Efficient Multiplierless FIR Filter Design: Minimizing Adder Count for Optimized Filtering

A project report submitted in partial fulfillment of the requirements for the award of the degree of

B.Tech.

in

Electronics and Communication Engineering

by

KAMBALA CHANDU (620147)

NAMALA SAI SREEKAR (620217)

PRODDUTURI SIRI (620228)

Department of Electronics and Communication Engineering

NATIONAL INSTITUTE OF TECHNOLOGY

ANDHRA PRADESH-534101

MAY 2024

BONAFIDE CERTIFICATE

This is to certify that the project titled **Efficient Multiplierless FIR Filter Design: Minimizing Adder Count for Optimized Filtering** is a bonafide record of the work done by

KAMBALA CHANDU (620147)

NAMALA SAI SREEKAR (620217)

PRODDUTURI SIRI (620228)

in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **ECE** of the **NATIONAL INSTITUTE OF TECHNOLOGY, ANDHRA PRADESH**, during the year 2023-2024.

Mrs.DHANASREE JILLELLA	**Dr.S.Yuvaraj**
Project Guide	Head of the Department

ABSTRACT

With the aid of adders and shift-add architecture, a multiplierless Finite Impulse Response (FIR) filter is designed and optimized in this project. The suggested method uses the Multiple Constant Multiplication (MCM) technique to make effective use of constants, which saves time and energy. Adder blocks, which are essential for implementing multiplication-free operations, are the main components used in the filter design. The project's goal is to compare several adder types in order to provide an optimal filter design. The speed, area utilisation, and power efficiency of several adder architectures, such as ripple-carry adders, carry-propagate adders, sum-propagate adders is assessed. The research looks at several adder configurations in an effort to find the best adder design that minimizes resource consumption and satisfies required filter parameters.The study intends to show the efficiency of the suggested multiplierless FIR filter design approach through rigorous analysis and simulation. The ultimate objective is to create a high-performance FIR filter with less computational complexity so that it can be used in applications where hardware resources and power consumption are strictly limited.

Keywords: FIR filter, MCM, sum-propagate adders.

ACKNOWLEDGEMENT

We would like to thank the following people for their support and guidance without whom the completion of this project in fruition would not be possible.

Mrs.DHANASREE JILLELLA, our project guide, for helping us and guiding us in the course of this project .

Dr. S. Yuvaraj, the Head of the Department, Department of ECE.

Our internal reviewers, **Mrs. Dhanasree J**, **Dr. Kiran Kumar G**, **Dr. V Prakash Singh** for their insight and advice provided during the review sessions.

We would also like to thank our individual parents and friends for their constant support.

TABLE OF CONTENTS

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

The purpose of this project is to create an efficient multiplier-less Finite Impulse Response (FIR) filter with adder blocks and shift operations. FIR filters are widely utilized in various signal processing applications, including audio and picture processing and communications. In classic FIR filter implementations, multipliers are employed to multiply input samples by filter coefficients. However, multipliers use a lot of resources and have a high power consumption. To solve this, we intend to create a multiplier-free FIR filter using adder blocks and shift operations to reduce hardware complexity and power consumption.

This project focuses on two main techniques: the Multiple Constant Multiplication (MCM) method and the use of adder architectures with shift operations. The MCM method involves representing the filter coefficients as a sum of smaller constants, reducing the need for multipliers. Instead of using dedicated multipliers, effectively scaling the input samples by powers of 2. The shifted samples will then be accumulated using efficient adder block architectures, such as carry-propagate adders, sum-propagate adders, to achieve high-speed operation and minimal delay. By adopting these techniques, we aim to design an efficient multiplier-less FIR filter that maintains the desired filtering characteristics while reducing hardware complexity and power consumption. The project will involve implementing and evaluating various adder block architectures, optimising the shift operations, and comparing different approaches to achieve the efficient design.

Chapter 2

Review of Literature

[1] **M. Faust, O. Gustafsson and C. -H. Chang, "Reconfigurable multiple constant multiplication using minimum adder depth"** The Reconfigurable Multiple Constant Multiplication (ReMCM) problem involves devising a cost-effective network comprising shifts, additions, subtractions, and multiplexers to implement the multiplication of a single input variable with various coefficient sets. Regarding polyphase decimation filters, the relationship between filter length and decimation factor significantly influences implementation costs. Experimental results demonstrate that ReMCM implementation can achieve up to a 38percent reduction in area for a decimation factor of 8 compared to a parallel implementation of polyphase subfilters. Additionally, solutions to single output problems yield results comparable to established algorithms.

[2] **R. Garcia and A. Volkova, "Toward the Multiple Constant Multiplication at Minimal Hardware Cost"** In this study, we enhance the existing optimal method for Multiple Constant Multiplication (MCM) over integers, crucial in highly optimized hardware for embedded systems. Our approach involves replacing resource-intensive generic multiplication with more efficient bit-shifts and additions, creating a multiplierless circuit. Incorporating error propagation rules into our ILP model ensures adherence to user-specified error bounds on MCM results. Through extensive experimentation, we evaluate our models and conduct a comprehensive analysis of the impact of design metrics on synthesized hardware.

[3] **G. Thakur, H. Sohal and S. Jain, "Design and Analysis of High-Speed Parallel Prefix Adder for Digital Circuit Design Applications"** Adders are essential com-

ponents in various applications, particularly in signal processing, where they serve as foundational building blocks. With the prevalent use of integrated circuits in modern electronics, the demand for efficient adder modules is paramount. This study investigates different types of adders, including Ripple-carry (RC), Carry-skip (CSk), Carry-lookahead (CL), and Kogge-stone (KS) adders, addressing specific design constraints such as speed and area in digital VLSI circuits. A novel KS adder is proposed in this paper for various bit-lengths, with a focus on analyzing delay. Results show that the proposed adder achieves delays of 7.487ns, 9.248ns, 10.295ns, 7.259ns, and 6.814ns for 8, 16, 32, 64, and 128 bit-lengths, respectively.

[4] **B. Koyada, N. Meghana, M. O. Jaleel and P. R. Jeripotula, "A comparative study on adders"**In digital circuits, the addition of a specific number of bits is a fundamental operation aimed at simplifying circuit complexity and enhancing its functionality. The selection of the appropriate adder with the required properties is crucial for ensuring efficient circuit operation. These properties significantly influence the operation and performance of the adders. The compared adders include Ripple Carry Adder, Carry Look Ahead Adder, Carry Save Adder, Carry Skip Adder, Carry Select Adder, Modified Carry Select Adder, and Kogge Stone Adder. The evaluation was based on two primary aspects: the number of slices and speed.

[5] **M. Kumm, A. Volkova and S. -I. Filip, "Design of Optimal Multiplierless FIR Filters With Minimal Number of Adders"** This study introduces two innovative methods aimed at optimizing both the design and hardware implementation of a finite impulse response (FIR) filter without using multipliers. These methods operate by either fixing the number of adders for product implementation (multiplier block adders) or bounding the adder depth (AD) for these products, particularly useful for designing filters with minimal AD for low-power applications. Unlike previous multiplierless FIR filter techniques, our methods prioritize adder count optimality. Extensive numerical experiments demonstrate that our simultaneous filter design approach often outperforms existing approaches cited in the literature.

Chapter 3

Proposed Work with its Methodology

3.1 Existing Method

This study proposes a novel method for implementing block Finite Impulse Response (FIR) digital filters using the Multiple Constant Multiplication (MCM) technique, aiming at high-speed and low-power processing. This method breaks this barrier by demonstrating MCM's effectiveness in the direct-form implementation of FIR filters. Experimental results across various filter orders and lengths show significant improvements in maximum sampling rate and energy-delay product, up to 3.5 and 7.6 times, respectively, compared to traditional approaches. Additionally, a potential reduction in total area by up to 20 percent compared to conventional filter implementations is observed. These findings open new possibilities for efficient and high-performance digital filtering in resource-constrained applications.

3.2 Previous Work

In the previous semester, our focus was on implementing Single Constant Multiplication (SCM) and Multiple Constant Multiplication (MCM) operations. Explored various techniques for efficiently performing these operations in digital circuits. Additionally, analysed the study of different types of adders and their characteristics.

One significant aspect of our study was the comparison of parameters between the coefficients generated by the MCM operations and the adder blocks used in Finite Impulse Response (FIR) filters. By analyzing these parameters, aimed to optimize the

design and performance of FIR filters in digital signal processing applications.

Through our investigation, gained insights into the trade-offs between different adder architectures, such as Ripple Carry Adders (RCA), Carry Look-Ahead Adders (CLA), and other advanced designs. Evaluated their performance metrics, including delay, power consumption, and resource utilization, to determine their suitability for implementing FIR filters.

By integrating the knowledge gained from SCM/MCM implementations with the study of adders, aimed to develop a comprehensive understanding of digital signal processing systems. This interdisciplinary approach allowed us to explore the synergies between arithmetic operations and filter design, paving the way for more efficient and optimized implementations in future projects and applications.

3.3 Proposed Method

The proposed method is designing FIR filters with coefficients using MCM (Multiple Constant Multiplication) methods and compare it with various adder architectures including RCA (Ripple Carry Adder), PPA (Parallel Prefix Adder), Sum-Propagate Adders, Carry Propagate Adders, Ladner Fisher and Han-Carlson, outlined the following steps:

Coefficient Decomposition: Decompose the FIR filter coefficients into constants and use MCM techniques to efficiently implement the multiplications. This involves expressing filter coefficients as sums of constant multiples, minimizing the number of unique constants required.

Parallel Architecture Design: Design the filter architecture to leverage parallelism for efficient computation. Each coefficient multiplication can be performed concurrently using dedicated hardware resources.

Adder Block Selection:Evaluate different adder architectures for their suitability in the filter implementation. Assess their speed, area, and power consumption characteristics.

Comparison Metrics: Define metrics for comparison, including performance (such as

speed and throughput), hardware resource utilization (area), and power efficiency.

Implementation and Simulation: Implement the FIR filter using the proposed MCM-based method and each adder architecture in a simulation environment. Evaluate the filters' performance and resource utilization under various conditions and input signals.

Performance Analysis: Analyze the simulation results to compare the performance of the FIR filters with different adder architectures. Consider factors such as maximum operating frequency, latency, and resource utilization.

Trade-off Analysis: Assess the trade-offs between speed, area, and power consumption for each adder architecture. Identify the most suitable architecture based on the application requirements and design constraints.

Conclusion: Concluding the study by summarizing the findings and recommending the most effective approach for implementing FIR filters with coefficients using MCM methods, considering the performance and resource utilization trade-offs.

By following these steps, a method for FIR filters is implemented with coefficients using MCM techniques and compare it with various adder architectures to determine the optimal design for our specific application.

3.4 FIR Filter

FIR (Finite Impulse Response) filters are digital filters used in signal processing to modify or enhance the characteristics of a signal. They are widely used in areas such as audio processing, image processing, communications, and sensor applications.The term ”finite impulse response” refers to the fact that the output of an FIR filter is only determined by a finite number of previous inputs. This means that an FIR filter does not have any feedback, which simplifies its implementation compared to other types of filters.The impulse response (that is, the output in response to a Kronecker delta input) of an Nth-order discrete-time FIR filter lasts exactly N+1 samples (from first nonzero element through last nonzero element) before it then settles to zero.

FIR filters operate by convolve a sequence of input samples with a set of coeffi-

cients, or tap weights, to produce the filtered output. The coefficients define the desired frequency response of the filter, allowing for various types of filtering operations such as low-pass, high-pass, band-pass, or even more complex filters.

Overall, FIR filters are powerful tools for signal processing due to their flexibility, ease of implementation, and desirable properties like linear phase response. They are widely used in various applications to achieve desired filtering effects and enhance the quality of signals.

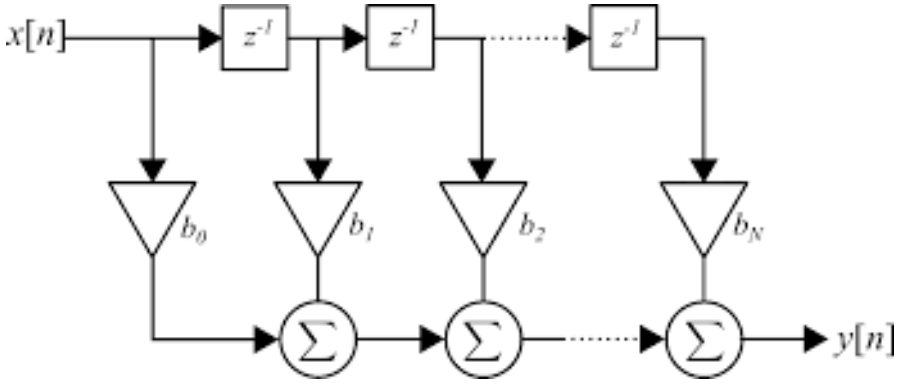


Figure 3.1: FIR Filter

3.5 Constant Multiplication

Constant multiplication, a fundamental operation in digital signal processing (DSP) and various other fields, involves multiplying a variable by a fixed constant value. Efficient implementation of this operation is crucial for optimising performance in applications like communication systems and numerical computation. Several methods exist for achieving constant multiplication in digital circuits.

One approach is the shift-and-add method, where the multiplication operation is executed using a combination of shift and addition operations. For instance, to multiply a number by 3, the number is shifted left by 1 bit (equivalent to multiplying by 2) and then added to the original number.

Another method involves using lookup tables, which store precomputed results of the multiplication operation for each possible input value. When multiplication by a constant is needed, the corresponding precomputed value is retrieved from the table.

This method is efficient for small constant values and can be implemented using memory elements like ROM.

Bit-level optimisation techniques can also be applied for certain constant values, simplifying the multiplication operation. For example, multiplying by a power of 2 can be achieved by shifting the input value left by a corresponding number of bits.

In scenarios requiring multiple constant multiplications performed simultaneously, parallel multiplier architectures can be employed to improve throughput. These architectures utilise multiple multiplier blocks operating in parallel to process multiple inputs concurrently.

3.5.1 Multiplexer-Based Reconfigurable Constant Multiplication

Constant multiplication is done using additions, subtractions and bit shifts only. The output constant c can be switched between a limited predefined set of N constants during run-time using multiplexers in the arithmetic data path. The multiplication cnx is performed, while x is a fixed-point input and n is the index of the selectable output constant cn.

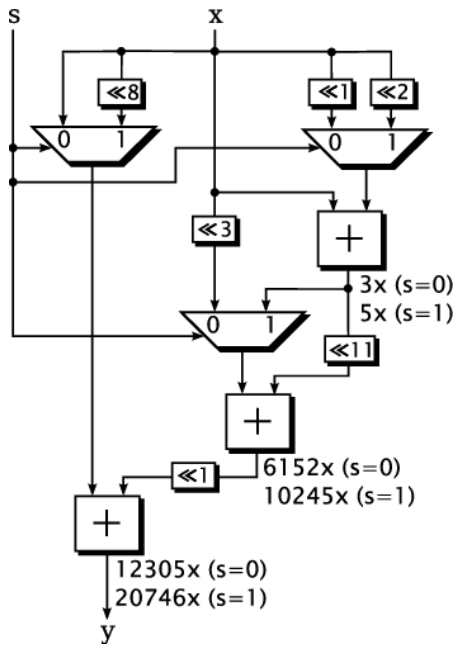


Figure 3.2: Reconfigurable multiplier with the coefficient set 12305, 20746.

3.5.2 Shift-Add Architrcture

For the pair, 29x and 43x without partial product sharing algorithm requires six addition and six shifting operations Using Common Sub-Expression Elimination Method algorithm requires four additions and four shifting operations. That allows great reductions in the number of operations and consequently, in area and power dissipation of the MCM design at the gate leveL.Minimum number of addition and subtraction operations that implement the constant multiplications.

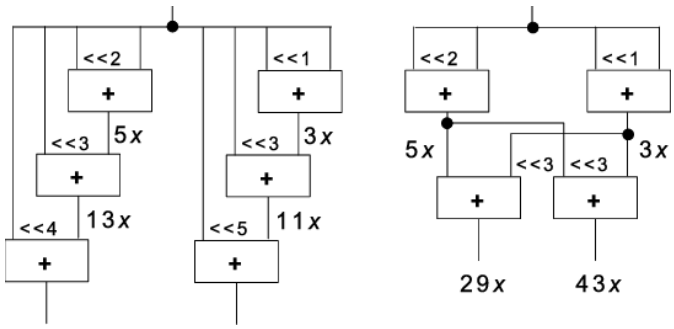


Figure 3.3: shift add

3.5.3 Single constant multiplication (SCM)

Single constant multiplication, a fundamental operation in digital signal processing (DSP) and various computational tasks, involves multiplying a variable by a fixed constant value. Efficient implementation of this operation is crucial for optimising performance in digital circuits. Several methods are commonly used to achieve single constant multiplication. One approach is the shift-and-add method, where the multiplication operation is executed through a combination of shift and addition operations. For instance, to multiply a number by 3, the number is shifted left by 1 bit (equivalent to multiplying by 2) and then added to the original number.

Example: In the below example the binary value for 29 is 11101 and in the second, third and fourth position the value is '1' and in that position the shift operation is performed.

$29x = (11101)b = x << 4 + x << 3 + x << 2 + x$

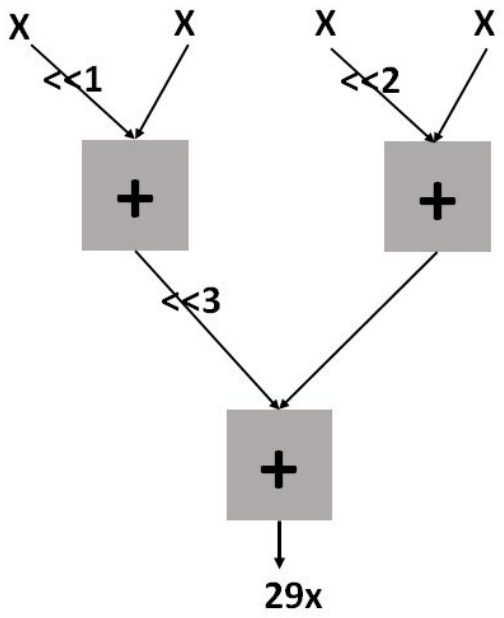


Figure 3.4: Single constant multiplication (SCM) example.

3.5.4 Multiple Constant Multiplication (MCM)

The multiple constant method is a sophisticated optimisation strategy frequently utilised in digital signal processing (DSP), numerical computations, and related fields to streamline the process of performing multiple constant multiplications. At its core, the method involves a two-step process aimed at reducing computational overhead while maintaining accuracy.

Firstly, during an offline phase, computations are carried out to determine the results of multiplying each possible input value by the set of constant values. These results are then stored in a lookup table, forming a mapping between input values and their corresponding precomputed multiplication outcomes. This pre-computation phase is a one-time process and typically involves extensive calculations to cover a wide range of potential input values and constant factors.

Secondly, during the runtime phase, when multiplication operations are required within an algorithm or computation, the multiple constant method allows for expedited processing. Instead of performing the actual multiplication calculation, the method involves accessing the precomputed result from the lookup table based on the current input value. This process effectively replaces the costly multiplication operations with

rapid memory lookups, significantly reducing the computational burden and accelerating the overall execution time of the algorithm.

The effectiveness of the multiple constant method lies in its ability to strike a balance between computational efficiency and memory utilisation. While it offers substantial performance enhancements by circumventing the need for repeated multiplication calculations, it does require additional memory resources to store the lookup table. The size of this table depends on the precision required for the results and the range of possible input values and constant factors.

Overall, the multiple constant method serves as a valuable optimisation technique in scenarios where algorithms involve multiple constant multiplications. By leveraging pre-computation and memory lookups, it enables efficient handling of computations, leading to improved performance and responsiveness in DSP, numerical computing tasks, and other applications where rapid processing is essential.

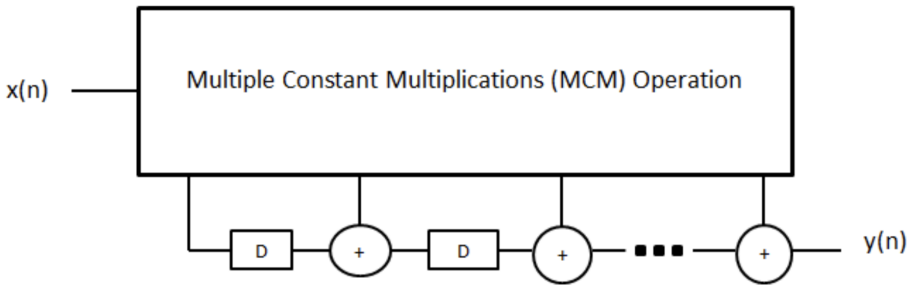


Figure 3.5: FIR filter implementation with an MCM block.

Example

$23x = (10111)b = x << 4 + x << 3 + x << 2 + x$

$7x = (101011)b = x << 3 - x$

3.6 Adders

3.6.1 Ripple carry adder

A Ripple Carry Adder (RCA) is a fundamental digital circuit employed to add two binary numbers. It stands as one of the simplest forms of adders utilized in digital electronics. The term ”ripple carry” stems from the necessity of each stage within the adder to await the carry bit from the preceding stage before computing its own sum and carry bits.

At its core, the RCA operates as follows: it accepts two binary numbers, denoted as A and B, as input. These numbers are typically represented using a series of bits, where each bit corresponds to a power of 2.

The addition operation occurs by summing corresponding bits from the two input numbers along with the carry-in (Cin) from the prior stage.The resultant sum bit (S) is determined by the addition of the corresponding bits from A, B, and Cin. If the sum of these bits is odd, the sum bit becomes 1; otherwise, it is 0. Simultaneously, the carry bit (Cout) indicates the carry-out from the current stage. If the sum of the corresponding bits from A, B, and Cin exceeds or equals 2, a carry-out (1) is generated; otherwise, it remains 0.

However, the sequential nature of carry propagation in RCA induces a propagation delay, impeding its performance. Consequently, RCA is ill-suited for high-speed arithmetic operations in contemporary digital systems. To mitigate this limitation, more advanced adder designs like carry look ahead adders and carry-select adders have been developed.

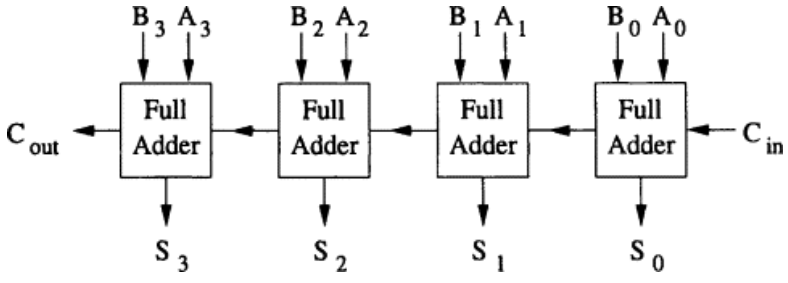


Figure 3.6: 4-Bit Ripple Carry Adder.

3.6.2 Carry Look ahead Adder

A Carry Look ahead Adder (CLA) represents a significant advancement in digital circuit design, particularly in binary addition operations. Unlike the traditional Ripple Carry Adder (RCA), which relies on sequential carry propagation, CLA computes carry signals for multiple bit positions simultaneously. This parallel computation minimizes the propagation delay inherent in carry generation, resulting in faster addition processes.

The key innovation of CLA lies in its approach to carry computation. Instead of waiting for carry signals to propagate from one bit position to another, CLA employs two critical terms: Generate (G) and Propagate (P). These terms enable CLA to assess whether a carry will be generated and whether the carry-in will propagate to the current bit position, respectively.

By computing carry signals in parallel, CLA optimizes the addition process, enhancing speed and efficiency. This parallelism is particularly beneficial for high-speed arithmetic operations, where minimizing latency is crucial. However, it's essential to note that CLA typically requires more complex hardware implementation compared to RCA due to its parallel processing nature.

Overall, Carry Look ahead Adders represent a significant advancement in digital circuit design, offering faster addition capabilities and improved efficiency, especially in applications requiring high-speed arithmetic operations.

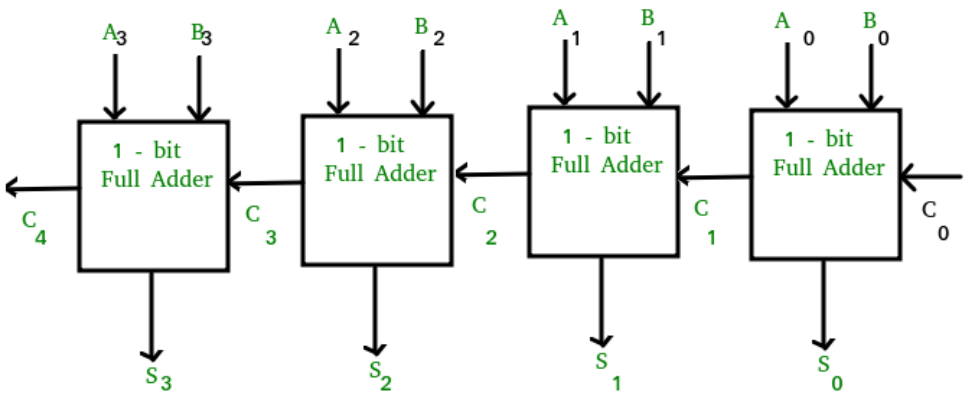


Figure 3.7: 4-Bit Carry Look Ahead Adder.

3.6.3 Carry Propagate Adder (CPA)

A Carry Propagate Adder (CPA) accelerates binary addition by parallelise carry signal computation, mitigating the propagation delay found in Ripple Carry Adders (RCA). In a CPA, each stage's carry-in is determined through simultaneous evaluation of Generate (Gi) and Propagate (Pi) terms. The Gi term indicates whether addition results in a carry, while the Pi term signifies carry propagation from the previous stage. By computing carry signals in parallel, CPAs eliminate the sequential dependency present in RCAs, leading to faster arithmetic operations.

Compared to RCAs, CPAs offer notable advantages in terms of speed. The simultaneous computation of carry signals ensures that each stage operates independently, reducing the overall propagation delay.

In summary, CPAs stand as a significant improvement over RCAs, providing faster binary addition through parallelized carry signal computation. Their reduced propagation delay and efficient resource utilization make them indispensable in high-speed arithmetic operations, addressing the demands of contemporary digital systems.

3.6.4 Sum Propagate Adder (SPA)

Sum propagate is a fundamental property of digital addition, especially in ripple carry adders. In ripple carry adders, each bit addition generates both a sum and a carry-out. The carry-out propagates to the next higher-order bit, while the sum propagates directly to the output.

SPAs may find utility in scenarios prioritizing resource efficiency, although they may incur higher delay due to backward carry propagation. Overall, the choice between CPAs and SPAs depends on the specific requirements and constraints of the application at hand.

3.6.5 Ladner-Fischer Adder

The Ladner-Fischer adder is a parallel prefix adder, also known as a carry-lookahead adder. It's named after its inventors, Peter Ladner and J. A. Fisher. This type of adder is designed to reduce the time complexity of carry propagation in large addition operations, particularly in hardware implementations.

Traditional adders, like ripple-carry adders, propagate the carry from one stage to the next, which can lead to significant delay, especially in long chains of adders. Carry-look ahead adders aim to address this by computing carry signals in advance based on the input operands.

The Ladner-Fischer adder specifically employs a recursive approach to compute carry signals efficiently. It recursively divides the addition operation into smaller subproblems, computing intermediate carry signals at each level and reducing the overall time complexity.

While the Ladner-Fischer adder can be more complex to implement compared to simpler adders like the ripple-carry adder, it offers significant improvements in speed for large addition operations, making it a popular choice in high-performance computing applications and in the design of integrated circuits.

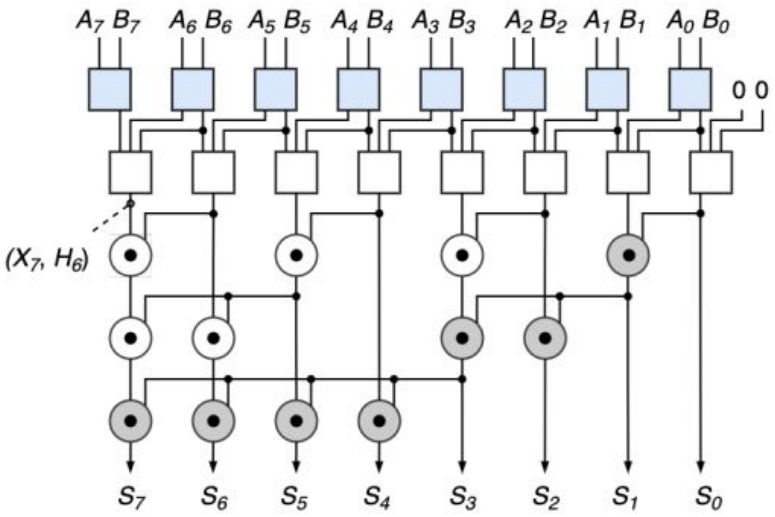


Figure 3.8: Ladner-fischer sum propagate structure.

3.6.6 Han-Carlson Adder

The Han-Carlson Adder, a variation of the Carry-Save Adder (CSA) introduced by B. Han and D. Carlson, stands out for its prowess in parallel addition, notably in applications like digital signal processing (DSP) and parallel multipliers. Operating initially with a carry-save addition approach, this adder calculates the sum and carry of each bit position separately. Unlike conventional methods where sum and carry are computed concurrently, carry-save addition enables parallelism and efficient hardware utilization.

The Han-Carlson Adder typically adopts a tree structure to amalgamate the carry-save sums and produce the final sum and carry. This architectural choice diminishes critical path delay and enhances overall performance. With its innate parallelism, this adder excels in high-speed arithmetic tasks, crucial for DSP algorithms and parallel multipliers.

Efficiency remains a hallmark of the Han-Carlson Adder, leveraging parallelism and carry-save addition to achieve elevated throughput with relatively simple hardware setups compared to alternative architectures. In summary, this specialized adder design optimizes parallel addition, delivering swift performance and efficient hardware usage, particularly prized in DSP and parallel processing realms.

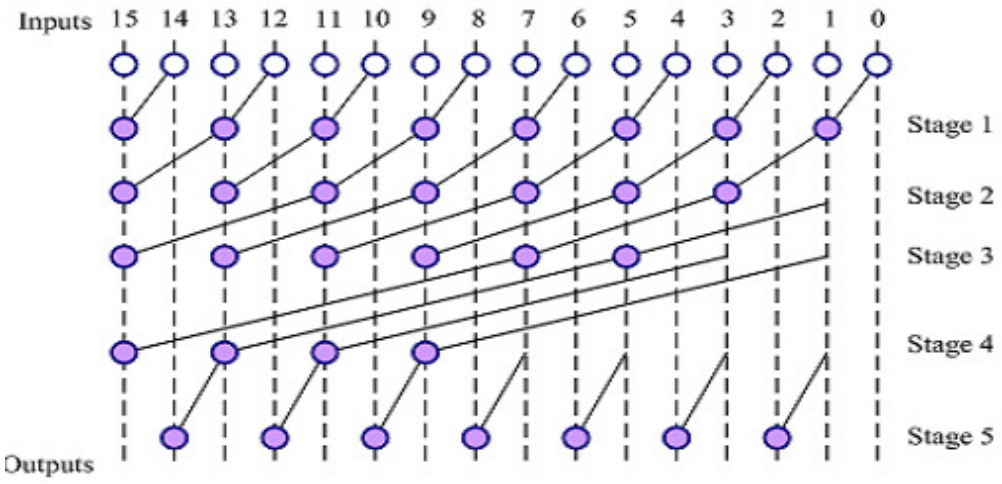
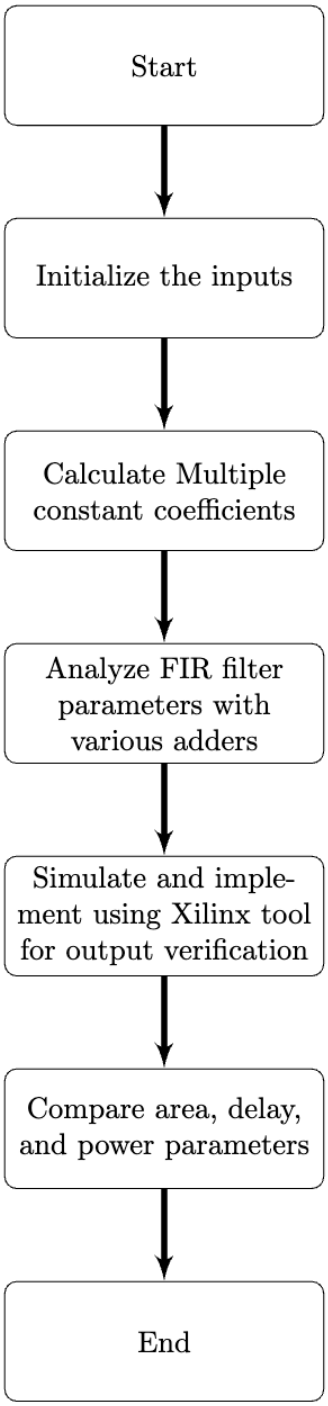


Figure 3.9: Han-Carlson carry tree structure.

3.7 Methodology

Figure 3.10: Flow Chart



3.8 Tools and Languages

Vivado Xilinx's FPGA and SoC design tool suite. Offers synthesis, implementation, and debugging. Intuitive interface, supports various FPGA families. IP integrator for rapid block integration. High-level synthesis (HLS) from C/C++. Debugging tools like ILA, VLA. Streamlined workflow for efficient design.

Verilog HDL Verilog is a hardware description language (HDL) widely utilized in digital circuit design. It serves as a means to model electronic systems, defining both their behavior and structure. With Verilog, designers can describe the functionality of digital circuits and simulate their performance before fabrication. Its modular and hierarchical structure allows for the creation of complex designs by combining smaller, reusable modules. Verilog supports both procedural and concurrent constructs, offering flexibility in design implementation. Due to its versatility and widespread adoption, Verilog is commonly used in application-specific integrated circuit (ASIC) and field-programmable gate array (FPGA) design, enabling the development of a wide range of electronic devices and systems.

Chapter 4

Simulation and Evaluation parameters

4.1 Simulation

4.1.1 Simulation results of MCM-FIR Filter

Input Parameters

Clock $->$ **Force Clock** Leading Edge Value - 1, Trailing Edge Value - 0 Period - 10ns

rst $->$ **Force Constant** Force Value - 0

x $->$ **Force Constant** Force Value - 1

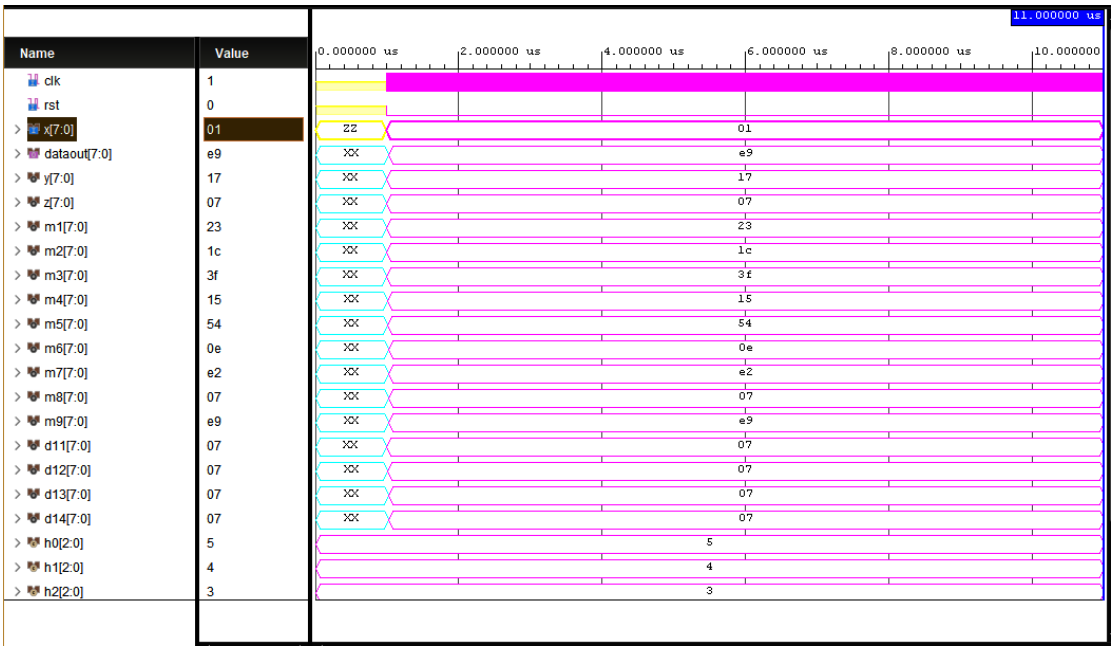


Figure 4.1: Waveform of Ladner-fischer adder(rst0).

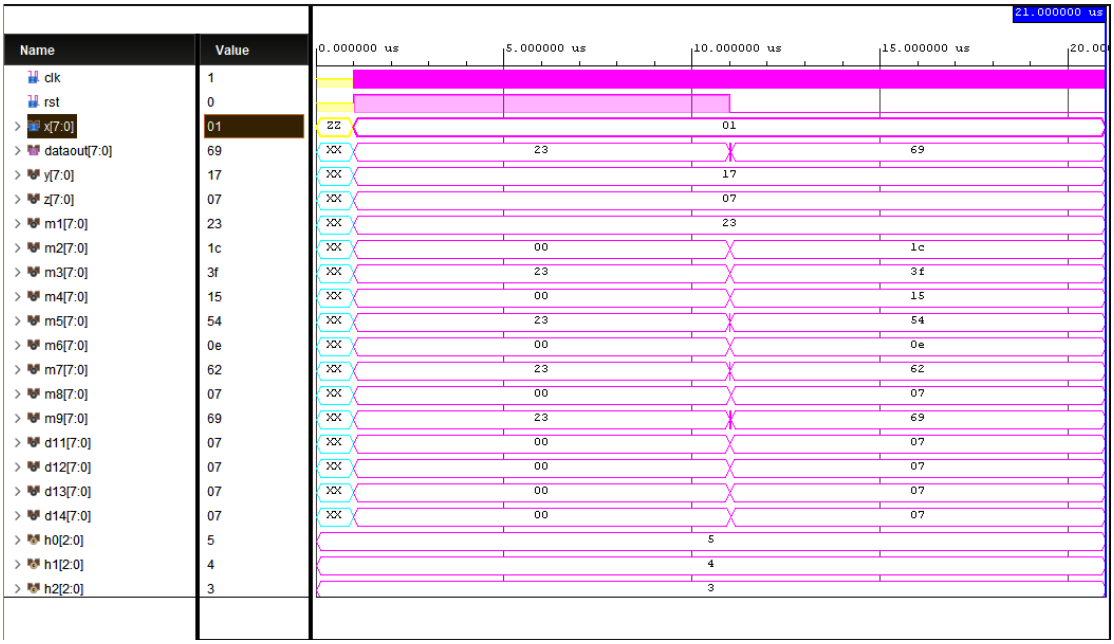


Figure 4.2: Waveform of Han-Carlson adder(rst0).

Input Parameters

Clock $->$ **Force Clock** Leading Edge Value - 1, Trailing Edge Value - 0 Period - 10ns

rst $->$ **Force Constant** Force Value - 1

x $->$ **Force Constant** Force Value - 1

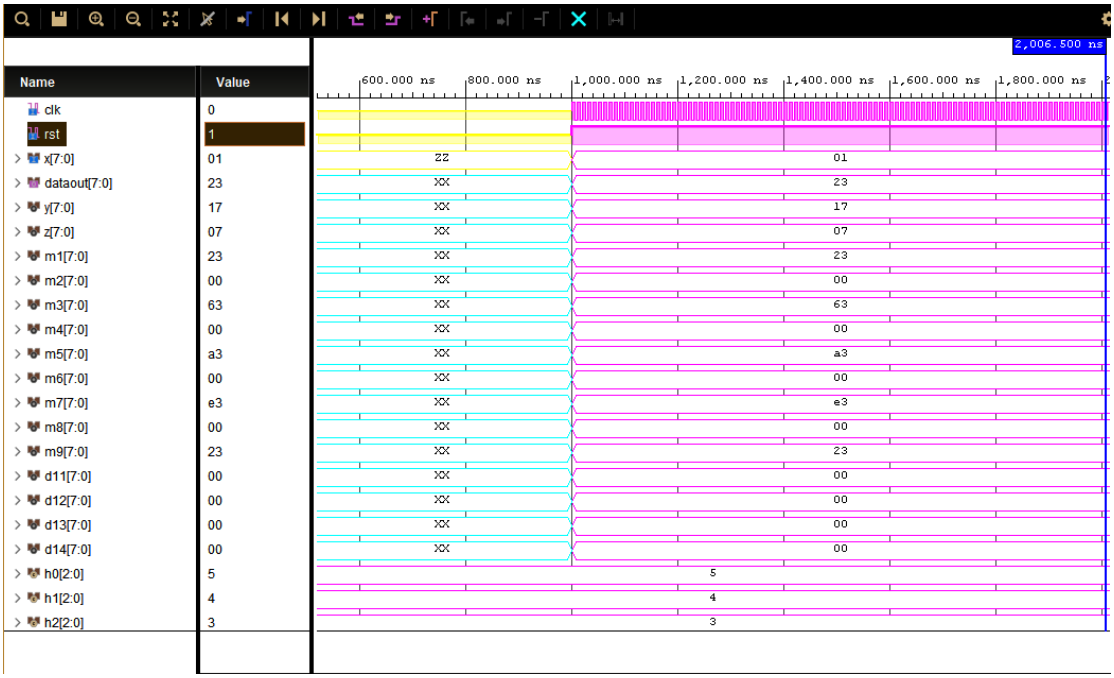


Figure 4.3: Waveform of Ladner-fischer adder(rst1).

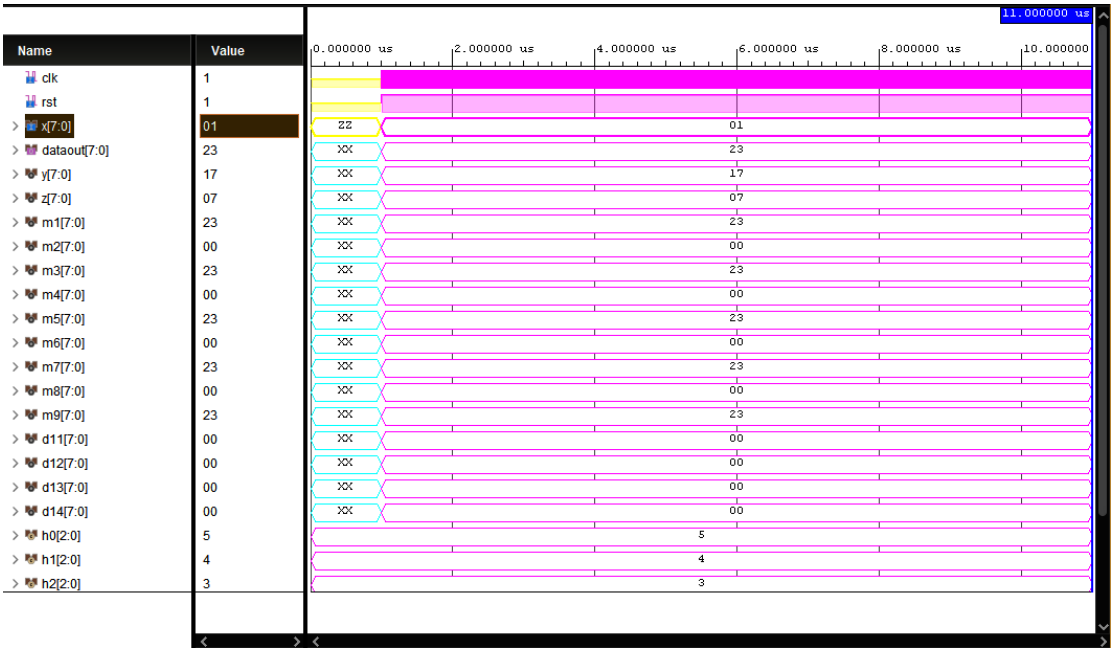


Figure 4.4: Waveform of Han-Carlson adder(rst1).

4.2 Evaluation Parameters

4.2.1 Timing Summary

The Timing Summary report in Xilinx FPGA design provides crucial insights into the timing performance of a design post-synthesis and implementation stages. It serves as a comprehensive overview, detailing critical aspects such as clock information, critical paths, timing constraints, slack, setup and hold violations, propagation and transition delays, clock uncertainty, and a summary of timing constraints. By meticulously examining this report, designers can pinpoint areas of concern, optimise critical paths, and adjust timing constraints as needed to ensure that the design operates reliably within the specified clock frequency. Ultimately, the Timing Summary report plays a pivotal role in verifying that the design meets timing requirements, enabling designers to deliver robust and high-performance FPGA implementations.

Setup Time: Setup time refers to the minimum time before the active clock edge (usually the rising edge) that the data input must be stable and valid for the flip-flop or register to reliably capture the data. By specifying ”none to none” for setup time, essentially saying that not imposing any specific requirement on how long before the clock edge

the data must be stable.

Hold Time: Hold time refers to the minimum time after the active clock edge that the data input must remain stable for the flip-flop or register to reliably capture the data. Similarly, setting ”none to none” for hold time indicates that no specific hold time requirement is imposed on the design.

Delay(Setup)

Runs | Timing | Utilization | Power

Unconstrained Paths - NONE - NONE - Setup

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	R
Path 1	∞	10	10	x[0]	dataout[6]	11.705	5.550	6.154	
Path 2	∞	10	10	x[0]	dataout[7]	11.308	5.576	5.731	
Path 3	∞	10	10	x[0]	dataout[5]	11.304	5.544	5.759	
Path 4	∞	10	10	x[0]	dataout[4]	10.829	5.550	5.278	
Path 5	∞	9	10	x[0]	dataout[3]	10.250	5.420	4.829	
Path 6	∞	9	10	x[0]	dataout[2]	9.925	5.614	4.310	
Path 7	∞	7	10	x[0]	dataout[1]	8.601	5.178	3.422	
Path 8	∞	4	5	u3/q_reg[0]/C	dataout[0]	5.754	3.926	1.829	
Path 9	∞	4	10	x[0]	u2/q_reg[6]/D	3.155	2.022	1.133	
Path 10	∞	4	10	x[0]	u2/q_reg[7]/D	3.063	1.930	1.133	

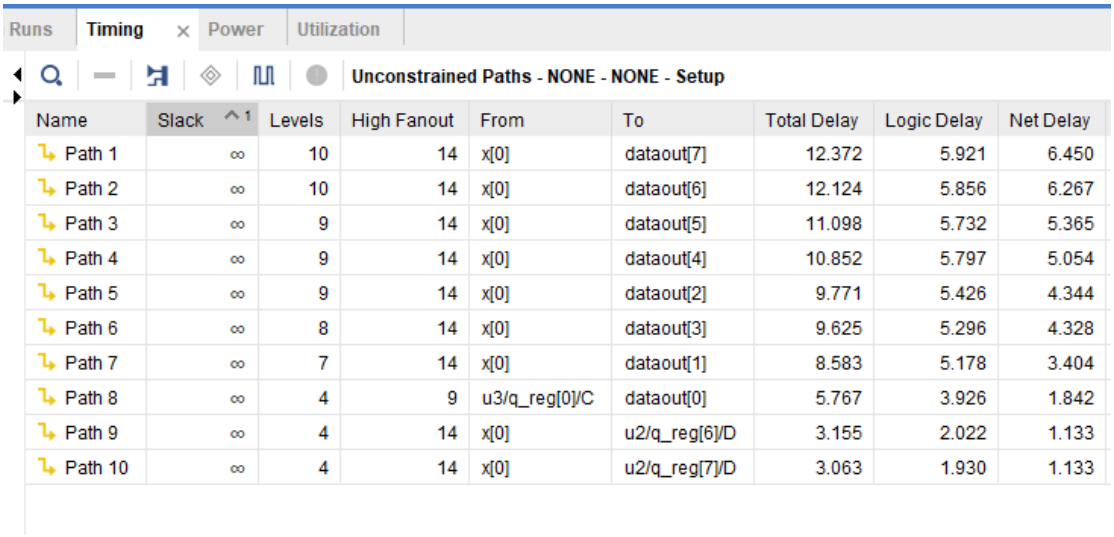
Figure 4.5: Delay of Ladner-fischer adder.

Runs | Timing | Power | Utilization

Unconstrained Paths - NONE - NONE - Setup

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay
Path 1	∞	13	7	x[0]	dataout[7]	13.121	8.103	5.017
Path 2	∞	12	7	x[0]	dataout[6]	13.006	8.235	4.770
Path 3	∞	12	7	x[0]	dataout[5]	12.982	7.905	5.076
Path 4	∞	12	7	x[0]	dataout[4]	12.865	7.788	5.076
Path 5	∞	11	7	x[0]	dataout[3]	12.484	7.407	5.076
Path 6	∞	12	7	x[0]	dataout[2]	11.983	7.452	4.530
Path 7	∞	12	7	x[0]	dataout[1]	11.319	7.068	4.250
Path 8	∞	8	4	u3/q_reg[0]/C	dataout[0]	8.220	5.432	2.789
Path 9	∞	4	7	x[0]	u2/q_reg[6]/D	3.155	2.022	1.133
Path 10	∞	4	7	x[0]	u2/q_reg[7]/D	3.063	1.930	1.133

Figure 4.6: Delay of Normal adder.

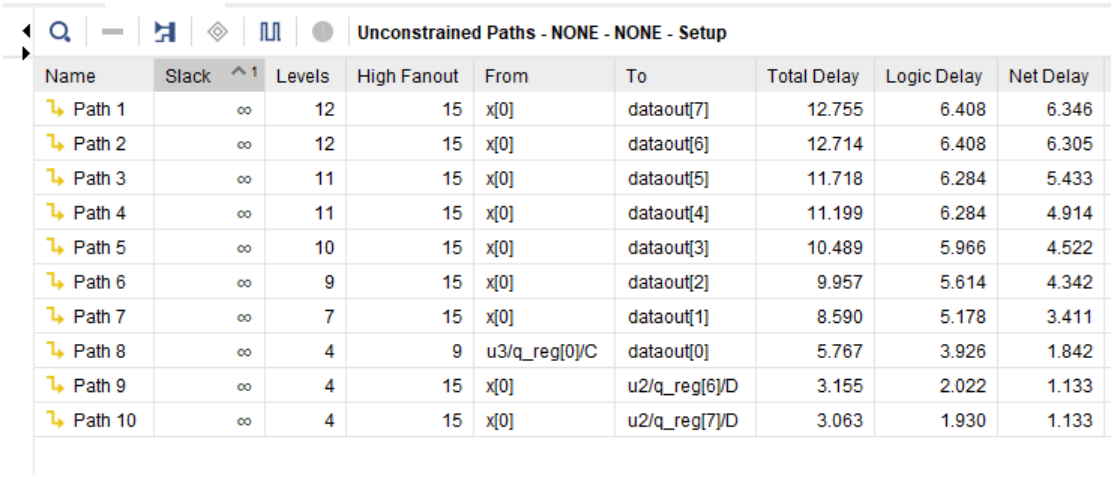


Runs | Timing | Power | Utilization

Unconstrained Paths - NONE - NONE - Setup

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay
Path 1	∞	10	14	x[0]	dataout[7]	12.372	5.921	6.450
Path 2	∞	10	14	x[0]	dataout[6]	12.124	5.856	6.267
Path 3	∞	9	14	x[0]	dataout[5]	11.098	5.732	5.365
Path 4	∞	9	14	x[0]	dataout[4]	10.852	5.797	5.054
Path 5	∞	9	14	x[0]	dataout[2]	9.771	5.426	4.344
Path 6	∞	8	14	x[0]	dataout[3]	9.625	5.296	4.328
Path 7	∞	7	14	x[0]	dataout[1]	8.583	5.178	3.404
Path 8	∞	4	9	u3/q_reg[0]/C	dataout[0]	5.767	3.926	1.842
Path 9	∞	4	14	x[0]	u2/q_reg[6]/D	3.155	2.022	1.133
Path 10	∞	4	14	x[0]	u2/q_reg[7]/D	3.063	1.930	1.133

Figure 4.7: Delay of Han-Carlson adder.



Unconstrained Paths - NONE - NONE - Setup

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay
Path 1	∞	12	15	x[0]	dataout[7]	12.755	6.408	6.346
Path 2	∞	12	15	x[0]	dataout[6]	12.714	6.408	6.305
Path 3	∞	11	15	x[0]	dataout[5]	11.718	6.284	5.433
Path 4	∞	11	15	x[0]	dataout[4]	11.199	6.284	4.914
Path 5	∞	10	15	x[0]	dataout[3]	10.489	5.966	4.522
Path 6	∞	9	15	x[0]	dataout[2]	9.957	5.614	4.342
Path 7	∞	7	15	x[0]	dataout[1]	8.590	5.178	3.411
Path 8	∞	4	9	u3/q_reg[0]/C	dataout[0]	5.767	3.926	1.842
Path 9	∞	4	15	x[0]	u2/q_reg[6]/D	3.155	2.022	1.133
Path 10	∞	4	15	x[0]	u2/q_reg[7]/D	3.063	1.930	1.133

Figure 4.8: Delay of Carry lookahead adder.

4.2.2 Power Report

Dynamic Power Consumption: This refers to the power consumed by the FPGA as a result of dynamic switching activities within the circuit. It includes the power consumed by the internal logic as it processes data and transitions between states.

Signal Power: Signal power refers to the power consumed by signals as they propagate through the design. It includes power dissipation due to switching activity on nets and wires. Signal power is influenced by factors such as signal frequency, capacitance, and load.

Logic Power: Logic power refers to the power consumed by the combinational and

sequential logic elements in the design. It includes power dissipation due to switching activity in logic gates and flip-flops.

I/O Power: I/O power refers to the power consumed by input and output buffers in the design. It includes power dissipation due to driving capacitive loads on I/O pins. I/O power consumption is influenced by factors such as I/O voltage levels, drive strength, and switching frequency.

Static Power Consumption: This is the power consumed by the FPGA when it is in a static state, i.e., when the logic is not actively switching. Static power is often associated with leakage current and can become a significant portion of total power consumption in modern semiconductor devices.

Total Power Consumption: This is the sum of dynamic and static power and represents the overall power consumption of the FPGA. To focus on the areas that contribute the most to the overall power budget.

For example, optimising the I/O power could involve strategies like reducing the number of active I/Os or adjusting the signal characteristics. Similarly, optimising dynamic power might involve techniques such as clock gating, reducing unnecessary transitions, or exploring power-efficient coding styles.Remember, these percentages are indicative of the power distribution in design and should be used as a guide for further optimisation. It's always a good practice to use power estimation tools during the design process to refine and improve the power characteristics of the FPGA.

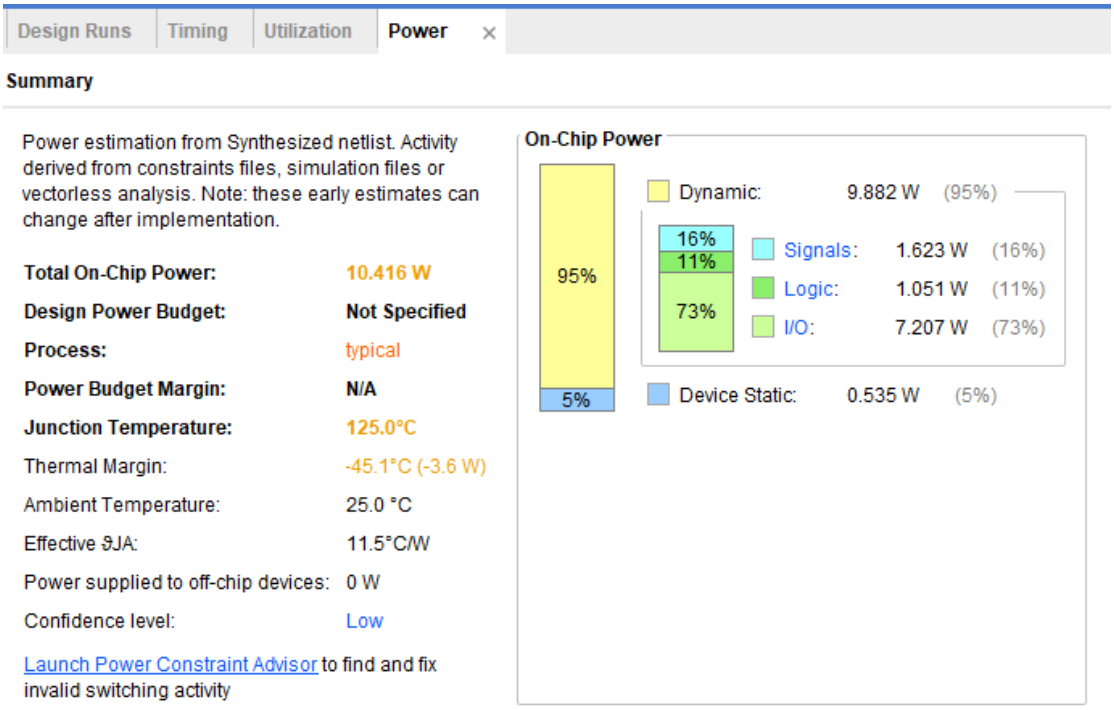


Figure 4.9: Power report of Ladner-fischer adder.

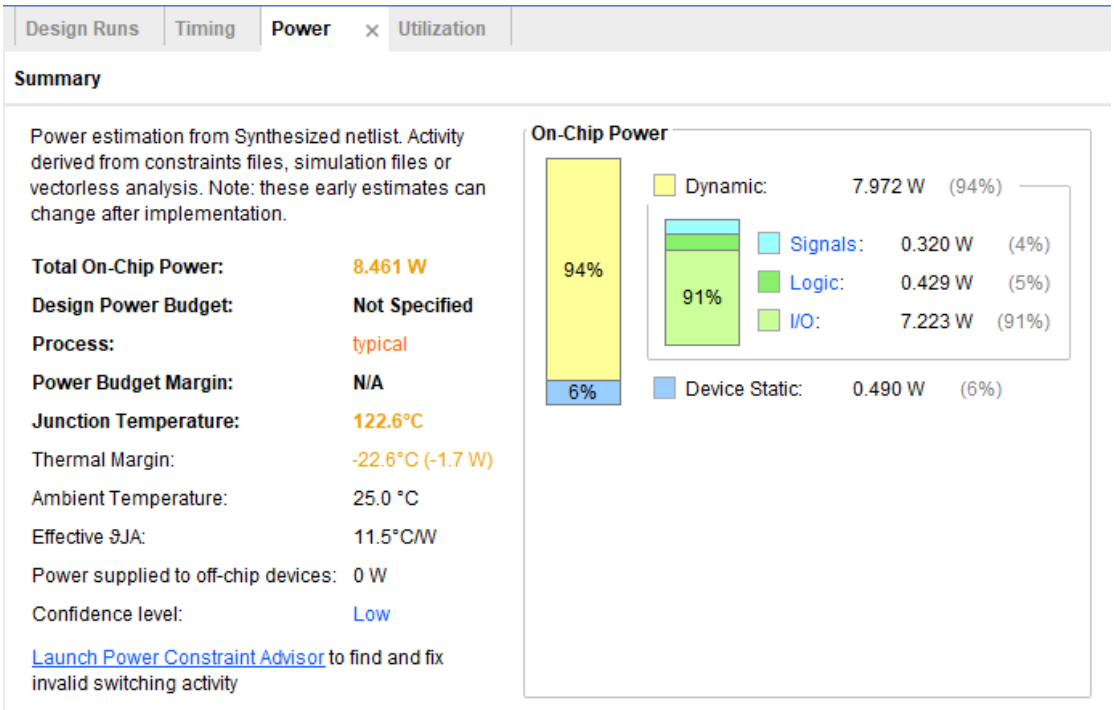


Figure 4.10: Power report of Normal adder.

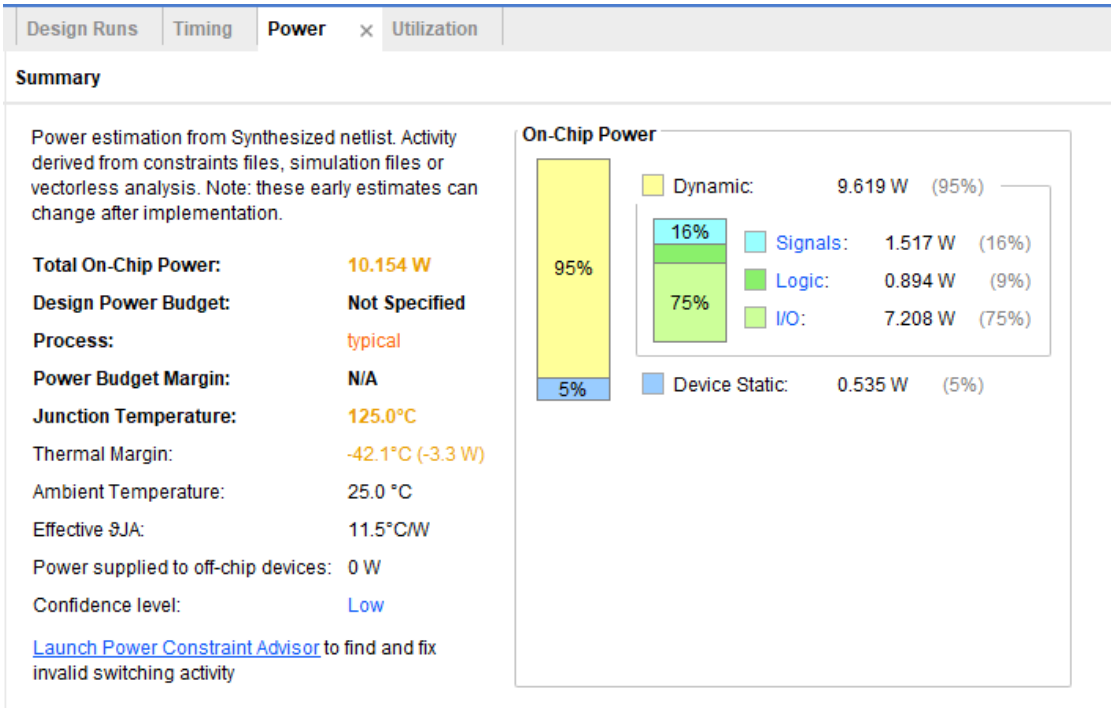


Figure 4.11: Power report of Han-Carlson adder.

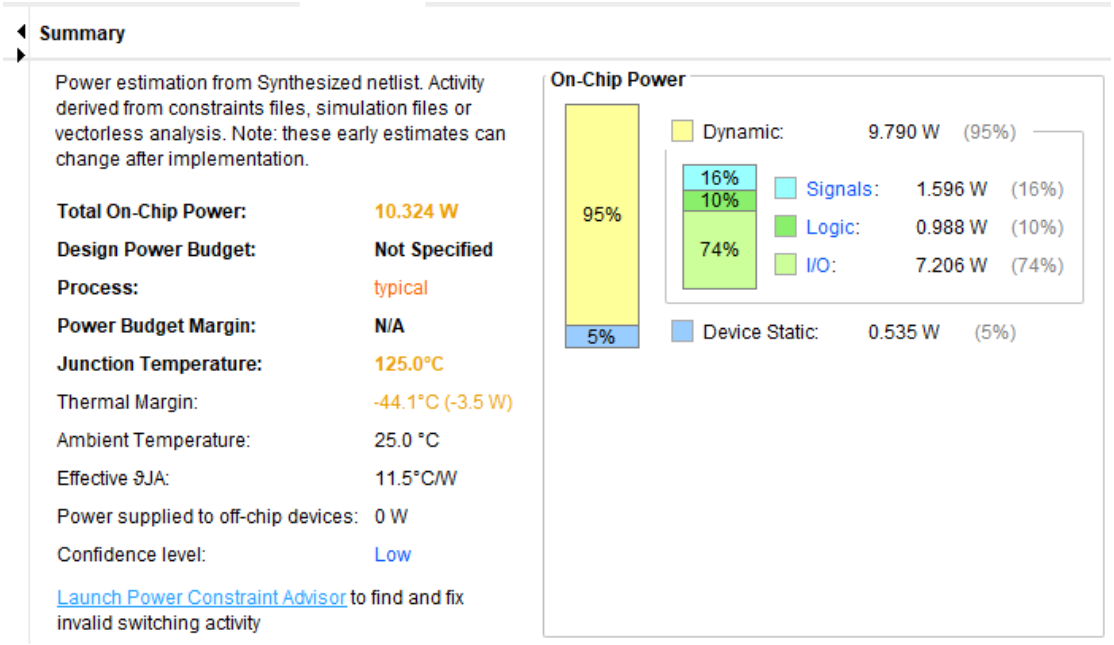


Figure 4.12: Power report of Carry look-ahead adder.

4.2.3 Utilisation

Utilisation refers to how the hardware resources of an FPGA (Field-Programmable Gate Array) or ASIC (Application-Specific Integrated Circuit) are allocated and utilised by a particular design implemented on the device. Understanding resource utilisation is

crucial for optimising designs for factors such as performance, area, and power consumption.

LUTs (Look-Up Tables):LUTs are the basic building blocks of FPGA logic. They are used to implement combinational logic functions. Utilisation is reported in terms of the total number of LUTs used and often broken down into specific types (e.g., LUT6, LUT5) based on the size of the LUTs.

Registers (Flip-Flops): Registers are used to store state information and implement sequential logic. Utilisation is reported in terms of the total number of registers used.

I/O Resources: I/O resources include input and output pins used to interface with external devices. Utilisation is reported in terms of the total number of I/O pins used and may include information about specific I/O standards and constraints.

Utilisation reports provide valuable insights into how effectively a design utilises the available resources of the FPGA or ASIC. Designers can use this information to optimise their designs for various factors, such as reducing logic complexity, minimising register usage, or maximising performance while staying within resource constraints imposed by the target device.

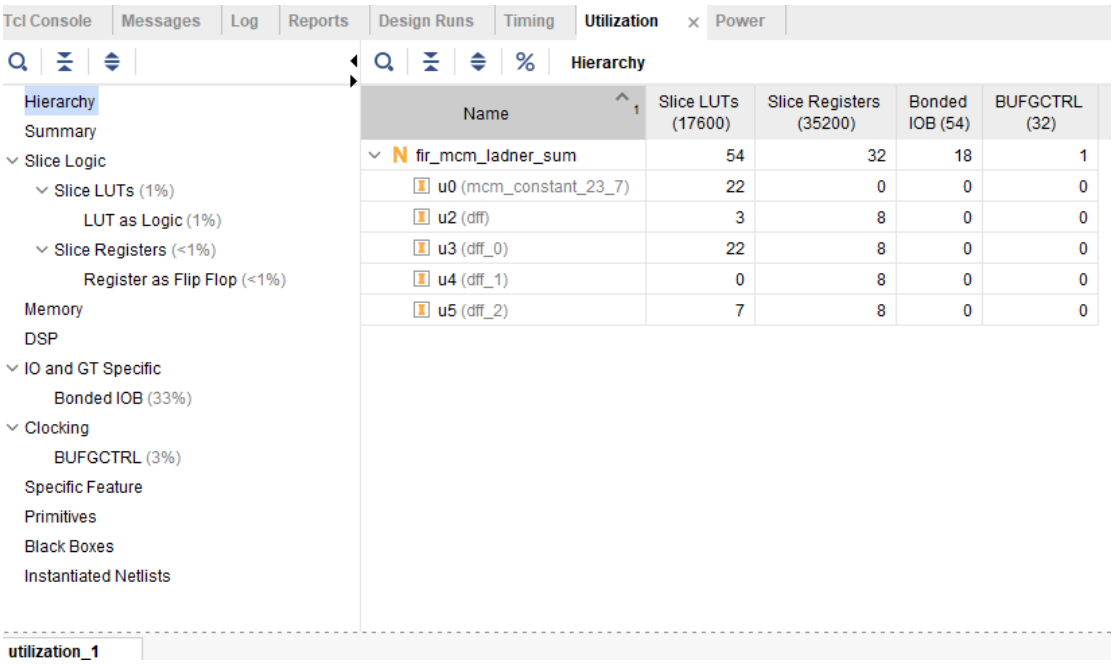


Figure 4.13: Ladner-fischer adder area utilisation.

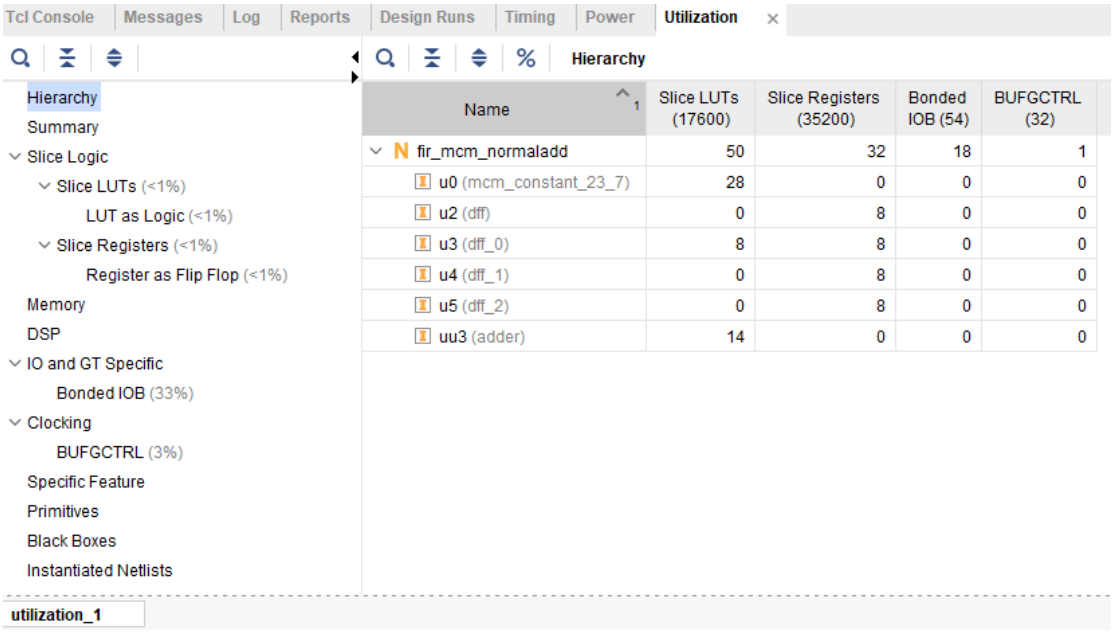


Figure 4.14: Normal adder area utilisation.

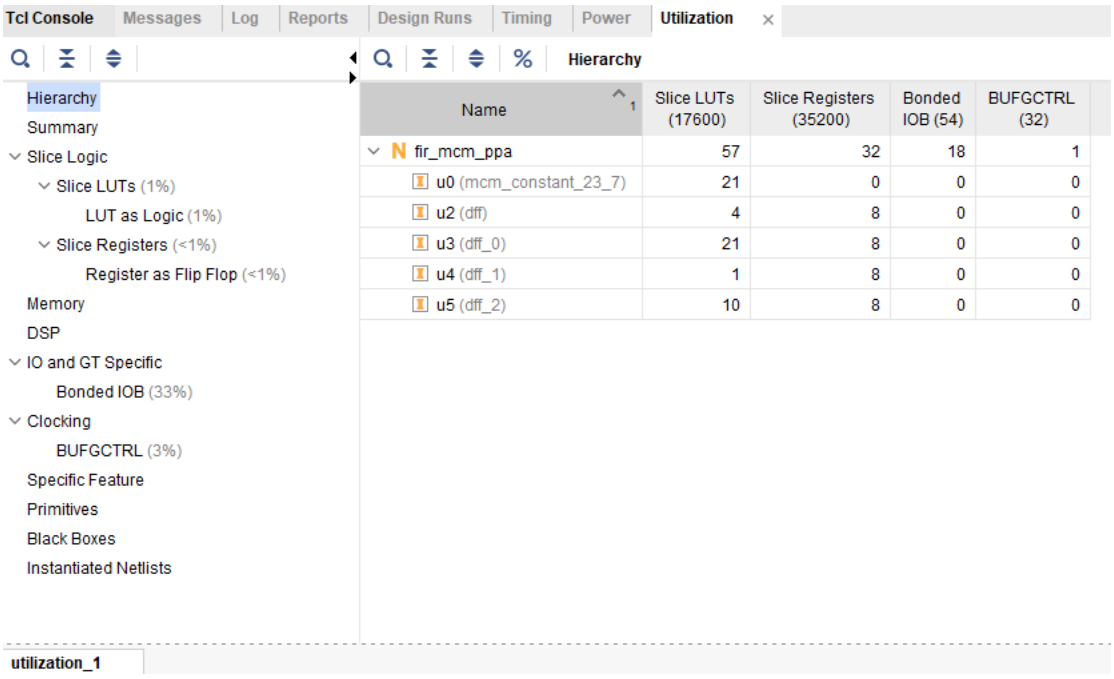


Figure 4.15: Han-Carlson adder area utilisation.

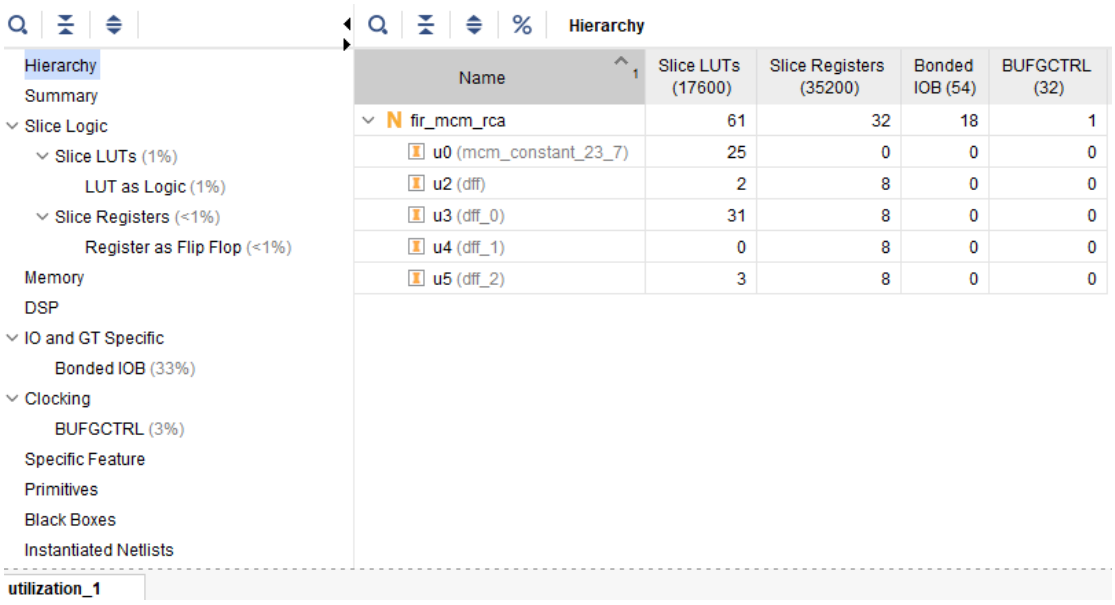


Figure 4.16: Carry look-ahead adder area utilisation.

4.2.4 Schematics

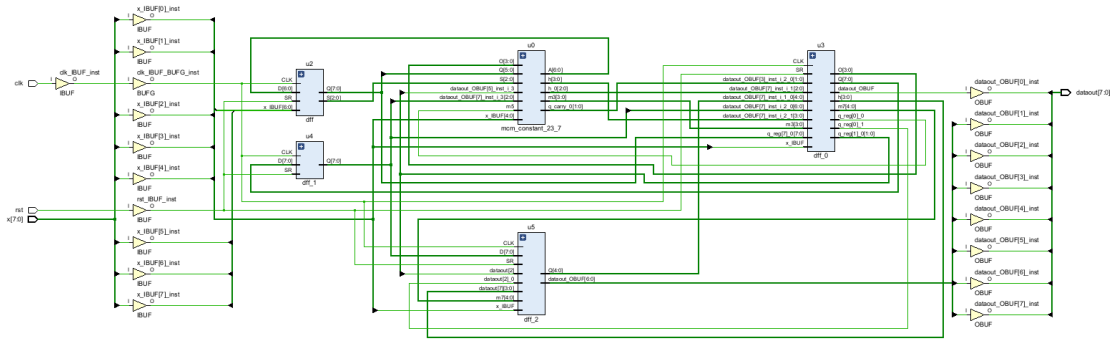


Figure 4.17: Technology Schematics of ladner-fischer adder.

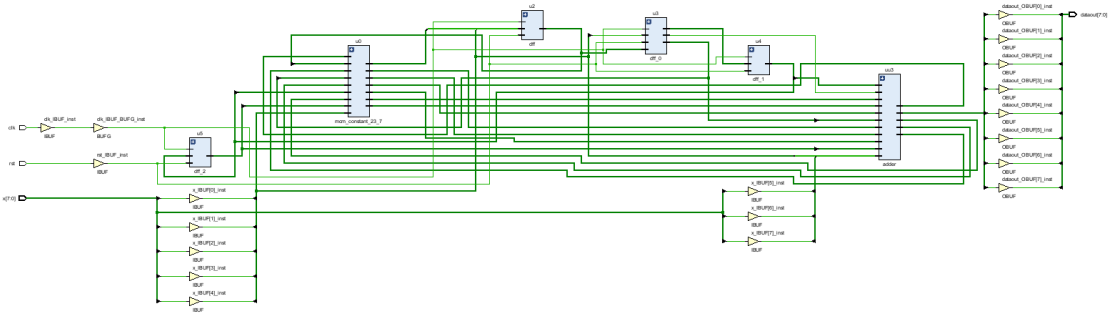


Figure 4.18: Technology Schematics of Normal adder.

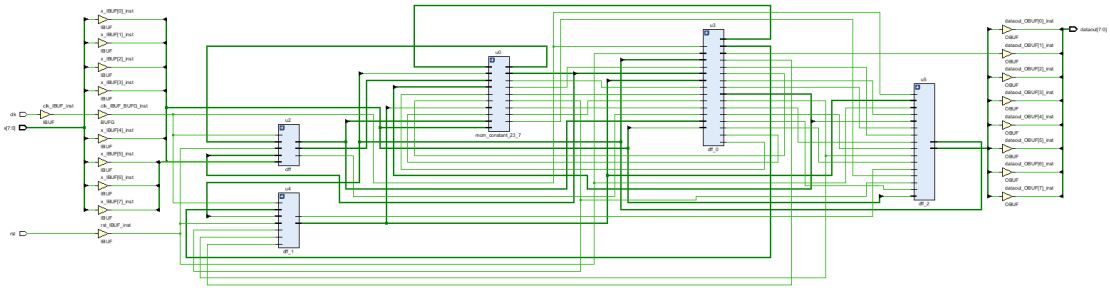


Figure 4.19: Technology Schematics of Han-Carlson adder.

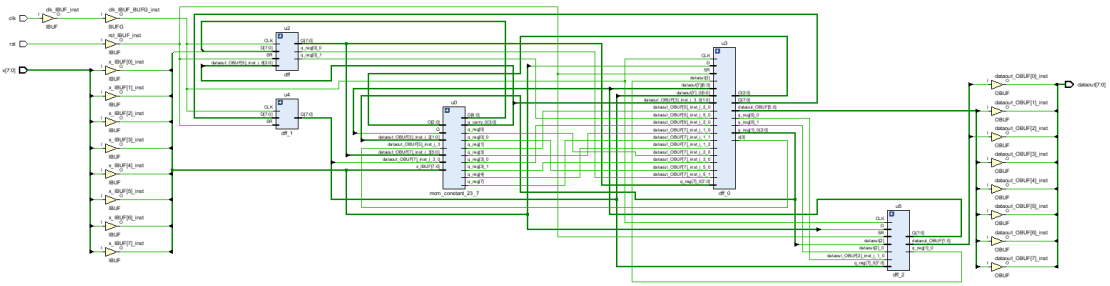


Figure 4.20: Technology Schematics of carry lookahead adder.

Chapter 5

Results and Discussion

5.1 Results

Table 5.1: Adder Performance Comparison

Adder	Delay	Power	Utilisation(LUTs)	IOBs
Normal Adder	13.121 (Setup) 0.282 (Hold)	8.461W	50/17600	18/54
Han-Carlson	12.372 (Setup) 0.282 (Hold)	10.154W	57/17600	18/54
CLA	12.755 (Setup) 0.282 (Hold)	10.324W	61/17600	18/54
Lander-Fischer	11.705 (Setup) 0.282 (Hold)	10.416W	54/17600	18/54

The Ladner-Fischer Adder emerges as a compelling choice for optimizing timing performance, particularly when compared to conventional approaches like the Carry Lookahead Adder (CLA) and the Parallel Prefix Adder (PPA) in conjunction with Multiple Constant Multiplication (MCM). Despite exhibiting slightly higher power consumption than PPA, Ladner-Fischer's efficiency in reducing delay makes it a favorable option, especially in scenarios where timing is critical.

In comparative studies, Ladner-Fischer Adder consistently demonstrates superior timing characteristics, effectively minimizing propagation delay compared to both CLA and PPA. This reduction in delay translates to enhanced overall performance in digital circuits, particularly in applications requiring rapid signal processing or real-time oper-

ations. Ladner-Fischer Adder also exhibits higher utilization of Look-Up Tables (LUTs) compared to both Parallel Prefix Adder (PPA) and Ripple Carry Adder (RCA). Despite this increased LUT utilization, Ladner-Fischer's efficiency in reducing delay remains its primary advantage, particularly in applications where meeting strict timing requirements is paramount. This higher utilization of LUTs underscores Ladner-Fischer's more complex implementation compared to PPA and RCA but is justified by its superior timing performance and efficiency in handling Multiple Constant Multiplication (MCM). Therefore, Ladner-Fischer Adder represents a holistic solution for applications where minimizing delay and optimizing timing performance are critical considerations, even if it entails slightly higher resource utilization in terms of LUTs.

Moreover, Ladner-Fischer's efficiency in handling Multiple Constant Multiplication (MCM) further underscores its suitability for time-sensitive tasks. While it may incur slightly higher power consumption than PPA, Ladner-Fischer's ability to optimize timing performance outweighs this drawback, particularly in scenarios where meeting strict timing requirements is paramount.

In summary, Ladner-Fischer Adder emerges as a compelling choice for applications where minimizing delay and optimizing timing performance are critical considerations. Despite marginally higher power consumption compared to PPA, Ladner-Fischer's efficiency in reducing delay makes it a preferred option in scenarios where timing matters most.

5.2 Advantages

Hardware Co-Design: Integrating hardware co-design methodologies can enhance the overall efficiency and performance of FIR filters. By jointly optimizing hardware architecture and algorithm design, it's possible to achieve synergistic improvements in terms of speed, power consumption, and resource utilization.

Parallel Processing: Investigating parallel processing architectures for FIR filters can enable the implementation of high-throughput filter designs. Parallelism can be leveraged at various levels, including filter stages, coefficient processing, and data handling,

to maximize processing speed and efficiency.

Energy-Efficient Implementations: Addressing the energy efficiency of FIR filter implementations is crucial for battery-powered and energy-constrained devices. Future research can focus on developing energy-efficient architectures, low-power design techniques, and dynamic power management strategies to minimize power consumption without compromising performance.

Chapter 6

Conclusion with Future Scope

6.1 Conclusion

In conclusion, the utilization of efficient adders, such as the Ladner-Fischer Adder, combined with Multiple Constant Multiplication (MCM) techniques, presents a promising approach to enhancing the performance of digital circuits, particularly in Finite Impulse Response (FIR) filters. Through rigorous comparative analysis, it's evident that Ladner-Fischer Adder offers superior timing performance compared to conventional adders like Parallel Prefix Adder (PPA) and Ripple Carry Adder (RCA), despite its slightly higher utilization of Look-Up Tables (LUTs). This highlights the importance of optimizing circuit architecture to achieve optimal trade-offs between resource utilization and timing efficiency.

6.2 Future scope

Integrating FIR filters with customized hardware platforms for MCM operations involves several steps aimed at optimizing signal processing and multiplication tasks in communication systems. Firstly, FIR filters are designed and implemented using digital signal processing techniques to meet specific requirements such as signal bandwidth, frequency response, and noise rejection. These filters are then integrated into the hardware platform alongside MCM modules, forming a cohesive system for signal conditioning and multiplication.

In practice, the FIR filters preprocess input data streams before they undergo mul-

tiplication operations, enhancing signal quality and reliability. This preprocessing may involve filtering out noise, shaping input signals, and removing unwanted frequency components to improve the accuracy of subsequent multiplication results. Additionally, FIR filters can be configured to adapt dynamically to changing signal conditions, adjusting their characteristics in real-time to accommodate variations in data rate and signal environment.They play a crucial role in ensuring the integrity and efficiency of data transmission across different communication standards such as Wi-Fi, LTE, and 5G, as well as in satellite communication, radio frequency identification (RFID), and software-defined radio (SDR) applications.

Bibliography

[1] Mathias Faust, Oscar Gustafsson, and Chip-Hong Chang. Reconfigurable multiple constant multiplication using minimum adder depth. pages 1297–1301, 2010.

[2] Rémi Garcia and Anastasia Volkova. Toward the multiple constant multiplication at minimal hardware cost. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 70(5):1976–1988, 2023.

[3] Garima Thakur, Harsh Sohal, and Shruti Jain. Design and analysis of high-speed parallel prefix adder for digital circuit design applications. pages 095–100, 07 2020.

[4] Bhavani Koyada, N. Meghana, Md. Omair Jaleel, and Praneet Raj Jeripotula. A comparative study on adders. In *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pages 2226–2230, 2017.

[5] Martin Kumm, Anastasia Volkova, and Silviu-Ioan Filip. Design of optimal multiplierless fir filters with minimal number of adders. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(2):658–671, 2023.