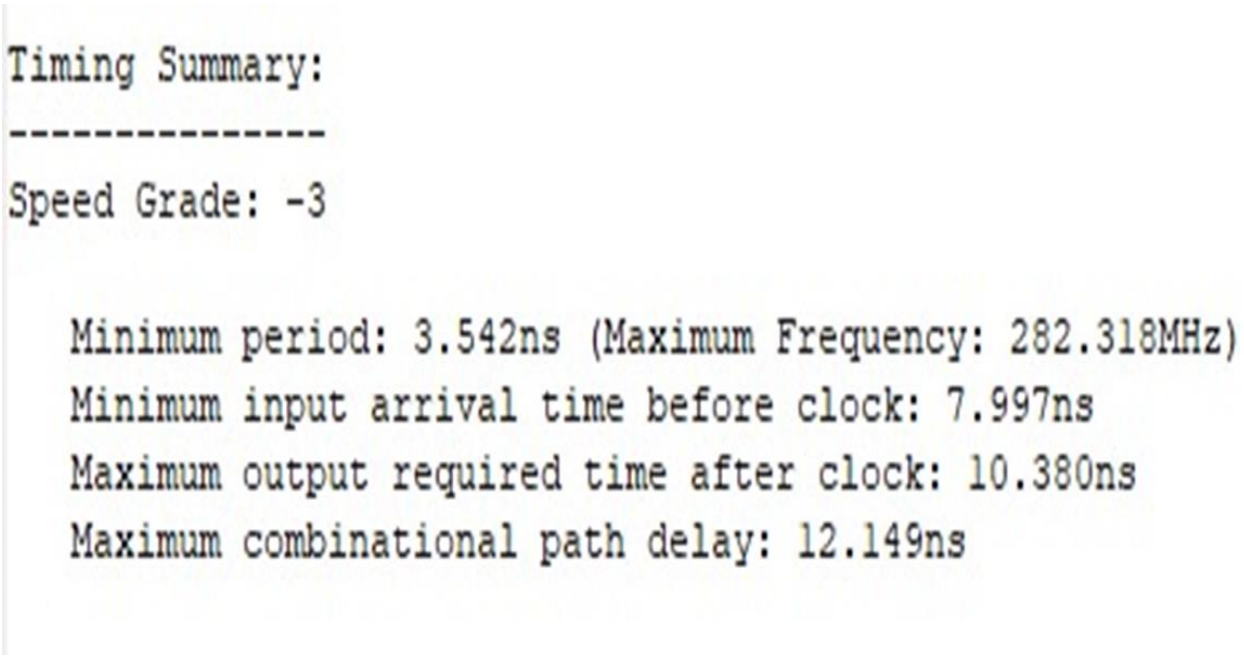


Fig 5.5: Timming summary

CHAPTER 6

CONCLUSION

6.1. Conclusion:

The Finite Impulse Response (FIR) filter emerges as an essential element in the realm of digital signal processing (DSP), offering a versatile toolset for a multitude of applications. Its adaptability and precision make it indispensable, providing users with the means to finely tune frequency response characteristics and execute complex signal processing tasks. This level of control is achieved through the strategic manipulation of input signals using a predetermined set of coefficients.

However, the successful implementation of a low-pass FIR filter design using Verilog necessitates meticulous attention to detail. Design considerations and performance metrics play pivotal roles in shaping the efficacy of the filter, whether it's realized in software or hardware. Each aspect, from conceptualizing the filter's behavior as a black box to meticulously crafting its technological schematic and conducting thorough timing analyses, contributes to a comprehensive understanding and optimization of its performance.

In the grand scheme of digital systems, the FIR filter assumes a critical role, serving as a linchpin for precise and efficient signal processing across diverse contexts. Its widespread adoption underscores its importance, enabling advancements in telecommunications, audio processing, and beyond. Ultimately, the FIR filter stands as a cornerstone upon which modern digital systems rely, facilitating the attainment of sophisticated signal processing objectives with unparalleled accuracy and efficiency.

Moreover, the inherent efficiency of the Dadda multiplier translates into reduced resource utilization within the FIR filter implementation. By optimizing the utilization of hardware resources, such as logic elements and routing interconnects, the design becomes more streamlined and cost-effective. This efficiency not only enhances the performance of the FIR filter but also contributes to the overall scalability and affordability of the digital system in which it is deployed.

Additionally, the Dadda multiplier technique exhibits favorable power consumption properties compared to traditional multiplication approaches. By minimizing the propagation delays and reducing unnecessary switching activity inherent in conventional multiplication circuits, the Dadda multiplier consumes less power while achieving comparable or superior performance. This reduction in power consumption is particularly advantageous in battery-powered or energy-constrained applications, where optimizing power efficiency is paramount.

In summary, integrating the Dadda multiplier and Brent-Kung adder technique into the FIR filter design enhances its speed, efficiency, and power consumption properties. By leveraging parallel processing

capabilities, optimizing resource utilization, and minimizing power consumption, this approach enables the FIR filter to deliver superior performance while meeting the stringent requirements of modern digital systems.

6.2 Advantages:

1. **Faster Filtering:** The Wallace tree multiplier within the FIR filter design, you can achieve faster overall filtering performance.
2. **Improved Efficiency:** The potential reduction in hardware complexity from the Wallace tree multiplier can lead to more efficient implementations of FIR filters, especially in hardware like FPGAs.
3. **Reduced Hardware Complexity:** Reduction in the number of logic gates required for multiplication compared to other techniques.
4. **Ease of Implementation:** Due to their finite impulse response, FIR filters are simpler to implement in hardware and software compared to other filter types like IIR filters.
5. **Linear Phase Response:** FIR filters can be designed to have a specific and linear phase response within the passband.

6.3 Applications:

Certainly! Verilog-based digital filter design with Wallace tree multipliers finds wide applications across various industries and domains, including:

1. **Telecommunications:** Used for channel equalization, signal modulation, and error correction in wireless communication systems.
2. **Consumer Electronics:** Employed in smartphones, audio devices, and televisions for audio and video processing tasks such as noise reduction and signal enhancement.
3. **Medical Devices:** Utilized in medical imaging systems, patient monitoring devices, and diagnostic equipment for filtering bio signals and image processing.
4. **Automotive:** Integrated into automotive radar systems, driver assistance systems, and infotainment systems for signal processing and data filtering.
5. **Aerospace and Defense:** Applied in radar, sonar, navigation systems, and satellite communications for signal processing, target detection, and noise filtering.
6. **Industrial Automation:** Deployed in control systems, robotics, and sensor networks for data filtering, condition monitoring, and predictive maintenance.
7. **Digital Audio Workstations (DAWs):** Utilized for real-time audio processing, effects, and mixing in music production and recording studios.
8. **Video Processing:** Employed in surveillance cameras, video conferencing systems, and drones for

video enhancement, compression, and noise reduction.

9. **Internet of Things (IoT):** Integrated into IoT devices and edge computing systems for sensor data processing, anomaly detection, and smart home automation.
10. **Financial Services:** Used in algorithmic trading systems and financial analytics platforms for processing market data and signal filtering.

These wide-ranging applications demonstrate the versatility and importance of Verilog-based digital filter design with Wallace tree multipliers in modern technology-driven industries.

6.4. Future Scope:

Growing demand for high-performance signal processing will drive the use of FIR filters with Wallace tree multipliers. Advancements in VLSI technology will make complex multiplier architectures more feasible. Exploration of hybrid approaches and hardware-specific designs for further optimization. Potential applications in machine learning and deep learning for accelerated computations.

6.5. References:

1. R. Kumar, K. H. L. P. Prasad, B. Sriraj, and P. R. Rajasri, “FPGA Implementation of Systolic FIR Filter Using Single-Channel Method,” in 2023 13th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Jan. 2023, pp. 215–220. doi: 10.1109/Confluence56041.2023.10048843.
2. V. Thamizharasan and N. Kasthuri, “FPGA implementation of high performance digital FIR filter design using a hybrid adder and multiplier,” Int. J. Electron., vol. 110, no. 4, pp. 587–607, Apr. 2023, doi: 10.1080/00207217.2022.2098387.
3. N. Chabini and R. Beguenane, “FPGA-Based Digital FIR Filters With Small Coefficients and Large Data Input,” in 2023 IEEE 13th Annual Computing and Communication Workshop and Conference (CCWC), Mar. 2023, pp. 0218– 0221. doi: 10.1109/CCWC57344.2023.10099131.
4. V. Jamuna, P. Gomathi, and A. Arun, “Design and Implementation of FIR Filter Architecture Using High Level Transformation Techniques,” Indian J. Sci. Technol., vol. 11, no. 17, pp. 1–5, May 2018, doi: 10.17485/ijst/2018/v11i17/122769.
5. N. Udaya Kumar, U. Subbalakshmi, B. Surya Priya, and K. Bala Sindhuri, “VLSI Implementation of FIR Filter Using Different Addition and Multiplication Techniques,” 2018, pp. 483–490. doi: 10.1007/978-981-13-1936-5_51. 6. Q. Fu, G. Zhang, X. Wu, and S. Yan, “Optimized Design of FIR Filter Based On FPGA,” J. Phys. Conf. Ser., vol. 1626, no. 1, p. 012143, Oct. 2020, doi: 10.1088/1742-6596/1626/1/012143