IMPLEMENTATION OF TROJAN AWARE FIR FILTER

A Project report submitted in partial fulfilment of the requirements for

the award of the degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted by

K. Divya Teja Sri (319126512155) M. Kalyan Karthik (319126512161)

Y. Phanindra (319126512158) A. Priyanka (319126512130)

R. Paavan Sri Sai (319126512178)

Under the guidance of

Dr. SRINIVAS SABBAVARAPU

M. Tech, Ph. D(IITH)

Associate Professor

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES

(UGC AUTONOMOUS)

(*Permanently Affiliated to AU, Approved by AICTE and Accredited by NBA & NAAC*)

Sangivalasa, Bheemili mandal, Visakhapatnam dist. (A.P)

2022-2023

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES
(UGC AUTONOMOUS)
(Permanently Affiliated to AU, Approved by AICTE and Accredited by NBA & NAAC)
Sangivalasa, Bheemili mandal, Visakhapatnam dist. (A.P)

PRAGNANAM BRAHMA
ANITS

CERTIFICATE

This is to certify that the project report entitled **"Implementation of Trojan Aware FIR Filter"** *submitted by* **K. Divya Teja Sri (319126512155)** in partial fulfilment of the requirements for the award of the degree of Bachelor **of Technology** in **Electronics & Communication Engineering** of Anil Neerukonda Institute of Technology and Sciences(A), Visakhapatnam is a record of bonafide work carried out under my guidance and supervision.

Project Guide

Dr. SRINIVAS SABBAVARAPU
M. Tech, Ph.D (IITH),
Associate Professor
Department of E.C. E
ANITS

Associate Professor
Department of E.C.E.
Anil Neerukonda
Institute of Technology & Sciences
Sangivalasa, Visakhapatnam-531 162

Head of the Department

Dr. B. JAGADEESH
Professor & HOD
Department of E.C.E
ANITS

Head of the Department
Department of E C E
Anil Neerukonda Institute of Technology & Sciences
Sangivalasa - 531 162

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our project guide **Dr. Srinivas Sabbavarapu**, M. Tech, Ph. D (IITH), Associate Professor Department of Electronics and Communication Engineering, ANITS, for his/her guidance with unsurpassed knowledge and immense encouragement. We are grateful to **Dr. Jagadeesh**, Head of the Department, Electronics and Communication Engineering, for providing us with the required facilities for the completion of the project work.

We are very much thankful to the **Principal and Management, ANITS, Sangivalasa,** for their encouragement and cooperation to carry out this work.

We express our thanks to all **teaching faculty** of Department of ECE, whose suggestions during reviews helped us in accomplishment of our project. We would like to thank **all non-teaching staff** of the Department of ECE, ANITS for providing great assistance in accomplishment of our project.

We would like to thank our parents, friends, and classmates for their encouragement throughout our project period. At last, but not the least, we thank everyone for supporting us directly or indirectly in completing this project successfully.

PROJECT STUDENTS

K. DIVYA TEJA SRI (319126512155)

M. KALYAN KARTHIK (319126512161)

Y. PHANINDRA (319126512158)

A. PRIYANKA (319126512130)

R. PAAVAN SRI SAI (319126512178)

CONTENTS

ABSTRACT

In numerous real-time applications, including communications, signal processing, and consumer electronics, Finite Impulse Response (FIR) filters are employed. Because of its inherent stability and linear phase, FIR filters are widely sought-after in the construction of digital filters. They benefit from being time-invariant and are simple to build with little computational requirements. Therefore, hardware security of FIR Filter is very essential for good performance and to produce perfect results. But there is a possibility of hardware threats like tampering, reverse engineering, hardware trojan etc as the design of filter involves many stages. As a result of these hardware attacks on the FIR filter, there will be several problems like performance degradation, leakage of confidential information, lack of stability etc. In this project, a trojan aware FIR filter will be designed by using Physical Unclonable Functions (PUFs) so that we can get the secured FIR filter with high performance and security.

APPROACH: This project aims to concentrate on the design of a trojan aware FIR filter by using Physical Unclonable Functions (PUFs). FIR filter coefficients will be generated by using Physical Unclonable Function (PUF) and then the generated coefficients from the PUF will be given to the designed FIR filter using Hardware Description Language (HDL).

Key Words: FIR Filters, Hardware Trojan, Physical Unclonable Function (PUF), Hardware Description Language (HDL)

LIST OF FIGURES

LIST OF EQUATIONS

CHAPTER 1

INTRODUCTION

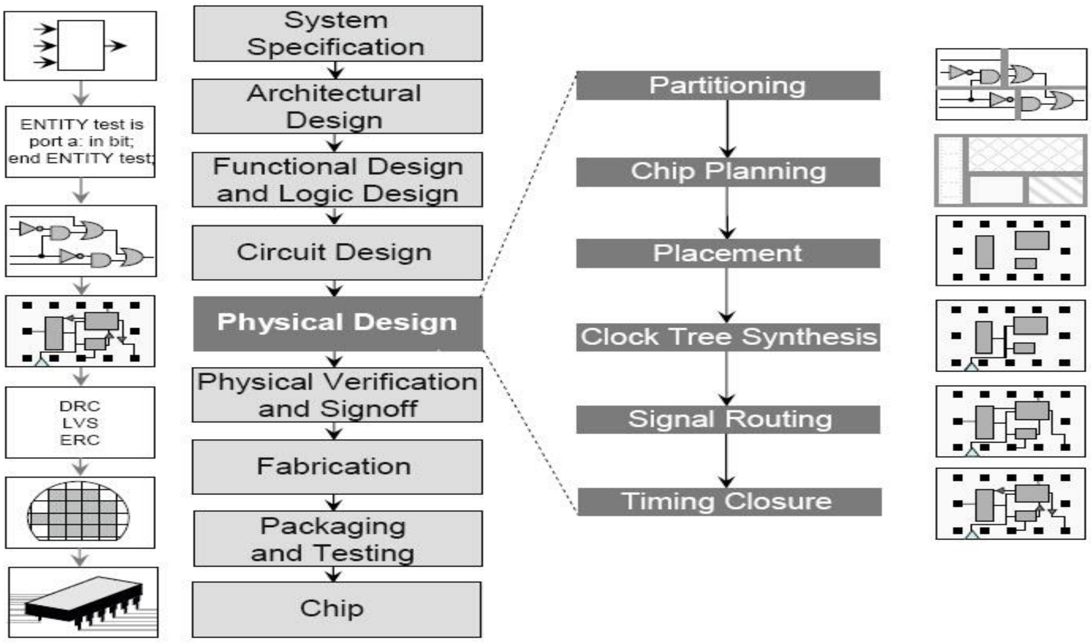
1.1 VLSI TECHNOLOGY

1.1.1 Introduction

The method of building an integrated circuit (IC) by fitting thousands of transistors onto a single chip is known as very large-scale integration (VLSI). Prior to the development of VLSI technology, the majority of integrated circuits (ICs) could only carry out a very small number of tasks. Among other glue logic, an electrical circuit may have a CPU, RAM, and ROM. All of these can be combined into a single chip because of VLSI.

1.1.2 VLSI Design Flow

There are several steps involved in designing VLSI IC circuits. The design requirements are the initial step in those processes; they provide an abstract description of the functionality, interface, and architecture of the digital IC circuit that needs to be created. The design is subsequently analyzed in terms of functionality, performance, compliance with established standards, and other specifications using behavioral descriptions. Using Hardware Description Languages, Register Transfer Level (RTL) description is performed. (HDLs). To evaluate functionality, this RTL description has been emulated. The RTL description is then translated into a gate-level netlist utilizing logic synthesis tools using Electronic Design Automation (EDA) tools. A gate level netlist is a description of the circuit's gates and connections in terms of how well they adhere to timing, power, and area requirements. The physical layout is then created, confirmed, and delivered to manufacture.



Courtesy: https://lh3.googleusercontent.com/-bjVA7_Y_Fyg/TXc_D0Zs77I/AAAAAAAAAK8/Gqb1_2VCcpc/w1200-h630-p-k-no-nu/VLSI+Design+Flow.bmp

Fig-1: VLSI Design Flow

1.1.3 Advantages

Here are some advantages of VLSI technology

- ***High packing density:*** VLSI technology allows for the integration of a large number of electronic components on a single chip. This high packing density results in smaller and more compact electronic devices.
- ***Low power consumption:*** VLSI technology allows for the design of electronic circuits that consume less power. This is achieved by using advanced fabrication techniques, such as CMOS (Complementary Metal-Oxide-Semiconductor) technology, which reduces power consumption by minimizing leakage current.
- ***High reliability:*** VLSI technology allows for the design of highly reliable electronic circuits. This is achieved by fault-tolerant design techniques, such as redundancy and error correction codes.
- ***High speed:*** VLSI technology allows for the design of electronic circuits that operate at very high speeds. This is achieved by advanced fabrication techniques, such as reduced feature sizes and interconnect lengths, and high-performance circuit design techniques.
- ***Flexibility:*** VLSI technology allows for the design of electronic circuits that are highly flexible and can be reconfigured for different applications. This is achieved by programmable logic devices, such as FPGAs (Field-Programmable Gate Arrays), which can be programmed to perform different functions.
- ***Cost-effective:*** VLSI technology has led to lower manufacturing costs for electronic systems. The integration of many transistors on a single chip reduces the need for external components, such as resistors, capacitors, and inductors. This has led to a reduction in the cost of electronic systems, making them more affordable and accessible.
- ***Small Size:*** The integration of many transistors on a single chip also means that the resulting electronic system can be made very compact. This makes VLSI technology ideal for applications that require small size and weight, such as mobile phones, laptops, and other portable devices.

1.1.4 Limitations

- ***Heat Dissipation:*** As the number of transistors on a chip increases, so does the amount of heat generated. The heat dissipation problem is exacerbated by the shrinking of transistor size, which limits the amount of space available for heat dissipation. This can lead to thermal management issues and reduced reliability.
- ***Fabrication Yield:*** The fabrication process for VLSI chips is highly complex and prone to defects, which can result in a low yield rate. The yield rate is the percentage of working chips per wafer, and it is a critical factor that affects the cost of production. Defects can be caused by many factors, including process variations, equipment malfunctions, and human errors.

- ***Environmental Concerns:*** VLSI chip manufacturing produces a lot of trash and necessitates the use of dangerous chemicals. If not handled appropriately, this could harm the environment and people's health.
- ***Obsolescence:*** VLSI chips are subject to obsolescence, as technology advances rapidly. This means that a chip that is state-of-the-art today may become outdated in a few years, making it necessary to replace it with a newer version. This can be a significant problem for applications that require long-term reliability and availability

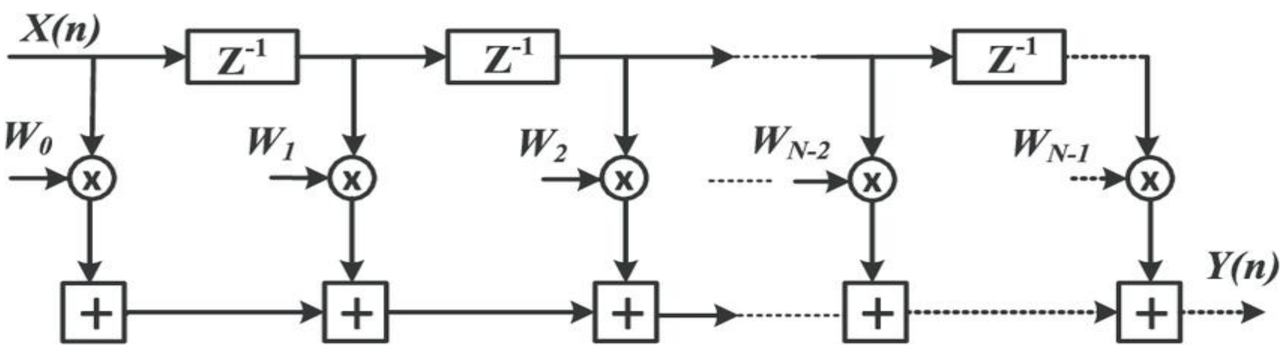
1.1.5 Role of FIR filter in VLSI

Digital signal processing (DSP) systems frequently use Finite Impulse Response (FIR) filters, which are also widely used in VLSI design. FIR filters are digital filters that convolutes a digital signal using a finite number of coefficients. Dedicated hardware, such as application-specific integrated circuits (ASICs) or field-programmable gate arrays, can be used to construct FIR filters in VLSI designs. (FPGAs). In comparison to other filter types, FIR filters provide a number of benefits, such as linear phase response, minimal sensitivity to coefficient quantization, and effective hardware implementation.

1.2 FIR FILTERS

1.2.1 Introduction

A Finite Impulse Response (FIR) filter is a digital filter whose impulse response is of finite duration, as it settles to zero in finite time. These FIR filters are different from Infinite Impulse response (IIR) filters, which may have internal feedback and continues to respond indefinitely. The impulse response of an Nth order discrete time FIR filter lasts exactly N+1 samples before it settles to zero. FIR filters can be continuous time or discrete time, and digital or analog. A Finite Impulse Response (FIR) filter is an inherently stable filter as it is a filter with no feedback. FIR filters have an exactly linear phase response.



Courtesy: https://www.researchgate.net/profile/Syed-Manzoor-Qasim/publication/262767435/figure/fig2/AS:392396354408448@1470565989493/Architecture-of-direct-form-FIR-filter.png

Fig-2: Block diagram of FIR Filters

A general FIR filter equation can be expressed as,

$$\boldsymbol{y(n) = \sum_{k=0}^{N-1} wk. x(n-k)} \qquad \text{..........} \textbf{(1)}$$

1.2.2 IIR Filters

Infinite Impulse Response filters, are a type of digital filters commonly used in signal processing. IIR filter is a digital filter whose impulse response is of infinite duration. These filters have internal feedback, which means that the output signal depends not only on the input signal but also on previous outputs. It is a recursive filter as it has feedback loop. IIR are not linear phase filter.

1.2.3 FIR and IIR Comparison

FIR (Finite Impulse Response) filters and IIR (Infinite Impulse Response) filters are two common types of digital filters used in signal processing. Here are some key differences between them:

Stability: FIR filters are always stable, while IIR filters may be unstable if their coefficients are not properly designed. This is because IIR filters have feedback, while FIR filters do not.

Phase response: FIR filters have a linear phase response, which means that they can preserve the phase relationships between different frequency components of a signal. IIR filters, on the other hand, typically introduce phase distortion in the filtered signal.

Filter order: FIR filters generally require a higher order (more coefficients) than IIR filters to achieve the same level of filtering. This makes FIR filters more computationally expensive.

Frequency response: IIR filters can achieve a sharper cut-off in the frequency domain with fewer coefficients than FIR filters. This is because IIR filters have feedback, which allows for more complex frequency response shapes.

Transient response: FIR filters have a finite impulse response, which means that they settle to zero in a finite amount of time. IIR filters, on the other hand, have an infinite impulse response, which means that they can have a non-zero response even after the input signal has ceased.

1.2.4 Working Principle:

A digital Finite Impulse Response (FIR) filter is a filter which operates on discrete-time signals. FIR filter is based on a mathematical operation known as “convolution”. The basic idea behind an FIR filter is to apply a set of coefficients, also known as taps, to the input signal to produce an output signal. The input signal is convolved with the impulse response of the filter, which is defined by the set of coefficients. The output signal is the sum of the products of the input signal samples and the corresponding filter coefficients.

The process of filtering an input signal using FIR filter can be divided into 4 steps:

1.Input Signal: The input is the digital signal that needs to be filtered. It is typically represented as a sequence of numbers.

2.Filter Coefficients: The filter coefficients are the weights that are applied to the input signa to produce the output signal. The number of filter coefficients depends on the filter order and the desired frequency response.

3.Convolution: The input signal is convolved with the impulse response of the filter, which is defined by the filter coefficients. This involves multiplying each sample of the input signal by the corresponding filter coefficient and summing the products over a certain number of samples.

4.Output Signal: The output signal is the result of convolution operation. It represents the filtered version of the input signal and can be used for further processing or analysis.

An analog FIR filter function by suppressing or blocking given frequency components of input signal and passing the original signal minus these suppressed components as the output, while a digital FIR filter works by performing mathematical operations on the input signal. The preliminary function of a digital filter is analog to digital conversion. The input signal will be converted into an 8-bit digital signal through the A/D (Analog to Digital Converter) device. Generally, a high-speed successive approximation A/D converter (SAR ADC) can be used. Regardless of the multiplication and accumulation method or the distributed algorithm to design the FIR filter, the data output by the filter is a series of sequences.

1.2.5 Advantages

Here are some advantages of FIR filters:

- ***Linear Phase:*** FIR filters have a linear phase response, which means that they can maintain the shape of the input signal while filtering it. This makes them particularly useful in applications where the phase distortion of the signal cannot be tolerated.
- ***Stability:*** FIR filters are inherently stable because they have no feedback. This means that they can be designed to have a desired frequency response without the risk of instability.
- ***Easy to design:*** FIR filters are relatively easy to design and implement because they are based on simple convolution operations. This makes them a good choice for applications where real-time processing is required.
- ***Variable frequency response:*** The frequency response of an FIR filter can be easily adjusted by modifying the filter coefficients. This makes them versatile and allows them to be used in a wide range of applications.

- ***Nonlinear distortion-free:*** As they have a linear response, FIR filters do not suffer from nonlinear distortion. This means that they can accurately filter signals without adding any unwanted harmonics or intermodulation products.
- ***Efficient implementation:*** FIR filters can be implemented using a variety of techniques, including direct convolution, FFT-based convolution, and polyphase decomposition. This allows them to be optimized for specific hardware platforms and signal processing applications.

1.2.6 Disadvantages

Along with the advantages, there are some disadvantages as well for FIR filters. Some of them are,

- ***Higher computational complexity:*** Compared to IIR (Infinite Impulse Response) filters, FIR filters require a higher number of filters taps to achieve a comparable frequency response. This can result in higher computational complexity and memory requirements, particularly for high-order filters.
- ***Longer delay:*** FIR filters have a longer delay than IIR filters, which can be a disadvantage in applications where real-time processing is required. The delay is proportional to the number of filter taps, and increasing the number of taps to improve the frequency response will increase the delay.
- ***Larger memory requirements:*** FIR filters require a larger memory to store the filter coefficients, particularly for high-order filters with many taps. This can be a disadvantage in applications with limited memory.
- ***Limited selectivity:*** FIR filters have a limited selectivity compared to IIR filters, particularly in the transition band. This means that they may require a higher number of taps to achieve a comparable roll-off rate.
- ***Limited flexibility:*** FIR filters have a fixed frequency response, and changing the frequency response requires designing a new filter with different coefficients. This can be a disadvantage in applications where the frequency response needs to be adjusted dynamically.
- ***Noncausal implementation:*** FIR filters are typically implemented using a noncausal convolution operation, which means that they require future samples to be processed. This can be a disadvantage in applications where causality is important, such as real-time processing

1.2.7 Applications

Finite Impulse Response (FIR) filters are widely used in various applications. Some of the common applications include:

- ***Audio and video processing:*** FIR filters are used in audio and video processing applications to remove noise, equalize the frequency response, and perform other signal conditioning operations. They are particularly useful in applications that require linear phase response and high quality signal processing.

- ***Digital Communication:*** FIR filters are used in digital communication systems to remove interference, perform equalization, and extract signal components. They are used in applications such as digital modulation, demodulation, and channel coding.
- ***Medical signal processing:*** FIR filters are used in medical signal processing applications to filter out noise and artifacts from biological signals, such as electrocardiograms (ECG), electroencephalograms (EEG), and medical images. They are also used in digital signal processing applications for hearing aids, cochlear implants, and other medical devices.
- ***Radar and sonar systems:*** FIR filters are used in radar and sonar systems to filter out unwanted signals and extract target signals. They are used in applications such as pulse compression, matched filtering, and Doppler filtering.
- ***Instrumentation and control:*** FIR filters are used in instrumentation and control applications to filter out noise and unwanted signals from sensor data. They are also used in feedback control systems to perform filtering and signal conditioning operations.

FIR filters are widely used in various applications. As the design of FIR filter includes various steps, there might be a possibility of trojan insertion in the filter design. A hardware trojan is a malicious modification of the circuitry, which is a piece of hardware embedded inside another large piece of hardware. If the hardware trojan is not in active state i.e., off state, it does not show any effect on the function of the circuit. But after the activation of hardware trojan i.e., on state, it does something malicious which is unpredictable for a user. So, hardware security of FIR filter is very essential for the good performance of the filter in required applications.

1.3 HARDWARE SECURITY

1.3.1 Introduction

Hardware Security is a vulnerability protection that comes in the form of a physical device rather than software that is installed on the hardware of a computer system. Hardware security is a mechanism used to secure the hardware components of an integrated circuit (IC) design against various threats as tampering, unauthorised access, counterfeiting or reverse engineering. VLSI hardware security is critical for ensuring the availability of sensitive data and functions stored on integrated circuit.

1.3.2 Types of Hardware Threats

Hardware threats in VLSI (Very Large-Scale Integration) can be broadly categorized into two types:

i. Physical attacks: These types of attacks involve the manipulation or alteration of the hardware at the physical level.

Examples of physical attacks include:

Tampering: This involves physically modifying the hardware to introduce a vulnerability. For example, an attacker may alter the design of a chip to add a backdoor that can be used to access the system.

Reverse engineering: This involves deconstructing a chip to understand its design and functionality. An attacker can use this knowledge to identify vulnerabilities and exploit them.

Side-channel attacks: These attacks exploit information leaked by the hardware during its operation. For example, an attacker may measure the power consumption of a chip to infer information about the operations being performed by the chip.

ii. Logical attacks: These types of attacks involve exploiting vulnerabilities in the design or functionality of the hardware.

Examples of logical attacks include:

Hardware Trojans: These are malicious circuits that are inserted into a chip to perform a specific task. For example, a Trojan may be designed to leak sensitive information or disrupt the functioning of the chip.

Malicious firmware: This involves modifying the firmware or software that controls the hardware to introduce vulnerabilities or enable unauthorized access.

Design vulnerabilities: These are flaws in the design of the chip that can be exploited by attackers. For example, a chip may have a weak encryption algorithm that can be easily broken.

1.3.3 Types of Hardware Security Techniques

Hardware security techniques are broadly divided into two types. They are:

I. Active techniques

II. Passive techniques

I. Active hardware security techniques involve measures that take active steps to protect the hardware from attacks. Some examples of active hardware security techniques include:

Tamper detection and response: Active hardware security techniques like tamper detection and response involve sensors or other components that actively monitor the hardware for signs of tampering. If tampering is detected, the hardware can take steps to prevent further damage, such as shutting down or erasing sensitive data.

Secure boot: Secure boot is an active hardware security technique that ensures that only trusted software is loaded during the boot process. This can prevent attacks such as rootkits and other types of malware that try to take control of the system at an early stage.

Physical unclonable functions (PUFs): PUFs are hardware circuits that generate unique identifiers based on random variations in the manufacturing process. These identifiers can be used to verify the authenticity of hardware components and prevent counterfeiting.

Side-channel analysis (SCA) countermeasures: SCA attacks exploit the unintended side-effects of a device's operation, such as power consumption or electromagnetic radiation, to extract sensitive information. SCA countermeasures include techniques such as masking, which adds noise to the device's operation to make it more difficult to detect useful information.

II. Passive hardware security techniques provide protection by design and do not require active intervention. Some examples of passive hardware security techniques include:

Obfuscation: Obfuscation is a technique that makes it difficult for an attacker to reverse engineer a device. This can involve techniques such as obscuring the circuit design, encrypting code, or hiding key components within the device.

Redundancy: Redundancy involves building in duplicate components or circuits that can take over if the primary circuit fails. This can prevent attacks that try to disable or bypass key components.

Design for security: Design for security involves building security into the hardware design from the ground up. This can involve techniques such as using secure programming languages, enforcing data access controls, and building in hardware-based encryption.

1.3.4 Techniques used for Hardware Security in VLSI

Obfuscation: This technique involves intentionally obscuring the logic or structure of a circuit to make it harder to reverse engineer or tamper with.

Randomization: This technique involves adding random variations to the design or manufacturing process to make it more difficult for an attacker to replicate the IC.

Hardware-based cryptography: This technique involves integrating cryptographic primitives, such as encryption or hashing, directly into the hardware of the IC.

Side-channel analysis resistance: This technique involves designing circuits to be resistant to side-channel attacks, which exploit information leaks from power consumption, electromagnetic emissions, or other sources.

Hardware Trojan detection and prevention: This technique involves identifying and mitigating the risk of malicious modifications to the IC, which could compromise its security or functionality.

Physical tamper resistance: This technique involves designing the IC to be resistant to physical tampering, such as by incorporating secure packaging or tamper-evident seals.

1.3.5 Use of PUFs in Hardware Security

PUFs (Physical Unclonable Functions) are a class of hardware security primitives that exploit the inherent variations in the physical properties of integrated circuits to generate unique and unpredictable responses to challenge inputs. PUFs offer a promising approach to hardware security, as they provide a means of generating unique and unpredictable responses that are resistant to many types of attacks. PUFs are unclonable, which means it is difficult to replicate PUFs. A PUF can be integrated into a circuit during the manufacturing process, and the unique response generated by the PUF can be used to verify the authenticity of it. PUFs can be integrated into a wide range of electronic devices, making them suitable for many different security applications.

1.4 PHYSICAL UNCLONABLE FUNCTIONS (PUFs)

1.4.1 Introduction

A Physically Unclonable Function (PUF) is a hardware security feature that generates a unique and unpredictable output for each instance of the PUF. A Physical Unclonable Function (PUF) is a hardware component or device that generates a unique and unpredictable digital fingerprint that cannot be reproduced or cloned. PUFs are used for hardware security applications such as authentication, encryption, and secure key generation. The unique fingerprint generated by a PUF is derived from the inherent physical variations in the hardware component, such as the minute variations in the manufacturing process, temperature, or power supply conditions. These physical variations make it virtually impossible to reproduce or clone the PUF.

1.4.2 Types of PUFs

The types of Physical Unclonable Functions that are mostly used are,

I. SRAM PUF

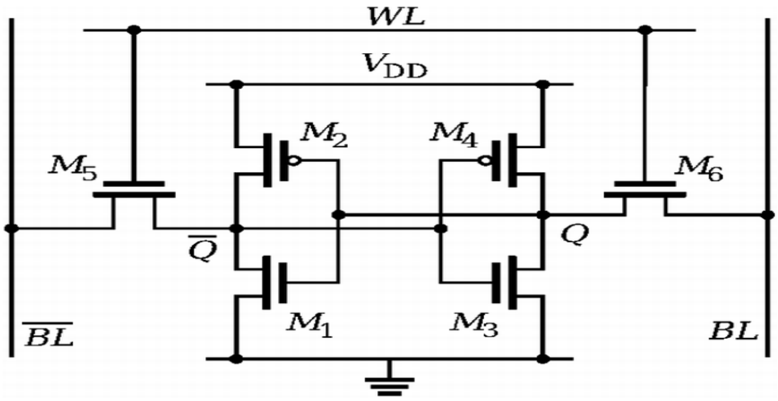
II. Arbiter PUF

III. Ring oscillator PUF

IV. Butterfly PUF

I.SRAM PUFs:

SRAM cells produce SRAM PUFs. Data bits are kept in SRAM cells. A SRAM cell is made up of two inverters that are cross-coupled. To implement read/write operations, two transistors are linked to these inverters. An SRAM cell's cross-coupled inverters should ideally be made to match. Therefore, the threshold voltages of the inverters should be equal in the ideal scenario. The threshold voltages of inverters vary in real life due to process changes like differences in doping concentration and ambient fluctuations like voltage and temperature. If environmental variation is ignored, the process variations are the remaining factors that are influencing threshold voltages. At the beginning of production, process variances are random, but once an SRAM cell is manufactured, it is static and will not change.

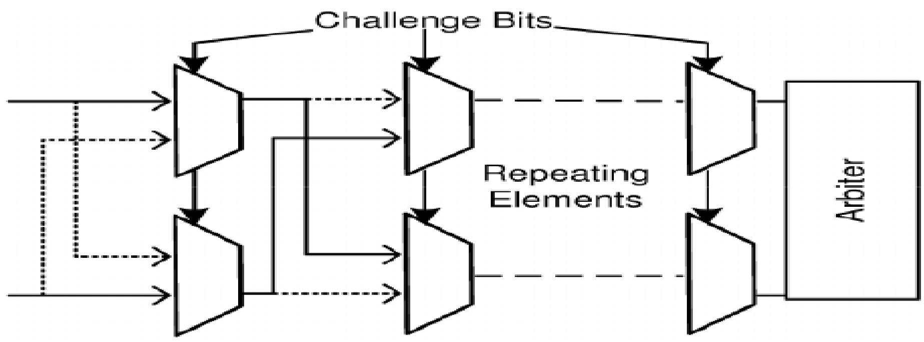


Courtesy: https://media.springernature.com/lw685/springer-static/image/chp%3A10.1007%2F978-3-030335823_42/MediaObjects/482498_1_En_42_Fig1_HTML.png

Fig-3: SRAM PUF

II. Arbiter PUFs:

Comparing two identical delay pathways is the notion. Both paths get an excitation at the same moment, and an arbitrator measures the paths' varying delays. This exposes details on the variances in the processes of two perfectly matched delay circuits. The arbiter-based PUF circuit is constructed to allow for the assertion of a challenge. Controlling switch components to create various delay routes presents difficulties. Two inputs, two outputs, and a challenge input are all included on a switch component.

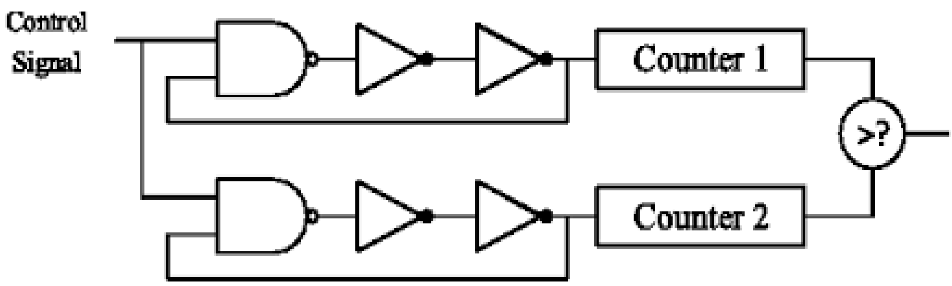


Courtesy: https://www.researchgate.net/publication/228851018/figure/fig1/AS:300916466044935@1448755484766/The-Arbiter-PUF-structure.png

Fig-4: Arbiter PUF

III. Ring Oscillator PUFs:

An oscillating signal is produced using ring oscillators. The total wire delay as well as the delays of each component employed in the ring oscillator loop affect the frequency of the oscillating signal. If identically created ring oscillators are taken into account, the delay model predicts that they should output a signal at a certain frequency. However, the oscillation frequency varies somewhat from ring oscillator to ring oscillator as a result of uncontrollable process variables. This shift can be magnified and used as a data to create device-dependent, physically unclonable identifying secret bits. Multiple ring oscillators, a frequency and comparison circuitry, and a PUF are the components of a ring oscillator-based PUF.

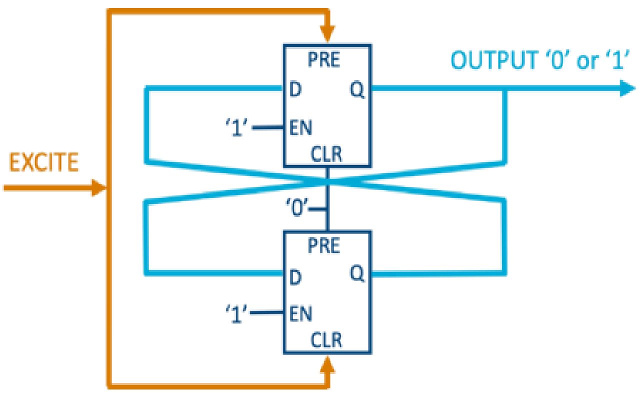


Courtesy: https://d3i71xaburhd42.cloudfront.net/ab6200a1076a8288d4b53fac7e2f4019de762157/3-Figure1-1.png

Fig-5: Ring Oscillator PUF

IV. Butterfly PUFs:

A pair of cross-coupled logic elements make up a PUF cell. By comparing the speeds of the logic components, a bit is produced at the cell's output. With an excitation, static process changes between logic components are monitored. The obtained bit is then utilised to produce secret bits. PUF is produced in a butterfly-based flip-flop algorithm. Using combinatorial loops, it is also possible to generate four butterfly PUF cells, but doing it in an FPGA is rather challenging. PUF cells are seen in latch-based butterfly structures.



Courtesy: https://www.intrinsic-id.com/wp-content/uploads/2020/03/Butterfly-PUF-based-on-latches-1024x540.png

Fig-6: Butterfly PUF

1.4.3 Advantages

- ***Uniqueness:*** PUFs generate responses that are unique to each individual device, making them highly suitable for authentication and identification purposes.
- ***Unpredictability:*** PUF responses are highly unpredictable, even for two devices that are manufactured under identical conditions. This property makes PUFs resistant to many types of attacks, including brute force attacks and side-channel attacks.
- ***Non-Volatility:*** PUF responses are non-volatile, meaning that they are retained even when power is removed from the device. This property makes PUFs suitable for use in low-power and battery-operated devices.

- ***Tamper Resistance:*** PUFs are highly resistant to tampering, as any attempt to modify the physical characteristics of the device will result in a change in the PUF response.
- ***Robustness:*** PUFs are robust against a wide range of environmental factors, including temperature, voltage, and aging effects. This property makes PUFs suitable for use in harsh operating conditions.
- ***Scalability:*** PUFs can be easily integrated into electronic devices of different sizes and form factors, making them suitable for a wide range of applications.

1.4.4 Applications

- ***Authentication:*** PUFs can be used to authenticate electronic devices and prevent counterfeiting. A PUF can be integrated into a device during the manufacturing process, and the unique response generated by the PUF can be used to verify the authenticity of the device.
- ***Secret Key Generation:*** PUFs can be used to generate unique cryptographic keys that are derived from the physical properties of the device. The keys generated by PUFs are resistant to many types of attacks, including brute force attacks and side-channel attacks.
- ***Secure Storage:*** PUFs can be used to securely store sensitive information, such as cryptographic keys and user credentials. The unique response generated by the PUF can be used as a key to encrypt the sensitive information, making it resistant to attacks.
- ***Anti-Counterfeiting:*** PUFs can be used to prevent counterfeiting of high-value products, such as pharmaceuticals and luxury goods. A PUF can be integrated into the product during the manufacturing process, and the unique response generated by the PUF can be used to verify the authenticity of the product.
- ***Tamper Detection:*** PUFs can be used to detect tampering with electronic devices. The response generated by the PUF can be used as a signature that is unique to the device. If the response changes, it indicates that the device has been tampered with.

1.4.5 Limitations

While PUFs (Physical Unclonable Functions) offer many advantages as a security primitive, they also have several disadvantages, including:

- ***Variability:*** The responses generated by PUFs can vary over time and with changes in operating conditions, such as temperature and voltage. This variability can lead to errors in authentication and identification.
- ***Reliability:*** The reliability of PUFs can be affected by manufacturing variations, aging effects, and environmental factors. These factors can lead to a reduction in the uniqueness and unpredictability of PUF responses over time.

- ***Sensitivity to Noise:*** PUFs are sensitive to noise and other forms of interference, which can affect the reliability and security of the device.
- ***Calibration:*** PUFs often require calibration to ensure reliable and accurate operation. This calibration process can be time-consuming and can add complexity to the implementation of PUFs in electronic devices.
- ***Limited Capacity:*** PUFs have a limited capacity for generating responses, which can limit their use in certain applications that require many responses.
- ***Security Risks:*** PUFs can be vulnerable to certain types of attacks, such as modeling and machine learning attacks, which can compromise the security of the device.

Overall, PUFs are a promising approach to hardware security, but they are not without their limitations and potential security risks. Careful consideration must be given to the implementation and use of PUFs to ensure their reliability and security in different applications.

CHAPTER 2

RELATED WORK

2.1 VLSI TECHNOLOGY

- The industry has fragmented from integrated device manufacturers to foundries and fabless, allowing each to concentrate on their area of expertise. John Y. Chen [1] outlined the fundamental changes to the VLSI business over the past few decades, including the growth of foundries and their ecosystem.
- It is challenging to achieve the speed and quality requirements of IC design given the rapid increase in size and complexity of VLSI. Wei Yan has therefore suggested an effective methodology for rapid floor planning in VLSI top-down hierarchical physical design flow utilising Active-Logic Reduction Technology. The suggested streamlined architecture significantly reduces the internal logical units by replacing the original modules in the netlist file with filler units that have no logical links [2].
- V. Kamakoti introduced a revolutionary method for creating transistor netlists from truth table descriptions of arbitrary digital circuits that makes good use of genetic algorithms (GA) [3]. It is known as the "genetic approach to gateless custom VLSI design flow."
- With the relentless advancement of device scaling continuing to excel in 10nm and beyond, Moore's Law has reached a new frontier. The design principles and the design processes, for both ASIC and custom designs, face unprecedented complexity as the physical dimensions of devices and interconnect are being reduced. Hsien-Hsin S. Lee thus outlined typical IC design issues and potential for improved technology [4].
- The traditional rule-based technology has reached its limits in tackling Electronic Design Automation (EDA) problems that have high dimensionality, discontinuous, and non-linearities as a result of the semiconductor industry's growing complexity. Machine learning (ML) is now gradually being used in EDA applications due to its quickly increasing demand. Therefore, Laura Wang [5] presented the numerous machine learning prospects and uses in IC design flow.
- In the simulation and synthesis of VLSI communication systems, CAD tools are crucial. Using a functional model library and scripting processes that automate iterative optimisation of algorithm parameters, the tools enable quick algorithm construction. To generate hardware designs quickly from algorithm descriptions, implementation tools are integrated into the algorithm design environment. R. Jain [6] discussed the significance of CAD tools for communication system design.
- J. Y. Lee provided an effective method for the design of signal processing systems implemented on VLSI that is based on signal flow graph analysis. A universal set of VLSI structures have been generated for homogeneous and shuffle-exchange SFGs in the proposed method, and it has been demonstrated that these structures are suitable to a broad range of signal processing methods [7].

- Self-timed digital design often employs a request-acknowledge protocol. Such a protocol has two main philosophies: two cycles and four cycles. There are seven primitives in data flow graphs. L. Merani developed the self-timed approach to VLSI digital filter design, in which all seven primitives were designed for both two-cycle and four-cycle protocols [8].
- Finite Impulse Response (FIR) filters can be created by formulating requirements that are appropriate to the needs of a given application. In order to upgrade filter components of the post-retimed circuit, such as adders and multipliers, based on VLSI design metrics such as area, speed, or power as according to designer's need, Deepa Yagain proposed an efficient FIR filter design using register minimization retiming technique and an optimisation environment design [9]. This code is synthesizable in Hardware Description Language (HDL).

2.2 FIR FILTERS

- In various applications of digital signal processing, a higher order of FIR filter is necessary to achieve precise frequency specifications. However, the complexity of the computation increases linearly with the number of additions and multiplications. Manish B.Trimale [10] suggested a variety of design strategies for the implementation of FIR filters in order to satisfy the aforementioned parameters.
- Both Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) filters can be created in digital signal processing from a given specification. Ranjushree Pal [11] provided a method to compare the effectiveness of planned FIR and IIR filters with regard to circuit complexity, group latency, phase distortion, and the output response.
- FIR filters offer a wide range of uses, including de-noising, biomedical applications, and signal processing. The FIR filter must be built with high speed and low power consumption in mind. S. Akash, M. Ajeeth, and N. Radha suggested the design of a parallel prefix adder FIR filter [12].
- The adder blocks, flip flops, and multiplier blocks are the three fundamental modules that make up the Finite Impulse Response (FIR) Filter. The multiplier, the slowest of all the blocks, has a significant impact on the FIR Filter's performance. Therefore, Shuchi Nagaria and Anushka Singh presented the design of a FIR filter utilising two different multipliers, namely the Array multiplier and the Booth Multiplier [13].
- Deepshikha Bharti and Kumari Nidhi Gupta proposed an efficient parallel adder that implements the two forms of FIR filter with a very small amount of delay while taking into account the cost of each operation in order to efficiently design various forms of FIR filters in terms of delay and area [14].
- Hui Zhao and Juebang Yu proposed a straightforward and effective method for designing one-dimensional variable fractional delay FIR digital filters [15] by combining two matrix equations, each

based on the weighted least-squares function of the optimal fixed fractional delay filter and the filter coefficient polynomial fitting.

- Adders, multipliers, and delay elements are the core components of FIR filters. Dr. H. V. Ravish Aradhya offered two innovative techniques for designing a high speed, low power 16-tap 32-bit digital FIR filter for DSP applications. Carry increment adders and 32-bit Vedic multipliers are used in the design of the proposed Filter-1 and proposed Filter-2, respectively [16]. The latter uses an updated carry skip adder and 32-bit Vedic multiplier.
- One of the crucial design criteria for the creation of digital filters is low complexity. For the effective design of FIR filters, Jiajia Chen developed a new approach to synthesise finite-precision coefficients with low implementation cost and low complexity from the frequency response requirements of linear phase FIR filters [17].
- Sung Hyun Yoon suggested a unique VLSI architecture for a multiplier-less FIR filter device offering variable-length taps. Two unique features—a data-reuse structure and a recurrent-coefficient scheme—were proposed for variable length taps. Due to the fact that it is a multiplier-less filter, these features lower the number of gates and take up less space than typical [18].
- By generating FIR structures without multipliers, Ken Sienski and Mike Legako have suggested a revolutionary FIR filter design method. Since there are no multipliers in this design, the occupied area is reduced and the filter's performance is enhanced by using linear programme techniques to limit the filter coefficients during the optimisation process [19].
- Hon Keung Kwan has described a technique for linear phase finite impulse response (FIR) digital filters utilising the Cuckoo search algorithm (CSA). The peak errors in passband and stopband obtained using the cuckoo search algorithm are similar to those obtained by the Parks-McClellan optimum approach, according to results of the Equiripple Linear Phase FIR Digital Filter Design [20].
- Due to its stability and simplicity in implementation for linear phase response, the finite impulse response (FIR) digital filter is a frequently used signal processing component in digital signal processing. Juan Zhao has presented a new optimisation method based on a sophisticated dynamic programming approach to reduce the amount of non-zero digits in the coefficient set, which results in reduced logic operator costs in the filter circuit implementation for an efficient FIR filter. The unnecessary nonzero digits in the coefficients were successfully eliminated by the suggested approach by utilising the knapsack method from dynamic programming [21].

2.3 HARDWARE SECURITY

- Security assurance has recently undergone a rapid evolution. Based on software security principles, many businesses have embraced the Security Development Lifecycle as a process to find and address vulnerabilities in their products. Therefore, Hareesh Khattri suggested a Hardware Security Development Lifecycle at the hardware technology level that has been applied to commercial CPUs, chipsets, and SoCs, which can quicken the identification of security flaws in computer hardware products [22].
- Hardware security and trust have grown in importance over the past two decades as a result of the globalisation of the semiconductor supply chain and the pervasive network connectivity of computer devices. Wei Hu [23] provided a summary of hardware security and trust from the perspectives of risks, countermeasures, and design tools.
- As network technology has advanced, more and more people and businesses are becoming concerned about network security. The world is aware of how software viruses affect information security, but research on hardware risks is significantly less extensive than that on software attacks. Zhiyong Jiao [24] provided an overview of hardware threats from the perspective of information security.
- The term "hardware trojan" refers to the malicious modification of hardware that occurs during fabrication or design. Rajat Subhra Chakraborty [25] has suggested a Trojan taxonomy, models of Trojan activities, and a study of the most recent Trojan prevention and detection approaches after examining the threat that hardware Trojans pose and the strategies for thwarting them.
- The computing systems that are utilised in practise are intricate and depend on the interaction of various hardware parts to function properly. Therefore, it is crucial to keep these computer systems secure. Nachiketh Potlapally [26] provided an overview of approaches to handle the complexity of certifying hardware security and listed the opportunities available for hardware security validation.
- Jiliang Zhang proposed a practical logic obfuscation method with low overheads to prevent an adversary from RE both the gate-level netlist and the layout-level geometry of IP/IC and protect IP/IC from piracy and overbuilding because the existing hardware security techniques require high overheads and IC camouflaging cannot provide any protection for the gate-level netlist of the third-party intellectual property (IP) core [27].
- Hardware security is a subtly placed extra circuitry in a circuit layout that, in very rare instances, can interfere with the functionality of the circuit. Hassan Salmani therefore suggested a fresh layout-level vulnerability analysis to hardware Trojan insertion. The vulnerability analysis flow enables the investigation of each circuit layout corner's susceptibility to various kinds of functional hardware Trojans [28].

- From the perspective of authentication, time to activate a hardware trojan circuit is a big challenge. Analyzed is the functional Trojan's time to generate a transition. Transitions are simulated using geometric distribution, and the expected number of clock cycles needed to make a transition is then calculated. Mohammad Tehranipoor then suggested a dummy scan flip-flop insertion process in an effort to shorten the time needed to generate transitions [29].
- The majority of third-party intellectual property (IP) cores, which may be infected with malicious hardware trojans that could reveal sensitive information, are used in the design of contemporary computer hardware. In order to prove the security features of hardware designs, Maoyuan Qin [30] introduced a novel formal verification method that employs the fine-grained gate level information flow model.
- Hardware Trojan is a term used to describe malicious IC design modifications. The involvement of numerous parties in the VLSI design cycle has made it extremely difficult to detect hardware trojans. Therefore, Sreeja Rajendran [31] proposes a path retrace approach based on reverse engineering method to detect the inserted or deleted nets, i.e., hardware trojan detection, by an adversary together with its location.
- The security against hardware trojan attacks becomes crucial since integrated circuits are used in the majority of the products and systems on which society depends. In order to detect any existing trojan circuitry, Chris Nigh suggested a combinational hardware trojan detection technique through adaptive test pattern generation [32]. This technique uses superposition to do a fine-grained circuit analysis.
- National security is seriously endangered by malicious hardware trojan circuitry implanted in safety-critical applications. Rajat Subhra Chakraborty thus suggested a novel application of a key-based obfuscation technique to achieve security against hardware trojans. In order to implement the Obfuscation method, a given circuit's state transition function must have its reachable state space expanded and be given the ability to operate in two separate modes [33].

In IOT applications, hardware-based security primitives are crucial for system protection. Physical Unclonable Function (PUF) and True Random Number Generator are the primary primitives. (TRNG). N. Nalla Anand Kumar [34] discussed the effective design, implementation, and analysis of hardware-based security primitives.

2.4 PHYSICAL UNCLONABLE FUNCTION (PUF)

- Electronic devices have become a regular aspect of our life in the current day. Therefore, it is becoming essential to protect sensitive data as well as the underlying technology. Therefore, Koustav Dev [35] offered a solution to the hardware security issue utilising Physically Unclonable Function (PUF), which are electronic circuits used to produce distinctive and trustworthy signatures of a particular electronic circuit.

- Physical Unclonable Function (PUF) is a physical thing that responds physically to a challenge yet cannot be duplicated. Thus, Srinivas Devadas offered an overview of PUFs, its applications, and the use of PUFs in low-cost authentication and key generation applications [36].
- Physical Unclonable Function is a cutting-edge hardware security primitive that creates a distinctive identifier that serves as a fingerprint for an object by using the physical properties of that object. Numerous indicators, including uniqueness, dependability, and uniformity, are used to evaluate PUF performance. So, Fahem Zerrouki [37] suggested a technique that evaluates the potency and measures the performance of a certain PUF.
- FPGAs are facing numerous security concerns in the present day as enemies attempt to benefit by copying the original design without making any expenditure. The main security concern in those is the introduction of harmful components known as hardware trojans. In order to provide identity and authentication for an IC, PUFs can be used to generate a secret key that must be implanted in the IC. Therefore, Mary Latha Ch [38] provided an effective design and implementation of a safe Physical Unclonable Function (PUF) in FPGA.
- Physical Unclonable Functions (PUFs), which create distinctive fingerprints based on physical device attributes, have grown significantly in popularity in recent years in security applications including entity authentication and cryptographic key creation. So, Achim Bittner suggested the creation of a cantilever PUF for security purposes that is based on a particular MEMS device [39].
- IOT devices are susceptible to attacks, both physical and virtual. The most crucial component in non-physical attacks is to protect the data on memory devices. Physical Unclonable Functions (PUFs) are the simplest and most effective way to protect the data on memory devices, so Parman Sukarno proposed a way to assess a PUF's randomness, uniqueness, and stability [40].
- Physical Unclonable Functions (PUFs) are being used more frequently as a result of growing hardware security and trust challenges. Therefore, Durba Chatterjee [41] presented a formal PUF representation language capable of representing any PUF construction or combination upfront for the formal analysis of Physical Unclonable Function (PUF).
- In recent years, there has been an increase in the usage of Physical Unclonable Functions (PUFs) to produce fingerprints for secure authentication and protection of electronic circuits and chips. This requires the effective design of a secure Physical Unclonable Function as well as the analysis of that design. Walter Hansch [42] has therefore suggested a measurement setup that is effective, simple, and extremely accurate.
- A Physical Unclonable Function's noisy measurements are used to store keys with reliability, security, privacy, and complexity restrictions. In order to produce bit-error probability outcomes that were much

better than all the current approaches for key binding with PUFs, Onur Gunlu [43] suggested a novel set of low-complexity and orthogonal transforms without multiplication.

- In order to validate the strength and security of an electronic network that offers secured network access and control as well as enabling user identification and hardware device verification, it is crucial to authenticate the authenticity of electronic devices. Therefore, A. M. Shah [44] provided a brief summary of the different types of PUFs, their uses, problems, and how PUFs might be implemented in hardware and software.
- Hardware security of restricted devices can be effectively solved with Physical Unclonable Function (PUF). A PUF with reversible logic has the capacity to produce ultra-powerful PUF, which may be applied in any limited situation. As a result, Ashok Kumar [45] proposed a low-cost, reversible-based tunable Ring oscillator design with Physical Unclonable Function.
- For many FPGA IP manufacturers, hardware design IP protection is the most crucial requirement. Sandeep S. Kumar then suggested a method for IP protection on FPGA utilising Butterfly PUF. PUFs are a special type of physical devices that extract secrets from intricately designed integrated circuits. These secrets, together with the unclonability of PUFs, make them a very secure way to create volatile secret keys for cryptographic operations [46].

CHAPTER 3

METHODOLOGY

3.1 INTRODUCTION

In numerous real-time applications, including communications, signal processing, and consumer electronics, Finite Impulse Response (FIR) filters are employed. Because of its inherent stability and linear phase, FIR filters are widely sought-after in the construction of digital filters. They benefit from being time-invariant and are simple to build with little computational requirements. Therefore, hardware security of FIR Filter is very essential for good performance and to produce perfect results. But there is a possibility of hardware threats like tampering, reverse engineering, hardware trojan etc as the design of filter involves many stages. As a result of these hardware attacks on the FIR filter, there will be several problems like performance degradation, leakage of confidential information, lack of stability etc. So, the main aim of this project is to design and implement a secured Finite Impulse Response (FIR) Filter using Physical Unclonable Functions (PUFs).

3.2 WHY PUFs?

A Physical Unclonable Function (PUF) is a hardware component or device that generates a unique and unpredictable digital fingerprint that cannot be reproduced or cloned. PUF responses are unique and highly unpredictable. Even for two devices that are manufactured under identical conditions PUF produces different results. The unique fingerprint generated by a PUF is derived from the inherent physical variations in the hardware component, such as the minute variations in the manufacturing process, temperature, or power supply conditions. These physical variations make it virtually impossible to reproduce or clone the PUF. This property makes PUFs resistant to many types of attacks, including brute force attacks and side-channel attacks.

There are different types of Physical Unclonable Functions (PUFs) available like SRAM PUF, Arbiter PUF, Ring Oscillator PUF, Butterfly PUF etc. From the above available Physical Unclonable Functions (PUFs), we chose Butterfly PUF.

3.3 WHY BUTTERFLY PUF?

Butterfly PUF is a type of PUF that has gained popularity due to its unique design and there are several advantages over other PUFs. Some of the advantages of Butterfly PUF over other PUFs are:

High Reliability: Butterfly PUF has high reliability, which means that it is less likely to generate the same response for different challenges. This feature makes it more difficult for attackers to clone the device.

Low cost: Butterfly PUF is less expensive to implement than other PUFs. It requires fewer components, which leads to lower fabrication costs.

Low Power Consumption: Butterfly PUF has low power consumption, which is beneficial for devices that have limited power resources, such as IoT devices.

Simple Design: Butterfly PUF has a simple design, making it easy to integrate into a system. Its simplicity also makes it easier to test and verify its functionality.

Scalability: Butterfly PUF can be easily scaled up or down depending on the application. This feature makes it suitable for a wide range of devices, from small IoT devices to larger systems.

Robustness: Butterfly PUF is highly robust, which means that it is resistant to various forms of attacks, such as side-channel attacks, invasive attacks, and machine learning attacks.

Overall, the advantages of Butterfly PUF make it an attractive option for device authentication and identification in various applications, such as IoT, embedded systems, and mobile devices.

3.4 IMPLEMENTATION

A PUF-based FIR filter design involves using a PUF as a secure key to generate the coefficients of an FIR filter. The PUF'S unique response to a challenge is used to generate a set of coefficients that are unique to the device, making it difficult to replicate or reverse engineer the filter. The PUF also provides tamper-resistance by making it difficult to modify the coefficients after manufacturing. This combination of security and uniqueness makes PUF-based FIR filters useful for applications such as secure signal processing, cryptographic key generation, and secure communication.

This project is mainly divided into 3 steps:

1. Design of FIR Filter
2. Generation of Coefficients from the Butterfly PUF
3. Mapping the generated coefficients to designed FIR filter

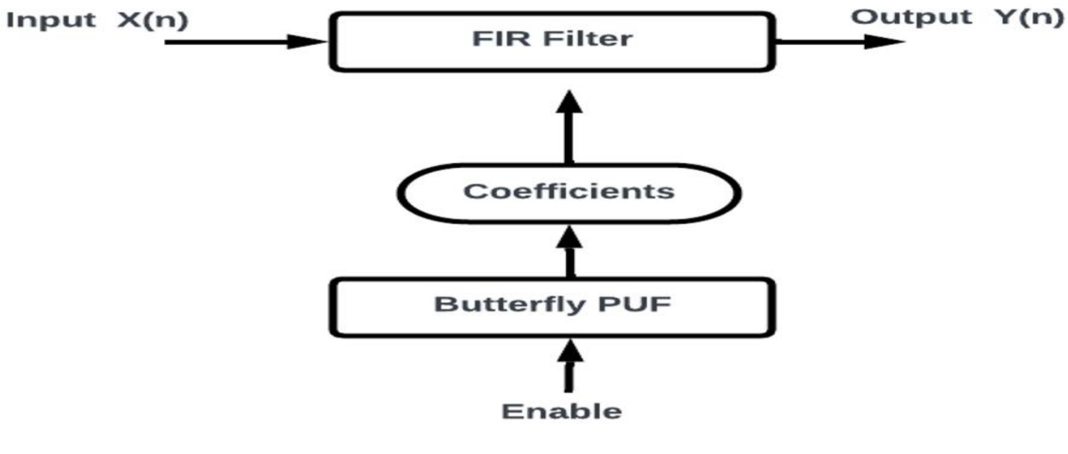


Fig-7: Flowchart of proposed methodology

3.4.1 Design of FIR Filter

As a first step, a basic 4-tap FIR Filter is designed using Verilog hardware description language (HDL). The output equation of an N-tap FIR Filter can be given as,

$$\mathbf{Y(n) = b_0 * x[n] + b_1 * x[n-1] + b_2 * x[n-2] + \ldots\ldots\ldots + b_N * x\,[n-N]} \;\ldots\ldots\ldots\ldots\ldots\ldots\mathbf{(2)}$$

Where, y(n) is the output response of the FIR Filter

N is the order of the FIR Filter

X(n) is the input signal

b_0, b_1 , ………. b_N are the coefficients of FIR Filter

The FIR Filter generates the filtered output by convolution of the input signal with a collection of filter coefficients or taps. A discrete-time signal is created by first sampling the input signal at regular intervals. Then, a set of filter coefficients that have been predetermined and stored in the filter are multiplied by the sampled input signal. The products of the input samples and the filter coefficients are added together, producing a sum. The sum obtained is output i.e., the filtered signal for that input sample. After the filtered output is obtained, the input samples are shifted by one position, and the process is repeated with the new input sample.

3.4.2 Generation of Coefficients from Butterfly PUF

A Butterfly PUF is designed using Verilog hardware description language (HDL) to generate the unique coefficients for the FIR Filter which acts as security keys for filter. Initially, the excitation signal is turned up in order to begin the operation of the Butterfly PUF. The butterfly PUF circuit reaches an unstable operating point (because the inputs and outputs of both latches are opposite signals). After a few clock pulses, the excite signal is lowered. This starts the PUF circuit's transition to one of the output signal's two stable states, '0' or '1'.

A Butterfly PUF can generate a single bit data i.e., 0 or 1 for a single clock pulse. The output of the PUF for 8 clock pulses is obtained since we want 8-bit data, and it is then placed in a register so that it can be utilized as a coefficient for the FIR Filter. Four 8-bit coefficients are produced by four 4-Butterfly PUFs for a 4-tap Finite Impulse Response (FIR) Filter, and N-Butterfly PUFs are needed for an N-tap FIR Filter.

3.4.3 Mapping of generated coefficients to designed FIR filter

As a final step, the generated coefficients from the Butterfly PUF will be given to the designed FIR Filter. After applying these coefficients to the filter, it performs convolution operation to produce the filtered output i.e., the coefficients will be multiplied with the input signal and then all the products will be added together to produce a sum, which is the output of the FIR Filter.

Let the generated coefficients from the Butterfly PUF for a 4-tap FIR Filter are, W_0, W_1, W_2 and W_3 and the input is x(n), then the output equation of the 4-tap FIR Filter can be written as,

$$Y = W_0\ X(n) + W_1\, X(n-1) + W_2\, X(n-2) + W_3\, X(n-3) \ldots\ldots..(3)$$

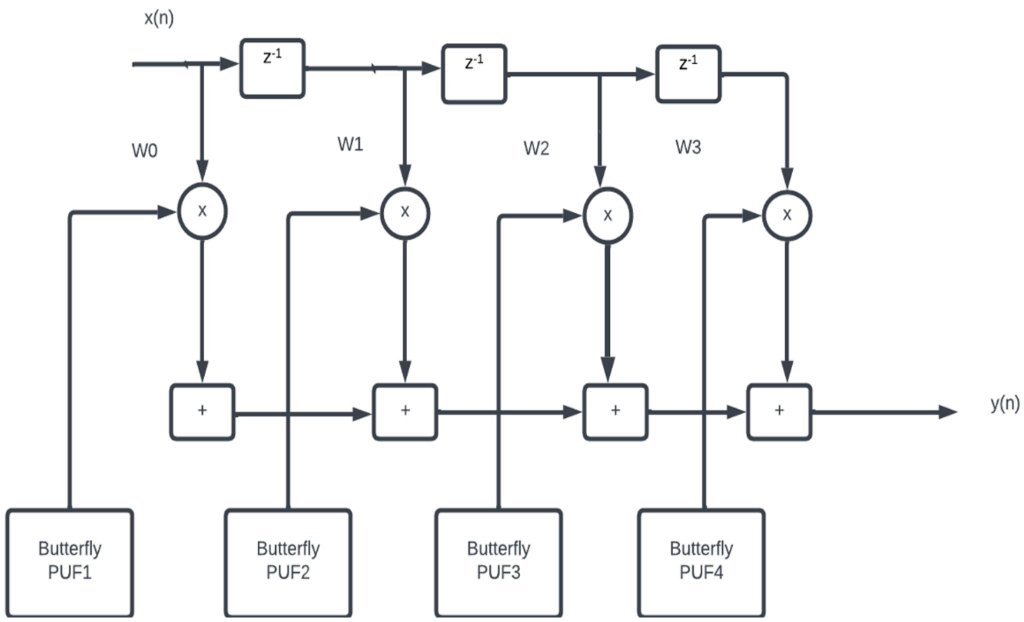


Fig-8: Block diagram of trojan aware FIR filter

As the coefficients generated from the Butterfly PUF are unique, it is difficult to replicate or reverse engineer the filter. This combination of security and uniqueness makes PUF-based FIR filters more secure and they can be used for applications such as secure signal processing, cryptographic key generation, and secure communication.

CHAPTER 4

RESULTS

4.1 Butterfly PUF results from Vivado

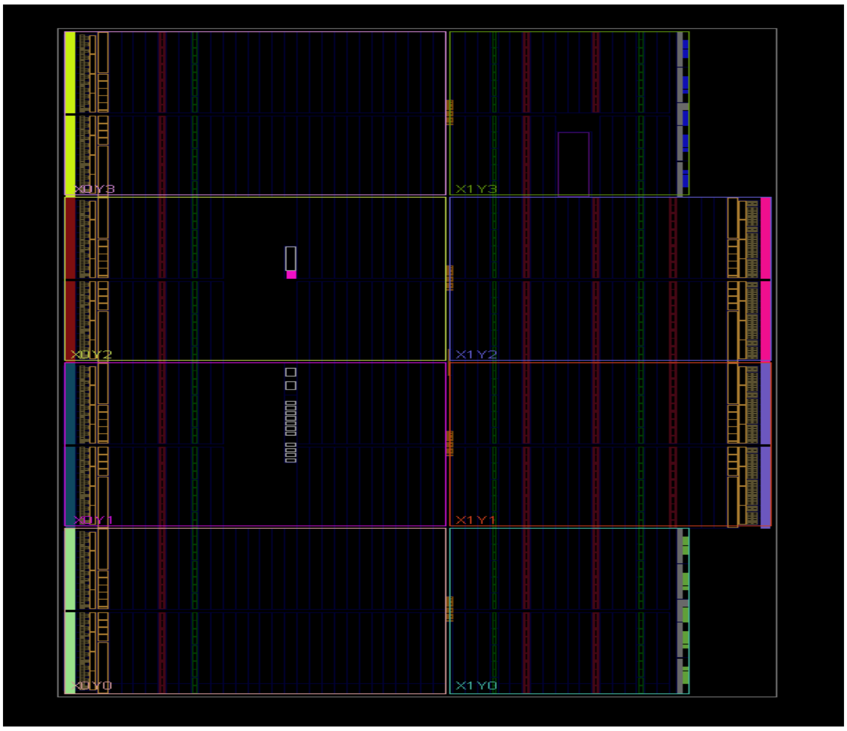


Fig-9: Implemented Design of Butterfly PUF

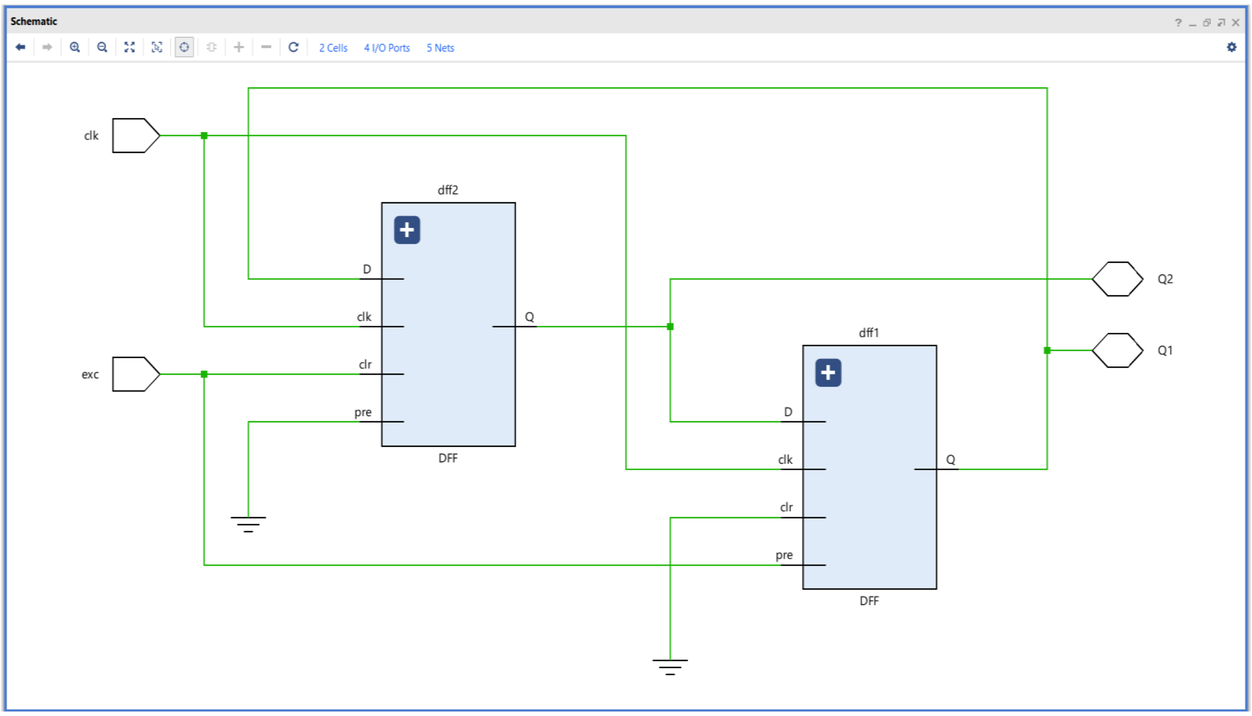


Fig-10: Schematic of Butterfly PUF

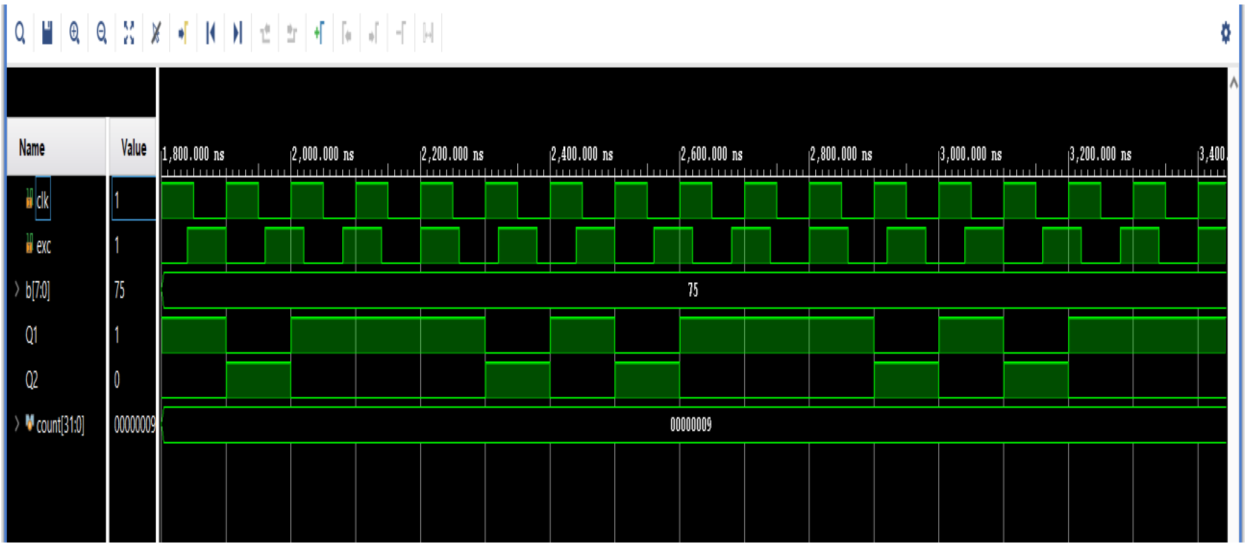


Fig-11: Output of Butterfly PUF

4.2 FIR Filter results from Vivado

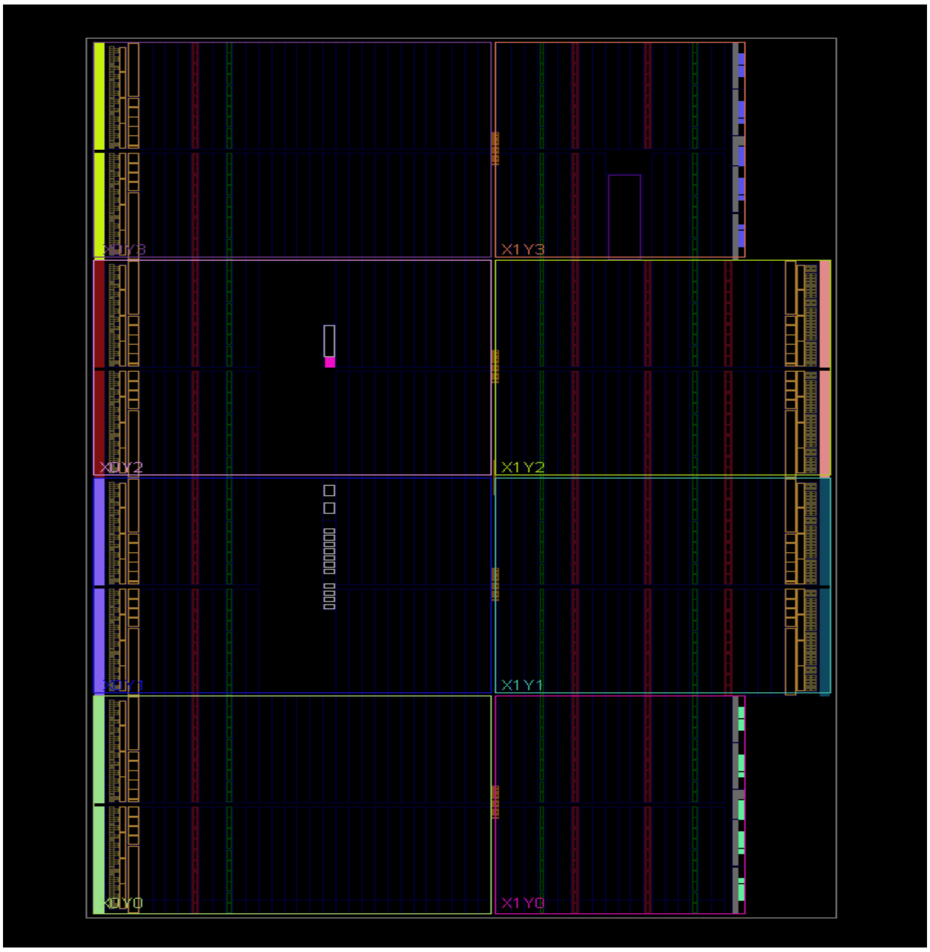


Fig-12: Implemented Design of Secured FIR filter

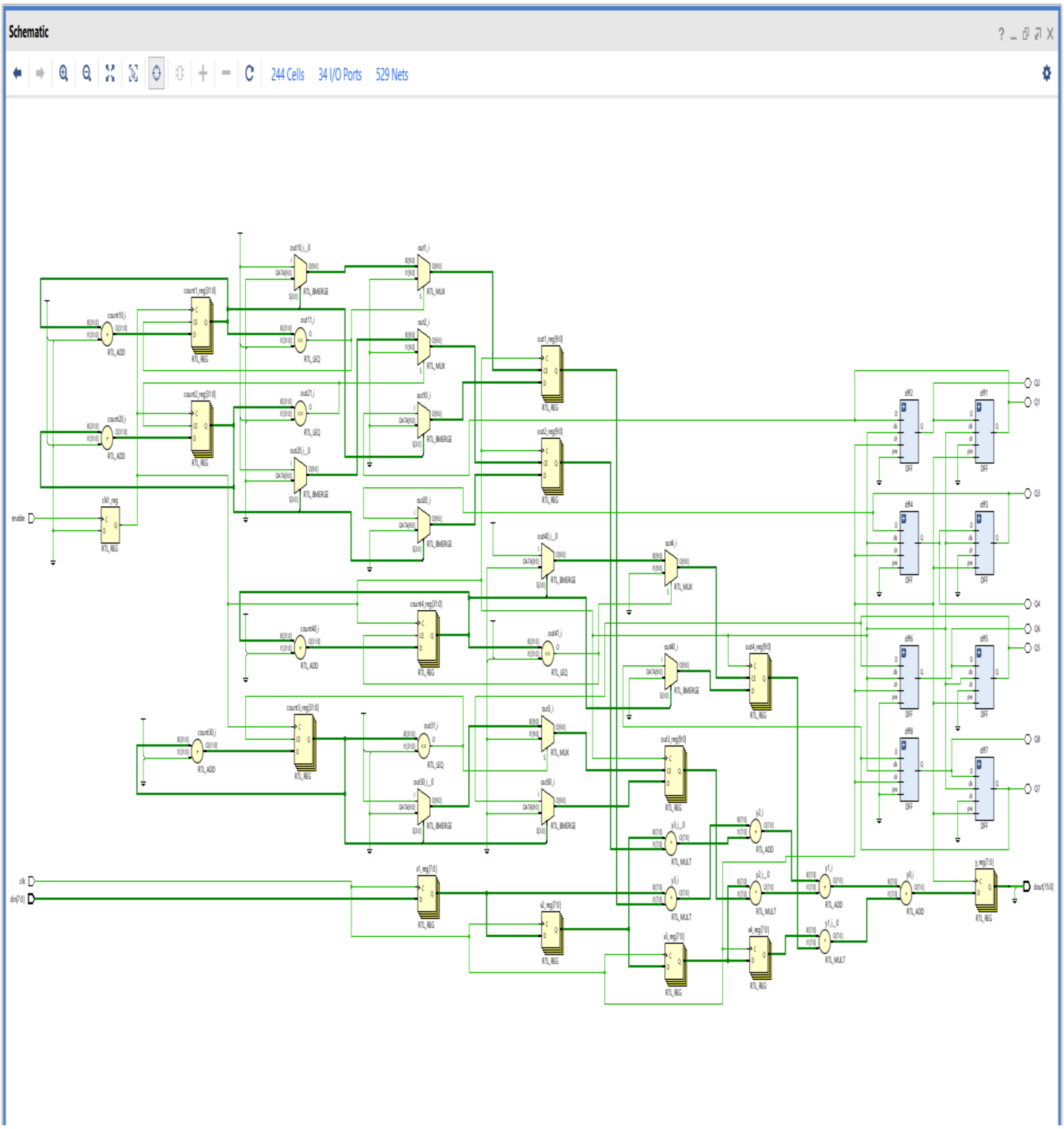


Fig-13: Schematic of Secured FIR Filter

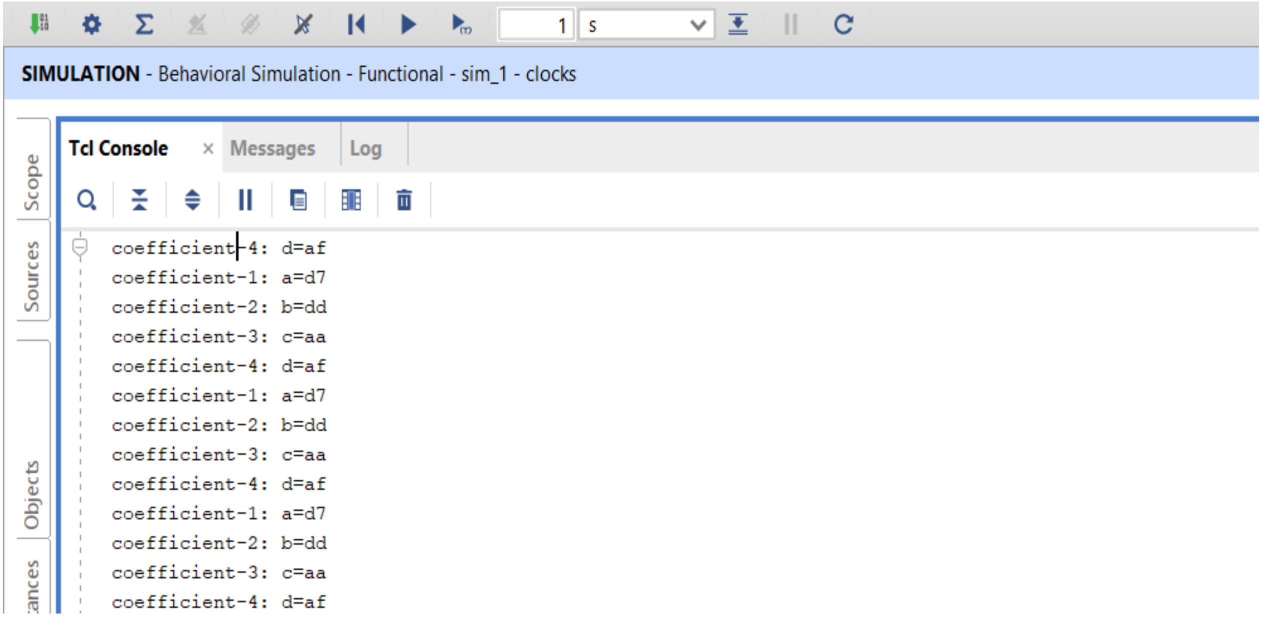


Fig-14: Coefficients generated from Butterfly PUF (1)

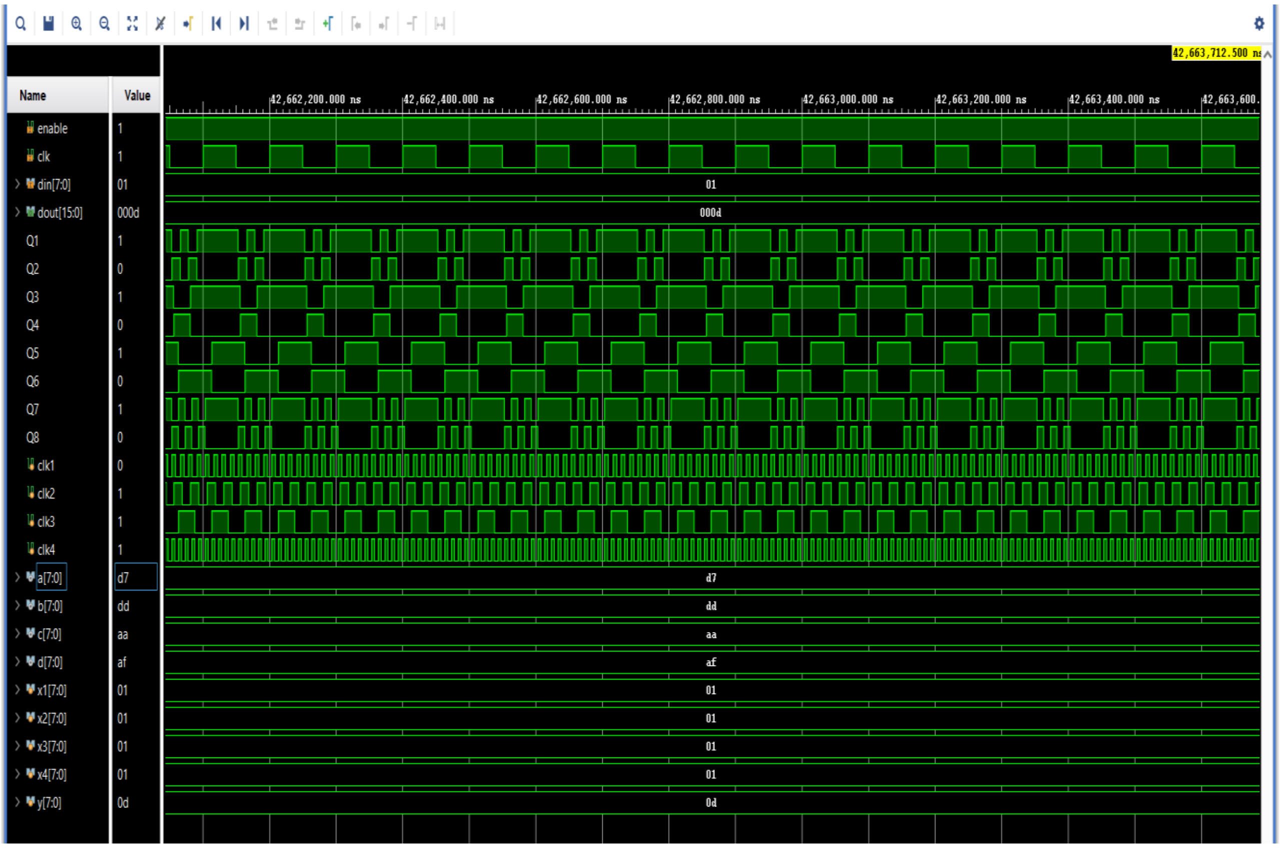


Fig-15: Output of FIR Filter (1)

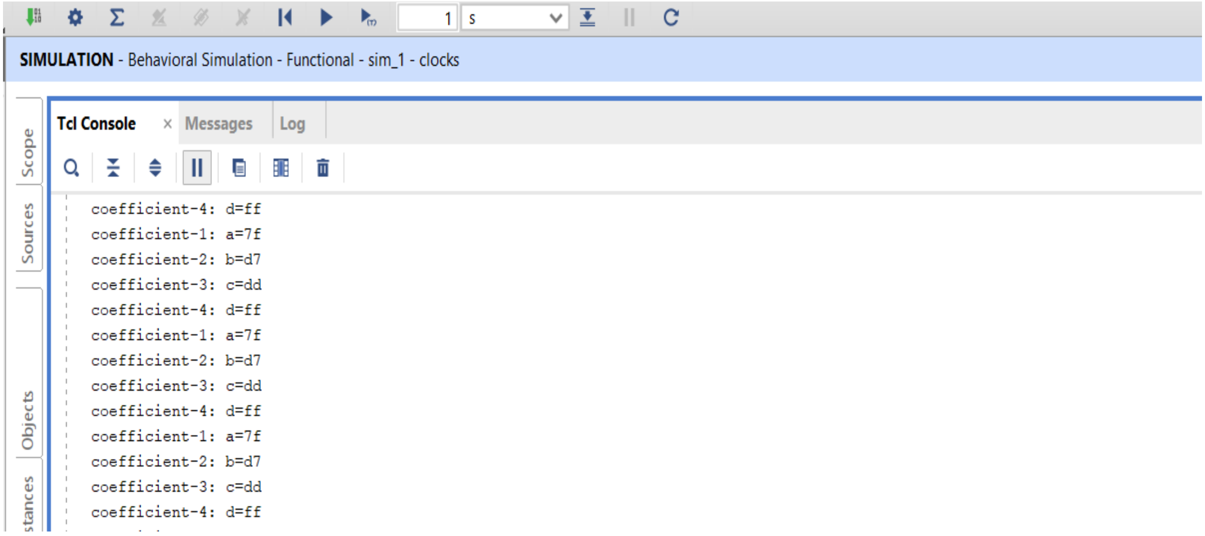


Fig-16: Coefficients generated from Butterfly PUF (2)

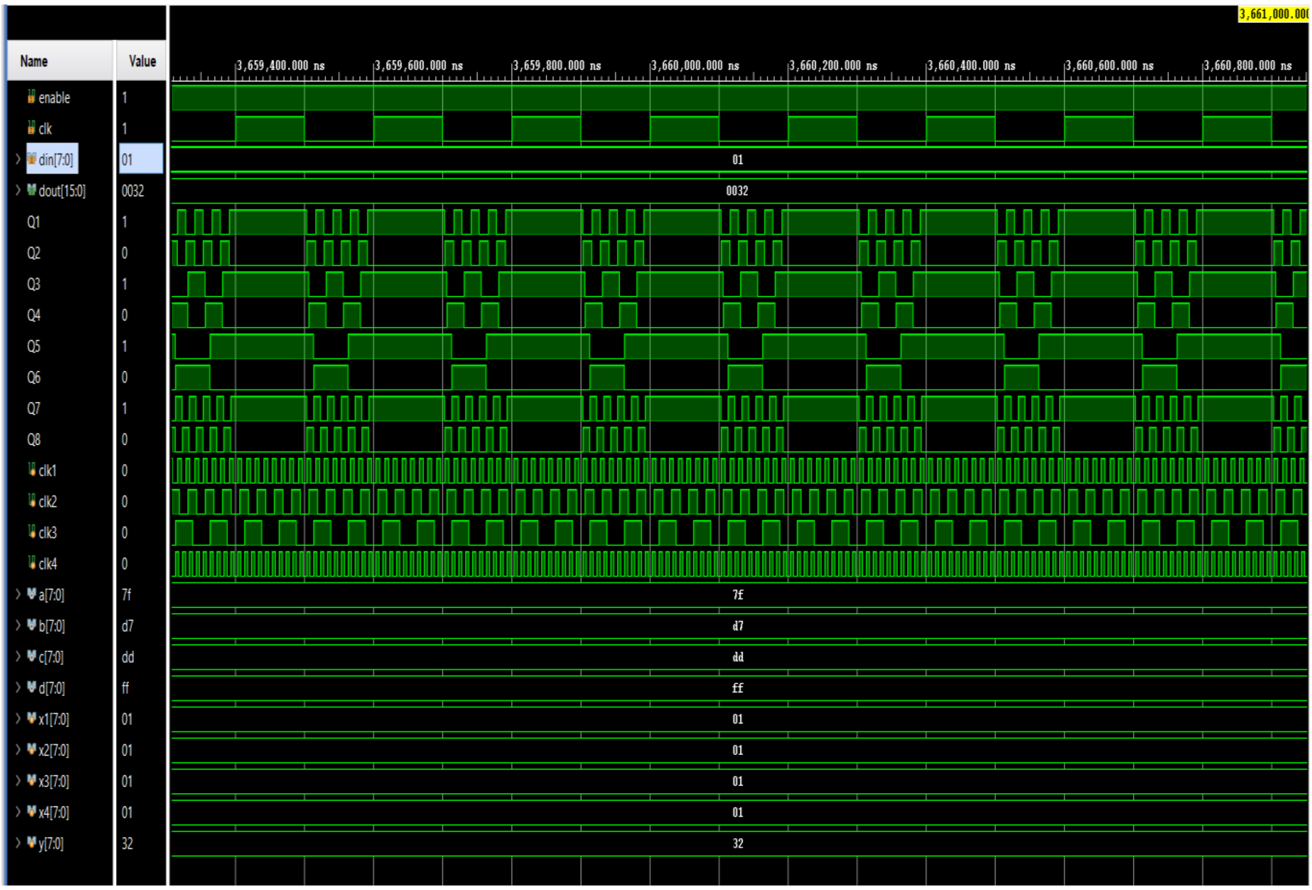


Fig-17: Output of FIR Filter (2)

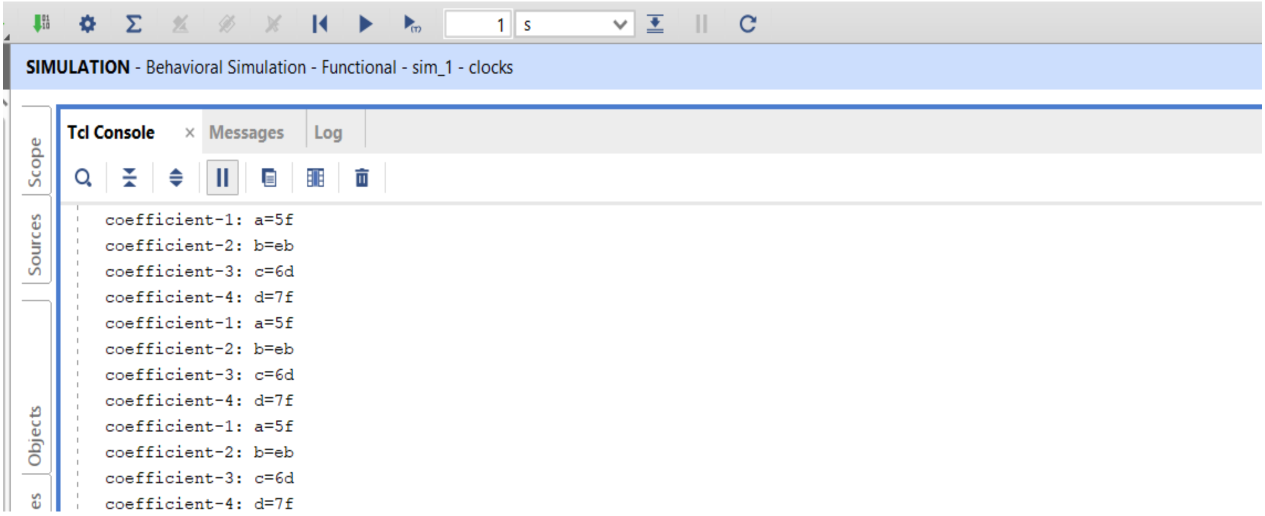


Fig-18: Coefficients generated from Butterfly PUF (3)

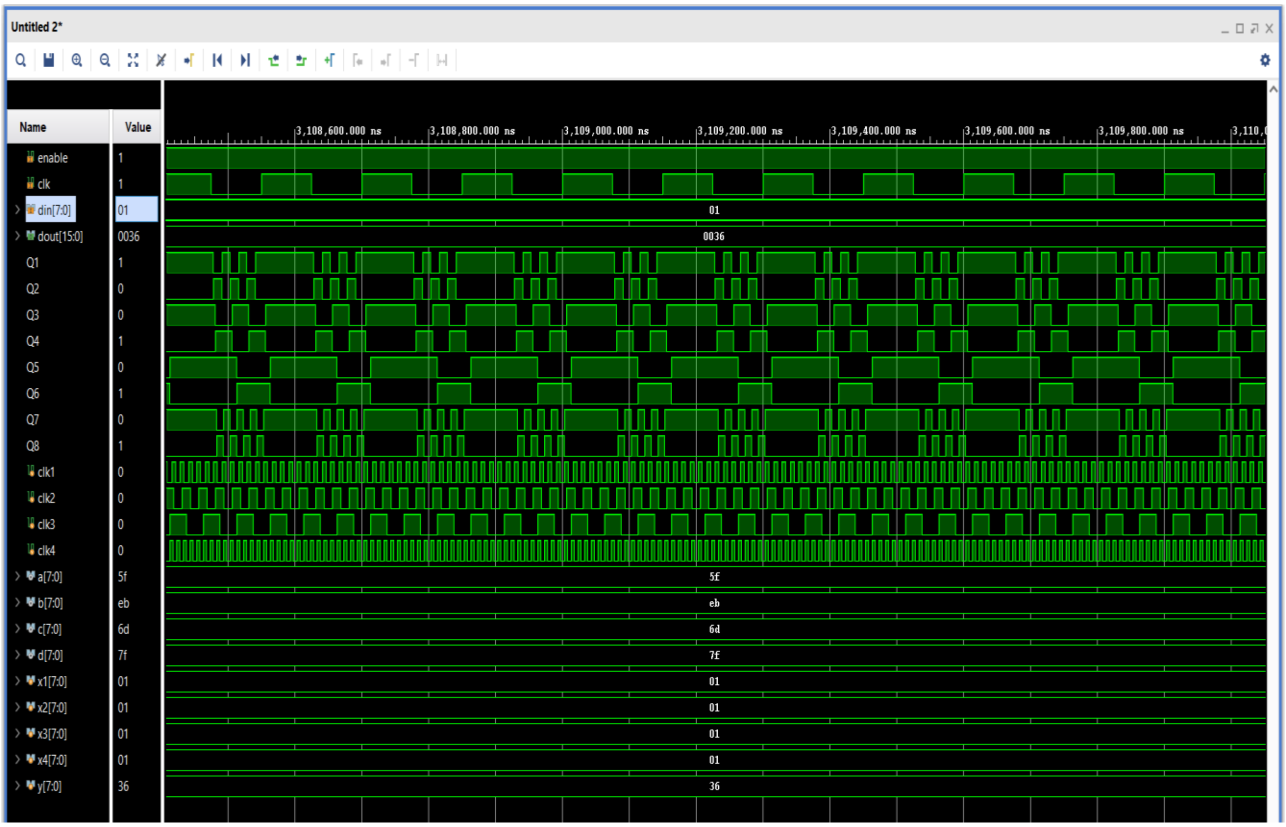


Fig-19: Output of FIR Filter (3)

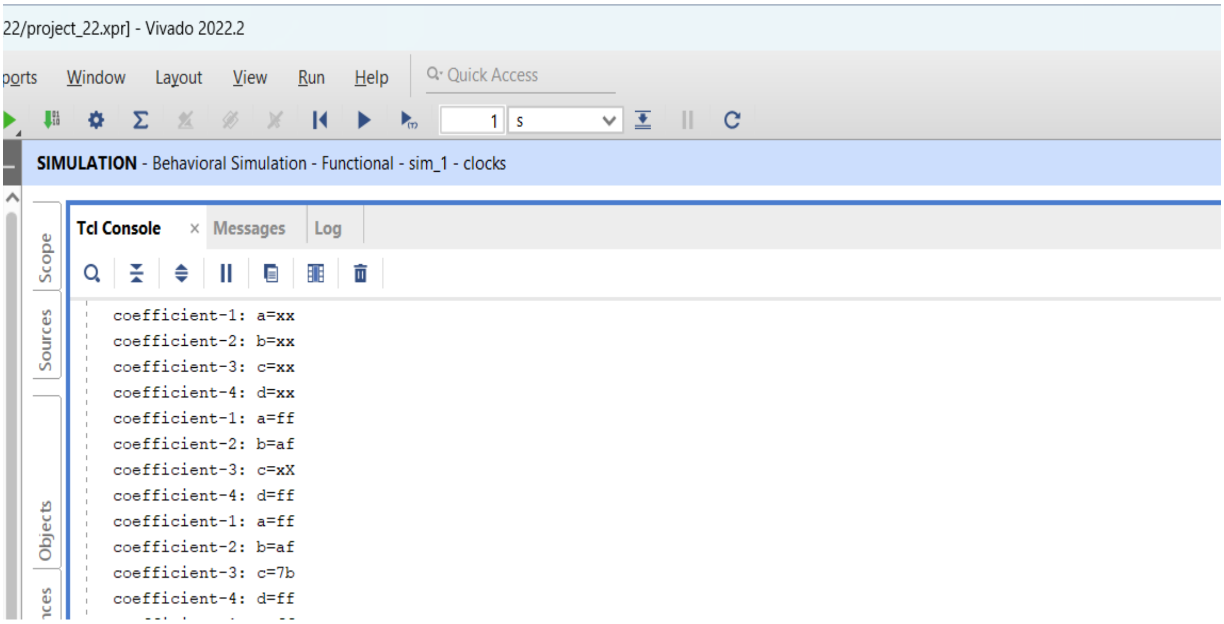


Fig-20: Coefficients generated from Butterfly PUF (4)

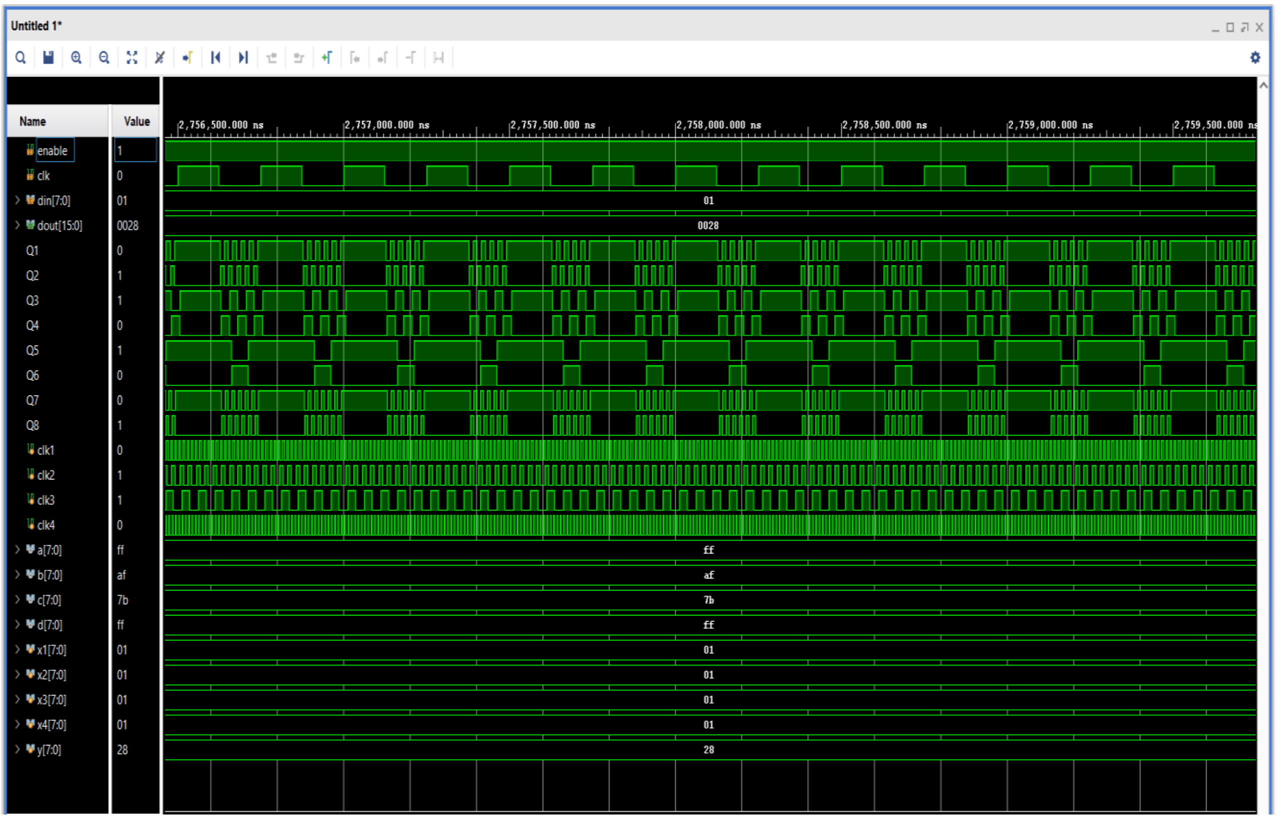


Fig-21: Output of FIR Filter (4)

CHAPTER 5

CONCLUSION AND FUTURE WORK

As the PUFs generate unique responses, it is difficult to replicate or clone or reverse engineer the filter. Here, the coefficients generated from the PUF acts as a secret key for the FIR Filter which are unpredictable. So, the security of the FIR Filter increases which makes it secure from hardware trojans or any other hardware attacks. Therefore, PUFs offer a promising approach to hardware security, as they provide a means of generating unique and unpredictable responses that are resistant to many types of attacks. Hence, PUF based FIR Filters acts as a trojan aware FIR filter. In future, the performance and robustness of the filter can be improved by increasing the efficiency of the PUF used.

APPENDIX

A. OVERVIEW OF VERILOG HARDWARE DESCRIPTION LANGUAGE (HDL)

Introduction

Verilog is a language used to create digital circuits and systems in the hardware description language (HDL). It was created by Gateway Design Automation in the middle of the 1980s, and Cadence Design Systems eventually purchased it. Digital signal processors (DSPs), microprocessors, and application-specific integrated circuits are just a few examples of the digital circuits and systems that are frequently designed and verified using Verilog. (ASICs).

Verilog is a language for describing the behaviour and structure of digital circuits. It allows designers to describe the functionality of their circuits at a high level of abstraction, and then simulate and verify their designs before they are implemented in hardware. Verilog includes a variety of constructs for modelling digital circuits, such as modules, wires, and registers, as well as a rich set of operators and control structures for specifying the behaviour of the circuit.

Verilog is often used in conjunction with hardware synthesis tools, which can automatically translate a Verilog design into a netlist of gates and flip-flops that can be implemented in hardware. Verilog is also commonly used in conjunction with simulation tools, which can be used to simulate the behaviour of a Verilog design before it is synthesized.

Verilog has evolved over the years and is now widely used in the design and verification of digital circuits and systems. It is supported by many EDA (electronic design automation) vendors and is used in a wide range of applications, from simple digital circuits to complex multi-million gate ASICs.

The fact that hardware description languages like Verilog have techniques to describe signal strengths and propagation times makes them similar to software programming languages. (sensitivity). Assignment operators come in two flavours: blocking assignments (=) and non-blocking assignments (=). Designers can express a state-machine update without having to declare and employ temporary storage variables thanks to the non-blocking assignment. Designers could readily create reasonably brief and succinct descriptions of massive circuits since these ideas are a part of the semantics of the Verilog language. When Verilog was first introduced (1984), circuit designers were already utilising graphical schematic capture software and specially created software programmes to describe and simulate electrical circuits. Verilog offered a significant productivity gain for these designers.

History of Verilog

Verilog was first introduced in 1984 by Gateway Design Automation, a company founded by Phil Moorby and Prabhu Goel. It was initially developed as a proprietary language for designing and verifying digital circuits and systems. The language was first called "Gateway Design Automation's Verilog-XL" and was later renamed to just "Verilog."

In 1985, Verilog was released as a public domain language, and it quickly gained popularity in the industry. Verilog's popularity was largely due to its ability to model complex digital circuits and systems with a high level of abstraction, allowing designers to quickly prototype and test their designs.

In 1990, Cadence Design Systems acquired Gateway Design Automation, and Verilog became one of Cadence's flagship products. Over the years, Verilog continued to evolve, and in 1995, the Verilog Hardware Description Language (HDL) standard was published by the Institute of Electrical and Electronics Engineers (IEEE). The standardization of Verilog helped to further its adoption and allowed for greater interoperability between different EDA (electronic design automation) tools.

In 2001, the IEEE published a new version of the Verilog standard, called Verilog-2001, which added several new features, including the ability to model complex testbenches and support for object-oriented programming.

In 2005, the IEEE published yet another update to the Verilog standard, called Verilog-2005, which added new features such as the ability to model analog circuits and support for mixed-signal simulation. Today, Verilog remains one of the most widely used hardware description languages in the industry, and it is supported by many EDA vendors and toolchains. Its popularity has been driven by its ability to model complex digital circuits and systems with a high level of abstraction, as well as its interoperability with other EDA tools and standards.

Features of Verilog HDL

Verilog HDL offers many useful features for hardware design

- Verilog (verify logic) HDL is a general-purpose, user-friendly hardware description language. The syntax is comparable to the C programming language. Verilog HDL will be simple for designers with knowledge in C programming to learn.
- Verilog HDL enables mixing of several abstraction layers within a single model. So a designer has the option of defining a hardware model in terms of switches, gates, RTL, or behaviour code. A designer simply needs to master one language to create hierarchical and stimulus designs.
- Most widely used logic synthesis software is Verilog HDL compatible. As a result, designers favour it as their language of choice.

- Verilog HDL libraries are offered by all fabrication vendors for post-logic synthesis simulation. Therefore, the most vendor options are available while developing a chip in Verilog HDL.

A strong component of Verilog is the Programming Language Interface (PLI), which enables users to interact with the internal data structures of Verilog using bespoke C code. A Verilog HDL simulator can be tailored by designers using the programming language interface. (PLI).

Module Declaration

In Verilog, a module is the main design entity. The name and port list are specified in the first line of a module declaration. (arguments). The width of each port and I/O type (input, output, or inout) are specified in the following few lines with a 1-bit port by default. Next, wire, reg must be declared for the port variables. Wire is the standard. Since their data is latched outside of the module, inputs are typically wired. When signals were saved inside the initial or always block, outputs are typed accordingly.

Syntax

```
module model_ name(port_list);
input [msb:lsb] input_port_list;
output [msb:lsb] output_port_list;
inout [msb:lsb] inout_port_list;
    ……. statements…………
endmodule
```

Example

```
module add_sub (add, in1, in2, oot);
input add;//defaults to wire
input [7:0] in1, in2, wire in1, in2;
output [7:0] oot;
reg oot;
          ………statements…….
endmodule
```

Verilog Modelling

Verilog has four levels of modelling:

1) The switch level Modelling.
2) Gate- level Modelling.
3) The Data-Flow level.

4) The behavioural or procedural level.

1.Switch level Modelling:

By explicitly demonstrating how to build a circuit utilising transistor like PMOS and NMOS, preset modules, a circuit is defined. Between the logic and analogue-transistor levels of abstraction, the switch level of modelling offers a level of abstraction. It describes how transmission gates, which are abstractions of specific MOS and CMOS transistors, are connected. Transistors at the switch level are modelled as either conducting or not conducting, on or off. The values carried by the interconnections are reduced to a handful of discrete values from the full range of analogue voltages or currents. Signal strengths are the names given to these quantities.

Example

module inverter (out,in);

output out;

input in;

Supply0 gnd;

Supply1 vdd;

nmos x1 (out, in, gnd);

pmos x2 (out, in, vdd);

endmodule

***2.*Gate level modelling**

By explicitly demonstrating how to fix a circuit using logic gates, a circuit is defined. modules with predetermined connections between them. In the first, we consider our circuit to be a box or module that is isolated from its external environment and only communicates with it through its input and output ports. Then, by explicitly specifying the module's gates and submodules, as well as how they connect to the module ports and one another, we set out to define the structure within the module. In other words, the circuit's schematic diagram is created using structural modelling. Think about the full-adder below as an example.

Example

module fulladder (a, b, sum, Cout);

input a, b;

output sum, Cout;

xor x1(a, b, y);

xor x2(a, b, y);

endmodule

3. Data-flow modelling

Boolean operators and expressions are used in dataflow modelling. We utilise the assign statement in this. Continuous assignments and the phrase assign are used in dataflow modelling. A statement that gives a value to a net is known as a continuous assignment. In Verilog HDL, the datatype net is used to describe a physical link between circuit components. An expression that makes use of operands and operators specifies the value that is allocated to the net.

Example

module fulladder (a, b, sum, Cout);

input a, b;

output sum, Cout;

assign sum=a^b;

assign Cout=a^b;

endmodule

4. Behavioral modelling

The behaviour of logic is modelled at a higher degree of modelling. There is an exception to the rule that Verilog behavioural code only exists inside procedure blocks: certain behavioural code also exists outside of them. In Verilog, there are two different kinds of procedural blocks. An algorithmic system is described at this level. (Behavioural). Each algorithm is sequential, which means that each of its instructions is carried out one at a time. The primary components are blocks, tasks, and functions. The design's structural execution is not taken into consideration.

Initial: initial blocks execute only once at time zero (start execution at time zero)

Always: always blocks loop to execute repeatedly; in other words, as other words as the name suggests, it executes always.

An always statement executes repeatedly, it starts and its execution at other 0 ns

Syntax: *always*@ (sensitivity list)

Begin

Procedural statements

end

Example

module fulladder (a, b, clk, sum);

input a, b, clk;

output sum;

always@ **(**posedgeclk)

begin

sum= a+b;

endmodule

Lexical Tokens

The source text files for the Verilog language are a stream of lexical tokens. One or more characters make up a token, and each character appears in exactly one token.

The Verilog HDL uses lexical tokens that are similar to those found in the C programming language. Verilog respects capitalization. The crucial words are all written in lower case.

White Space

Characters for spaces, tabs, new lines, and form feeds can be placed in white spaces. Except when they are used to divide tokens, these characters are ignored.

Tab, carriage returns, new lines, blank spaces, and form feeds are examples of white space characters.

Comments

There are two forms to represent the comments

1) Single line comments begin with the token // and end with carriage return.

Ex.: //this is single line syntax

2) Multiline comments begin with the token /* and end with token */

Ex.: /* this is multiline Syntax*/

Numbers

The format of a number can be specified as binary, octal, decimal, or hexadecimal. The compliment numbers of 2 includes negative values. Integers, real numbers, and signed and unsigned numbers are all acceptable in Verilog.

The syntax is as follows: size, radix, and value.

Size can describe a sized or unsized integer, and radix can indicate whether the number is binary, octal, hexadecimal, or decimal.

Identifiers

The name given to an object, such as a function, module, or register, is its identifier. Identifiers ought to start with an alphabetic character or an underscore. Example: A_Z, A_Z, _

A combination of alphabetic, numeric, underscore, and $ characters make up identifiers. They have a maximum character count of 1024.

Operators

Operators are unique characters that are used to specify conditions or control variables. To operate on variables, one, two, and occasionally three characters are utilised.

Ex. >, +, ~, &! =.

Verilog Keywords

The Verilog keywords are words used in Verilog that have specific meanings. Assign, case, while, wire, reg, and, or, nand, and module are a few examples. They must not be used as names or numbers. Compiler directives, system functions, and tasks and tasks are additional Verilog terms.

SOFTWARE DESIGN AND DEVELOPMENT

A high-level hardware description language (often VHDL or Verilog) is used to write a description of the structure and behaviour of the hardware. These codes are then built and downloaded before being executed. Although schematic capture can be used for design entry, it has lost favour as a result of more complicated designs and the advancement of language-based tools.

The approach a developer must take to the problem is where hardware and software design diverge most noticeably. Even when asked to programme a multithreaded application, software professionals frequently think sequentially. The source code is typically always performed in that order. Hardware designers must think and programme simultaneously at the beginning of the design process.

Each ore contains a sequence of macro cells and linkages that are channelled towards their destination output signals, processing all of the input signals simultaneously. As a result, when a hardware description language is used, structures are created and processed simultaneously. (Normally the link between each macro cell to another macro cell usually -synchronized to some other signal, like a common clock).

The next phase in a typical design is to run functional simulations for a set amount of time once each design entry is finished. A simulator can be useful in situations like these. It is employed to carry out the design and verify that the necessary outputs are generated for a certain set of test inputs.

Before moving on to the next stage of development, this phase is to make sure the designer's logic is functionally sound. As opposed to modelling a full-scale design entry, this is a smart practise. The troubleshooting will become considerably more challenging and time-consuming as the design entry becomes more sophisticated.

SOFTWARE TOOLS USED

Xilinx Vivado

VIVADO gives designers the ability to synthesise their designs, carry out timing analysis, go through RTL diagrams, simulate a design's response to various stimuli, and code the target device. A design environment for FPGA products from Xilinx, Vivado is incompatible with FPGA products from other manufacturers since it is strongly connected to the architecture of these chips.

Programs written in C, C++, and System C can be directly targeted at Xilinx devices without manually creating RTL thanks to the Vivado High-Level Synthesis compiler. It has been established that Vivado HLS supports C++ classes, templates, functions, and operator overloading, and that it improves developer efficiency.

Xilinx vivado enables simulation, verification, and synthesis for the following languages

1. VHDL
2. System Verilog
3. Verilog

B. XILINX VIVADO

XILINX VIVADO DESIGN SUITE

Digital circuit designs are created, synthesized, simulated, tested, and verified using the robust software programme Xilinx. A hardware description language or the schematic entry tool can be used by the designer to define the digital design. To ensure that the design operates as intended, we will use this programme to create Verilog design input files, which are the hardware descriptions of the logic circuit. We will also compile Verilog source files, build a test bench, and simulate the design. (Functional simulation). The goal of this is to introduce new users to the fundamental and essential processes needed to create and evaluate your own concepts using the Vivado environment. In the Xilinx environment, a Verilog input file consists of: Module declarations include the module name, interface details (I/O), a list of the ports' modes—the direction of data flow—and data types. Logic operation of a component is defined by its architecture. For the body of architecture, there are various styles: A set of sequential assignment statements for behaviour; a set of concurrent assignments for data flow; a set of interconnected components for structure; and a gate level modelling for structural analysis. These styles can also be combined to simulate a Verilog file. The American technology company Xilinx specialises in providing programmable logic chips. It is credited with creating the FPGA.

By opening the Xilinx Vivado design suite, three options will be displayed. They are

1) Quick start
2) Tasks
3) Learning Center

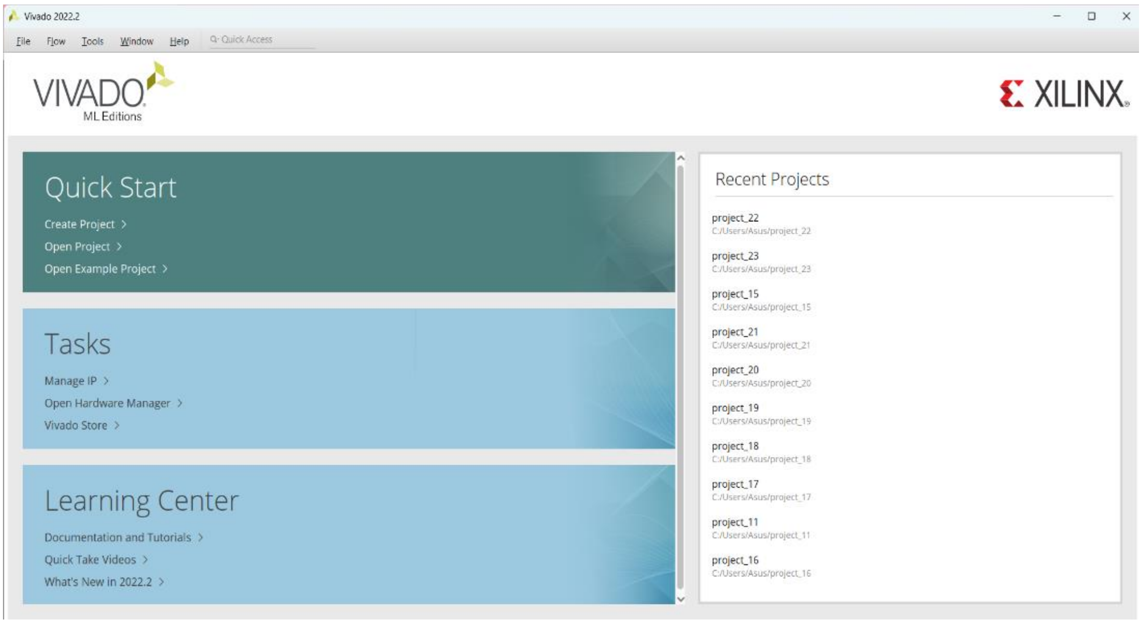


Fig-22: Xilinx Vivado Project Navigator

In the Quick start block, we can create a new project or open an existing project or open an example project.

In Tasks, we have Manage IP, open hardware manager, Xilinx Td store.

In the Learning center, we have documentation and tutorials, quick take videos and any new updates regarding the Vivado.

There are several steps for simulating a Verilog HDL file in Xilinx Vivado. They are:

Creating a New Project

After opening Xilinx Vivado, project navigator will be displayed. In that, by clicking on "create new project", a new project will be created.

The New Project wizard will launch, click the "Next >" button to proceed

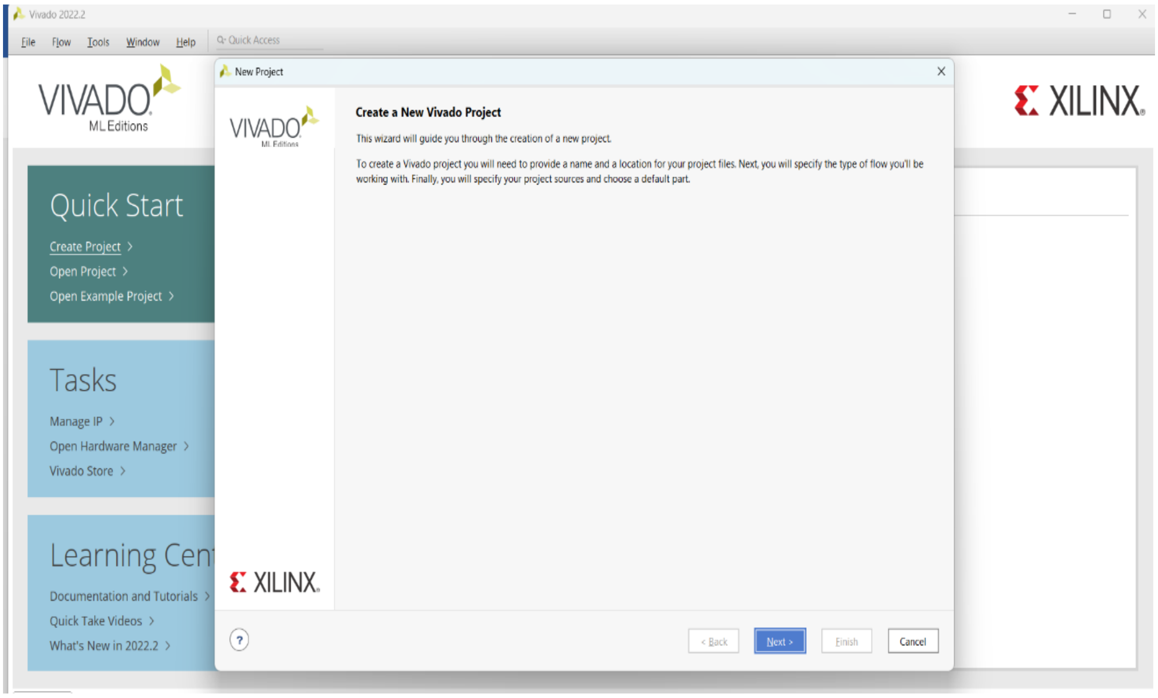


Fig-23: Guiding Wizard for the project

Choose a project location and enter the project name. Make sure neither has any empty spaces! Simply make a new directory for your Xilinx Vivado projects in your root disc (for example, C:Vivado) if necessary. The "Create project sub-directory" check-box should also always be chosen. With a directory for each project, this keeps things orderly and helps prevent issues. To continue, click the "Next >" button.

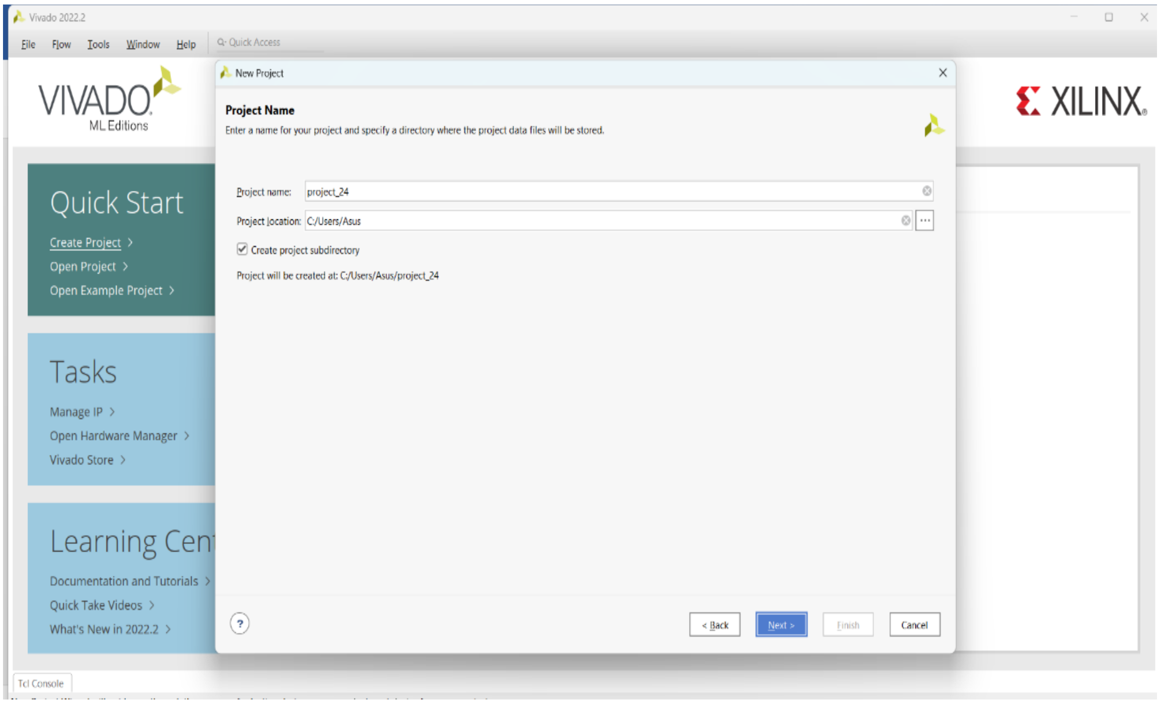


Fig-24: Creating a new project name

Choose the "RTL Project" radial, then check the box that says "Do not specify sources at this time." The wizard will walk you through further steps to opportunistically add existing objects like VHDL or Verilog source files, Vivado IP blocks, and if you don't check the box. For device pin and timing setting, use XDC constraint files. You can add the necessary components for this first project later. To continue, click the "Next >" button.

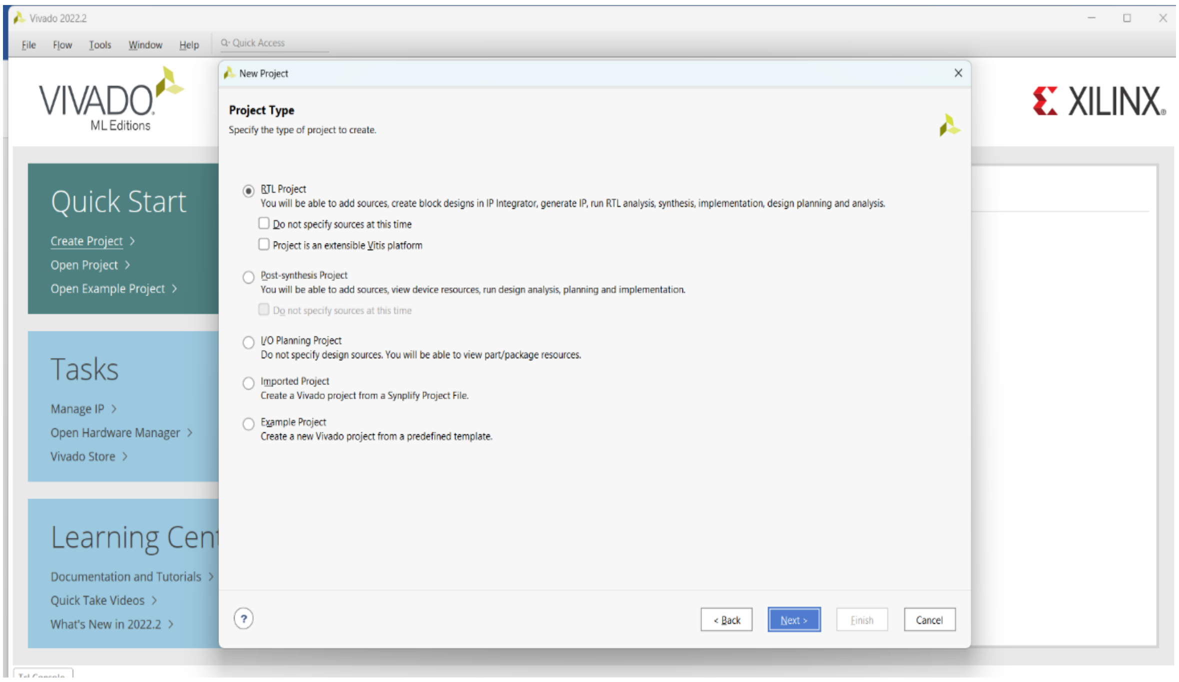


Fig-25: Specifying the RTL project

Now, we must select a default part or board for the project. There will be several options like Category, family, package, speed etc. which are required to specify the board for the project. Once you select the correct device click the "Next >" button to proceed.

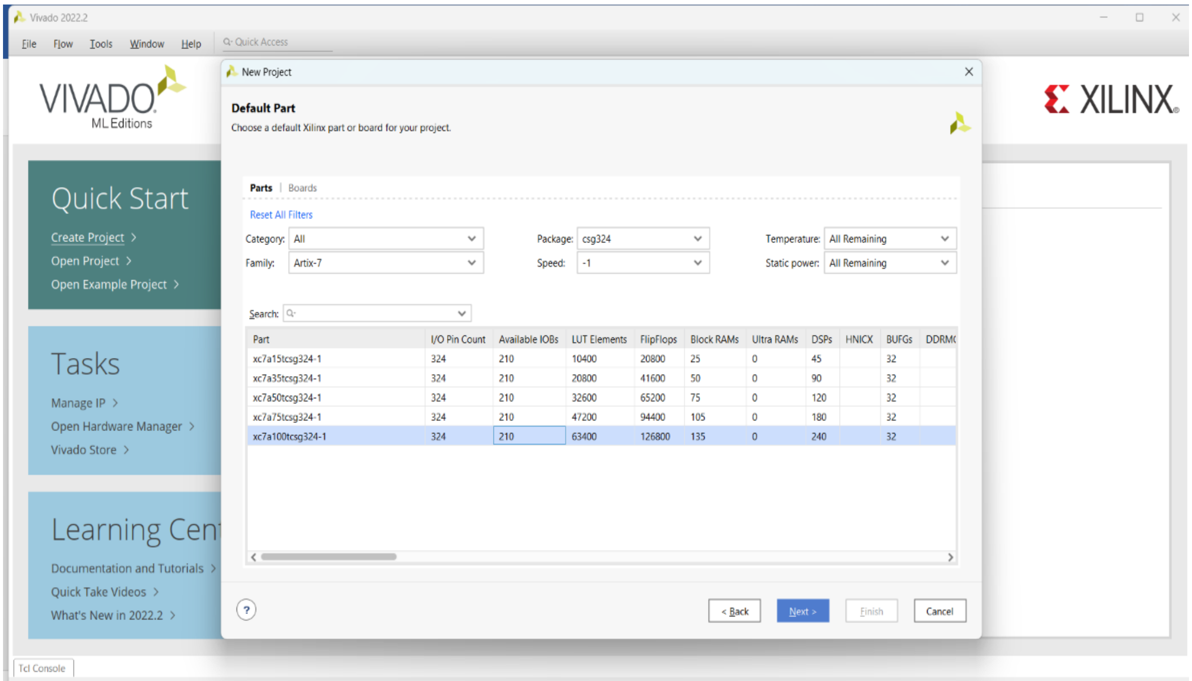


Fig-26: Choosing a board for the project

Then the project summary of the created project will be displayed. Click the "Finish" button and Vivado will proceed to create your project as specified.

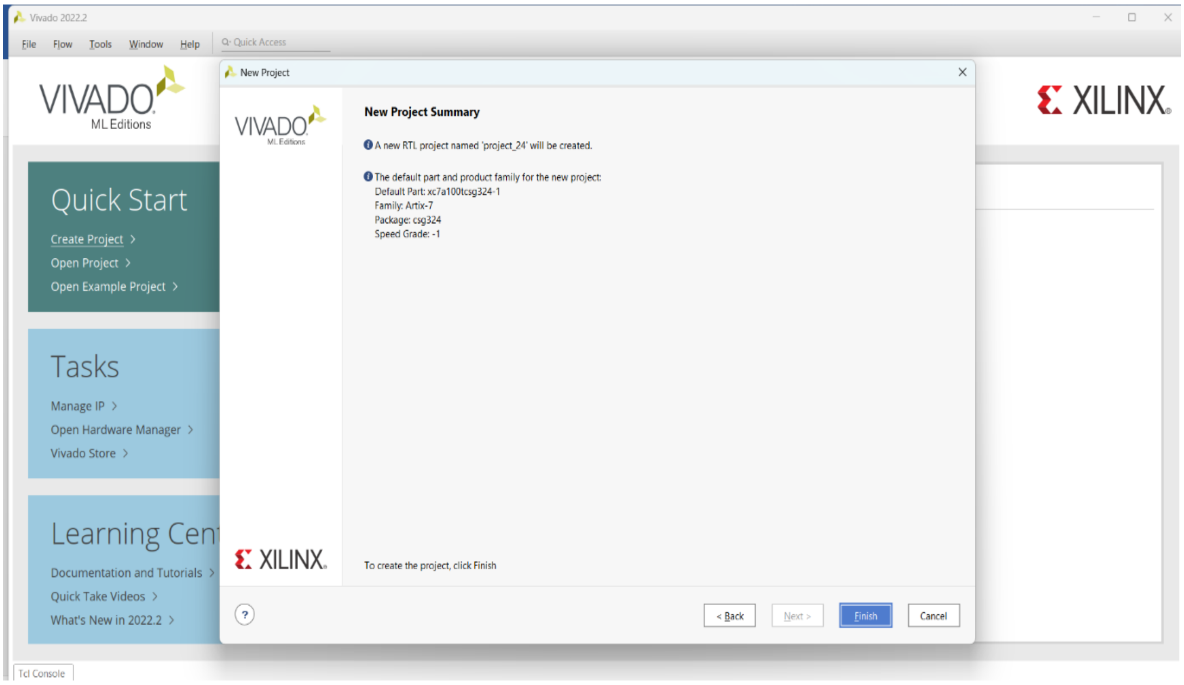


Fig-27: Project Summary

Design entry

Working through the Basic Project Flow

As you progress through the steps of your project, the information displayed in the Vivado project window can alter based on what area of the design you are now viewing. As you read through this guide, keep this in mind because it's conceivable that you're not in the right area of the design if you don't see a certain sub-window or sub-window tab.

All of the main project phases are arranged in the "Flow Navigator" on the left side of the screen from top to bottom in the order that they naturally occur. The banner at the top of the screen adjacent to the Flow Navigator indicates that you are starting in the "Project Manager" section of the flow. Which phase of the design is open is shown by this header and the correspondingly highlighted area in the Flow Navigator.

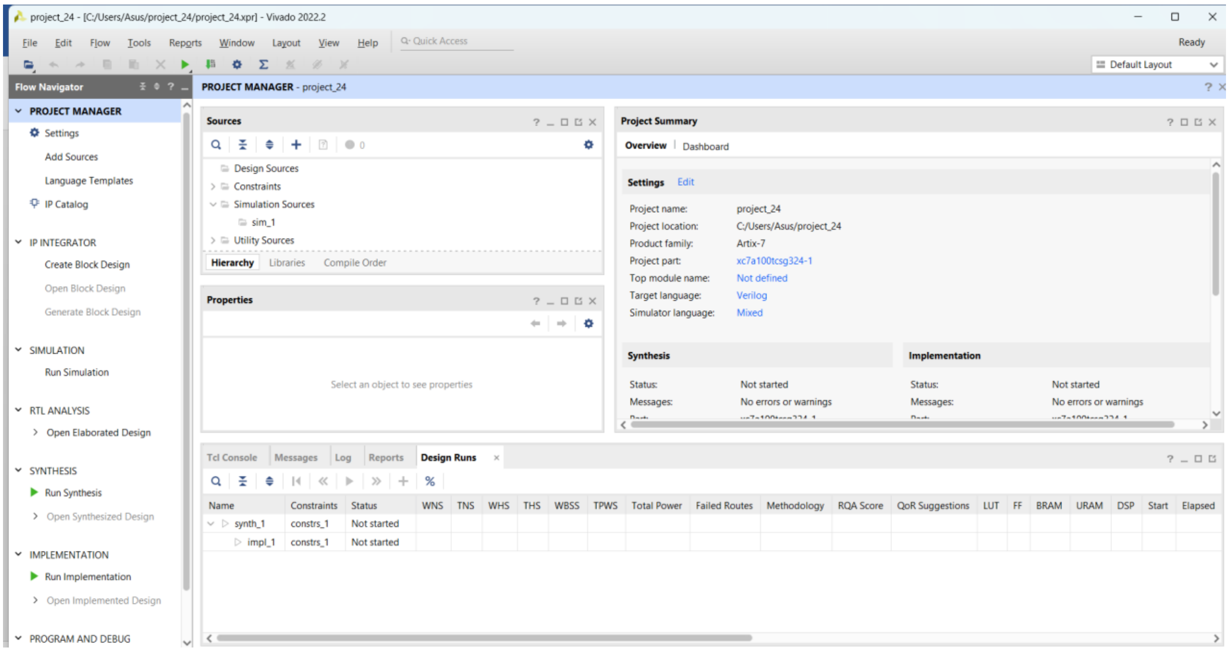


Fig-28: Main window for the project

Project Settings:

Begin by clicking on "Project Settings" under the Project Manager phase of the Flow Navigator

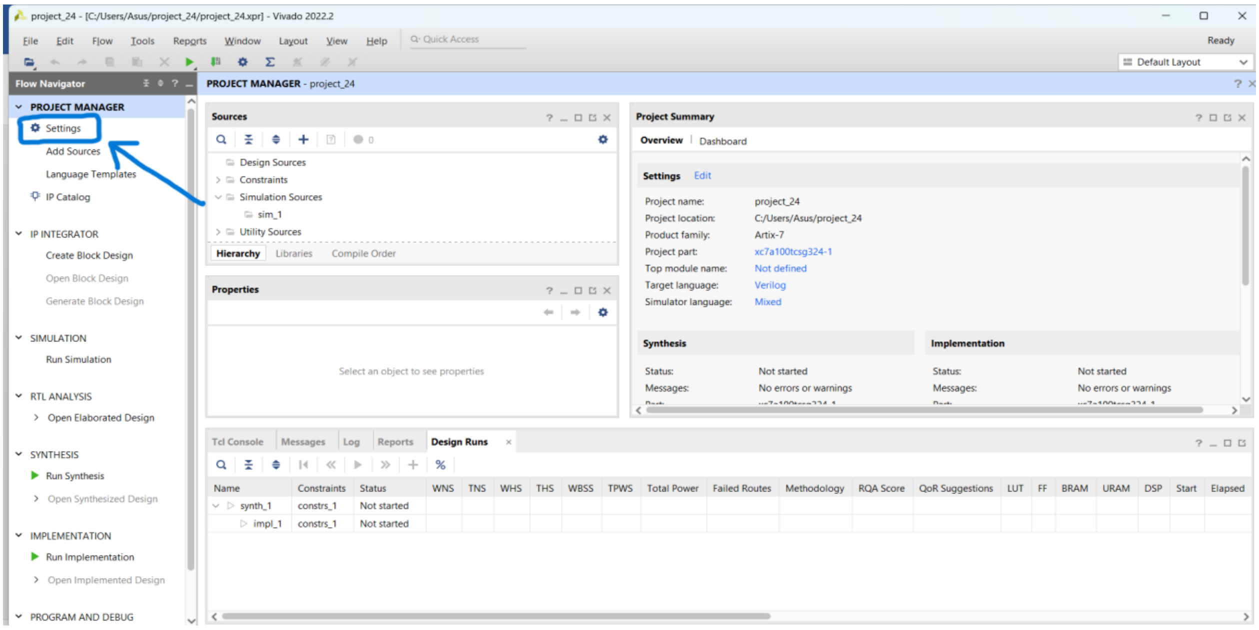


Fig-29: Project Settings window

There are a lot of choices available here for all phases of the project flow, but for now simply select “Verilog” from the drop-down for the “Target language” in the “General” project settings and click the “OK” button.

Add Sources

Now click on “Add Sources” under the Project Manager phase of the Flow Navigator

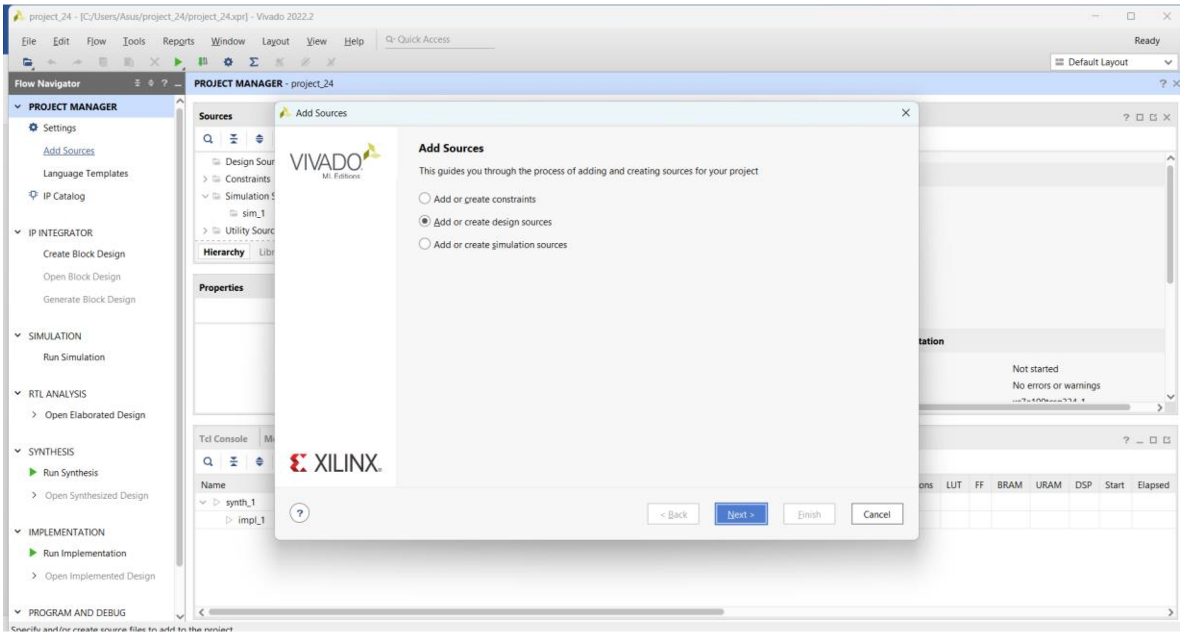


Fig-30: Adding the source files

Select the “Add or create design sources” radial and then click the “Next >” button

Click the “Create File” button or click the green “+” symbol in the upper left corner and select the “Create File… option

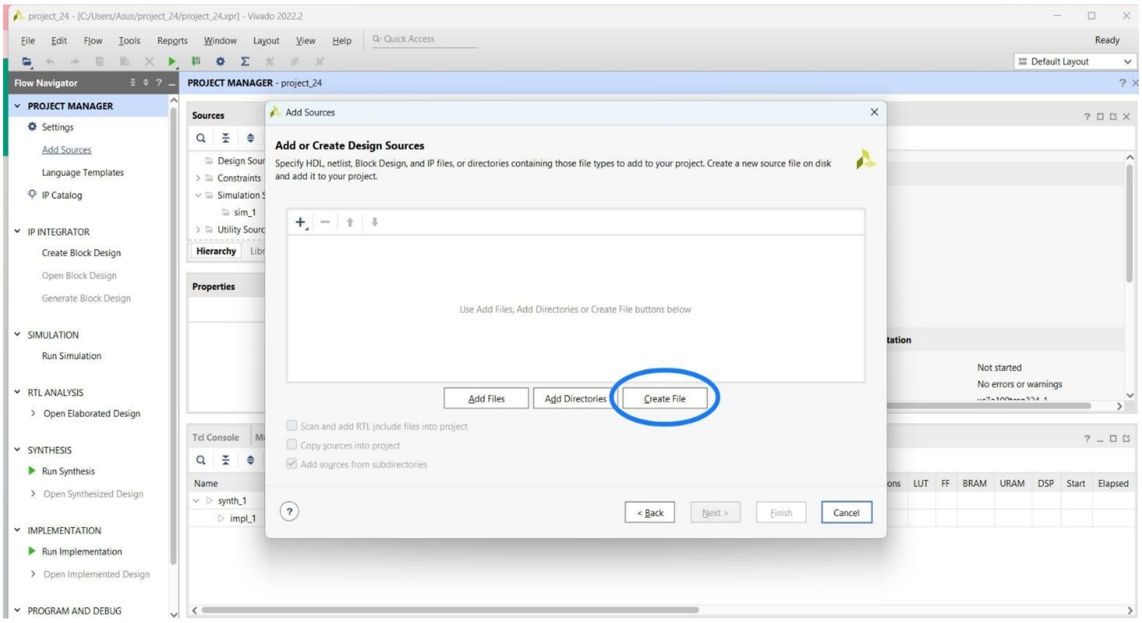


Fig-31: Creating a new file name for design source

Next, give a proper name to the file and then select a location to store the file and then click “OK” to proceed.

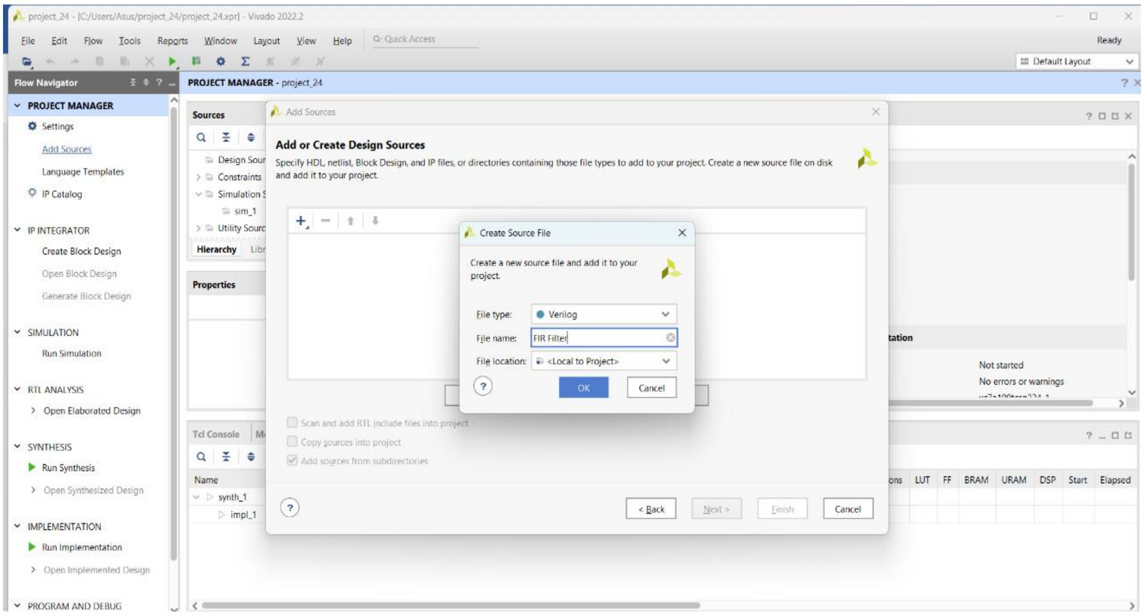


Fig-32: Naming the file and specifying the location

Click the “Finish” button and Vivado will then bring up the “Define Module” window.

Define Module

Some of the Verilog code can be automatically written for you using the "Define Module" box. You can add more "I/O Port Definitions" by clicking the green "+" sign in the top left corner or by merely clicking on the next empty line. When done, click the "OK" button. Alternately, you can merely click the "Cancel" button, in which case Vivado will produce a Verilog source file inside of your project that is entirely empty. The fundamental Verilog HDL code structure will still be written by Vivado if you click the "OK" button without declaring any "I/O Port Definitions," but the port definition will be left blank and commented out for you to fill in later.

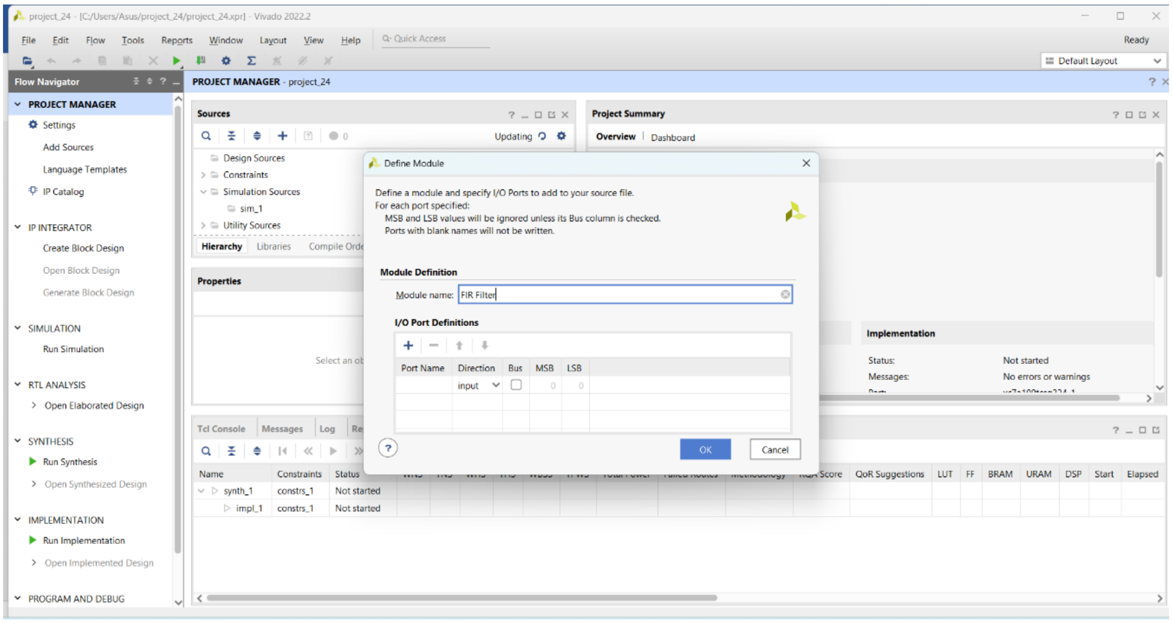


Fig-33: Module defining with ports

A module with the specified ports will be created and then write the Verilog code for the required design and click on “Run simulation” to get the simulation results for the design.

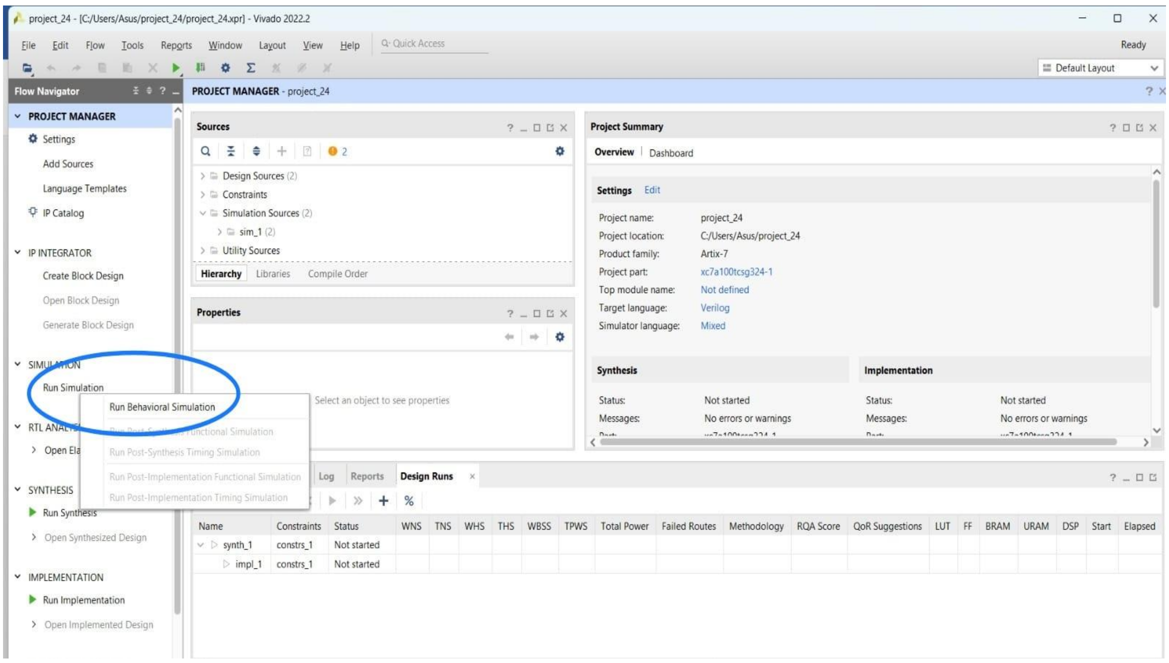


Fig-34: Simulating the design

REFERENCES

[1] John Y. Chen "Transformation of VLSI Technologies, Systems and Applications, The rise of Foundry and its Ecosystem" 2013 IEEE International Symposium on VLSI technology, Systems and Application (VLSI-TSA), 22 April 2013.

[2] Yanling Zhou, Yunyao Yan and Wei Yan" A method to speed up VLSI hierarchical physical design in floorplanning", 2017 IEEE 12th International Conference on ASIC (ASICON), 25 October 2017.

[3] Mohammed Shoaib, Noor Mahammad Sk and V Kamakoti, "A genetic approach to gateless custom VLSI design flow" 2007 IEEE International Conference on Microelectronics, 29 December 2007.

[4] Hsien-Hsin S. Lee, "IC design challenges and opportunities for advanced process technology", IEEE Conference on VLSI design, Automation and Test (VLSI-DAT), 27 April 2015.

[5] Laura wang and Matt Luo, "Machine Learning Applications and Opportunities in IC Design Flow" ,2019 IEEE International Symposium on VLSI Design, Automation and Test (VLSI- DAT), 22 April 2019.

[6] R. Jain, C. Chien, E. Cohen, and L. Ho, "Simulation and Synthesis of VLSI Communication Systems", 1998 IEEE Proceedings Eleventh International Conference on VLSI Design, 4 January 1998.

[7] K. A. Wen, J. Y. Lee, J.F. Wang, and C. S. Wu, "Design of VLSI implemented signal processing systems based on signal flow graph analyses", 1988 IEEE International Symposium on Circuits and Systems (ISCAS), 7 June 1988.

[8] L. Merani, S.-L. Lu, "A self-timed approach to VLSI digital filter design", 1993 Proceedings of IEEE pacific rim conference on Communications Computers and Signal Processing, 19 May 1993.

[9] Deepa Yagain and Krishna A. Vijaya, "FIR Filter design based on retiming automation using VLSI design metrics", 2013 IEEE International Conference on Technology, Informatics, Management, Engineering and Environment, 23 June 2013.

[10] Manish B. Trimale and Chilveri, "A Review: FIR Filter Implementation", 2017 2nd IEEE International Conference on Recent Trends In Electronics Information & Communication Technology, 19 May 2017.

[11] Ranjushree Pal, "Comparison of the design of FIR and IIR filters for a given specification and removal of phase distortion from IIR filters", 2017 International Conference on Advances in Computing, Communication and Control (ICAC3), 01 December 2017.

[12] S. Akash, M. Ajeeth and N. Radha, "An Efficient Implementation of FIR Filter using high speed adders for Signal Processing Applications", 2020 second International Conference on Inventing research in Computing Applications (ICIRCA), 15 July 2020.

[13] Shuchi Nagaria, Anushka Singh, Vandana Niranjan, “Efficient FIR Filter design using Booth Multiplier for VLSI Applications”, 2018 IEEE Conference on Computing, Power, and Communication Technologies (GUCON), 28 September 2018.

[14] Deepshika Bharthi and Kumari Nidhi Gupta, “Efficient Design of different forms of FIR filter”, 2014 IEEE conference on recent trends in information technology, 10 April 2014.

[15] Hui Zhao and Juebang Yu, “A simple and efficient design of variable fractional delay FIR Filters”, 2013 IEEE Transactions on Circuits and Systems-II: Express Briefs (Volume:53 , Issue:2, February 2006).

[16] N. Sameeksha Rai, Pannaga Shree B. S, Meghana Y.P, Arunkumar P. Chavan and H. V. Ravish Aradhya, “Design and Implementation of 16 tap FIR Filter for DSP Applications”, 2018 Second International Conference on Advances in Electronics, Computers and Communications (ICAECC) , 9 February 2018.

[17] Jiajia Chen, Jinghong Tan, Chip-Hong Chang and Feng Feng , “ A new cost-aware Sensitivity-Driven Algorithm for the design of FIR Filters”, IEEE Transaction on Circuits and Systems I: Regular Papers (Volume: 64, Issue: 6, June 2017).

[18] Sung Hyun Yoon and M. H. Sunwoo, “An efficient Multiplier less FIR Filter chip with variable-length taps”, IEEE Proceedings of 1998 Asia and South Pacific Design Automation Conference, 13 February 1998.

[19] K. Seinski and M. Legako, “Hardware Efficient FIR Filter structures from linear program design constraints”,1996 IEEE International Conference on Acoustics, Speech and Signal Processing Conference Proceedings, 9 May 1996.

[20] Hon Keung Kwan and Jiajun Liang, “Minimax design of linear phase FIR filters using Cuckoo search algorithm”, 2016 8th IEEE international conference on wireless communications and signal processing (WCSP) ,13 October 2016.

[21] Juan Zhao, Yujia Wang, Jiajia Chen and Feng Feng, “A new area-efficient FIR Filter design algorithm by dynamic programming”, 2016 24th European Signal Processing Conference (EUSIPCO) , 29 August 2016.

[22] Hareesh Khattri, Narasimha Kumar V Mangipudi and Salvador Mandujano, “HSDL: A Security Development Lifecycle for hardware technologies”, 2012 IEEE International Symposium on Hardware-Oriented Security and Trust, 3 June 2012.

[23] Wei Hu, Chip-Hong Chang, Anirban Sengupta and Swarup Bhunia, “An overview of Hardware Security and Trust: Threats, countermeasures, and Design Tools”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (Volume :40, Issue:6, June 2021), 29 December 2021.

[24] Hongbo Gao and Zhiyong Jiao, “Hardware Threat: The Challenge of Information Security”, 2008 IEEE International Symposium on Computer Science and Computational Technology, 20 December 2008.

[25] Rajat Subhra Chakraborty, Seetharam Narasimhan and Swarup Bhunia, “ Hardware Trojan: Threats and Emerging Solutions”, 2009 IEEE International High Level Design Validation and Test Workshop, 4 November 2009.

[26] Nachiketh Potlapally, “Hardware Security in Practice: Challenges and Opportunities”, 2011 IEEE International Symposium on Hardware-Oriented Security and Trust, 5 June 2011.

[27] Jiliang Zhang, “A Practical Logic Obfuscation Technique for Hardware Security”, IEEE Transactions on Very Large Scale Integration (VLSI) Systems (Volume:24, Issue :3, March 2016) , 18 June 2016.

[28] Hassan Salmani and Mark M. Tehranipoor, “Vulnerability Analysis of a Circuit Layout to Hardware Trojan Insertion”, IEEE Transactions on Information Forensics and Security (Volume:11, Issue :6 , June 2016) , 22 January 2016.

[29] Hassan Salmani and Mohammad Teharnipoor, “A novel Technique for Improving Hardware Trojan Detection and Reducing Trojan Activation Time”, IEEE Transactions on Very Large Scale Integrated (VLSI) systems (Voulem:20, Issue:1, January 2012), 6 January 2012.

[30] Maoyuan Qin and Wei Hu, “Property based Formal Security Verification for Hardware Trojan”, 2018 IEEE 3rd International Verification and Security Workshop (IVSW) ,2 July 2018.

[31] Sreeja Rajendran and Mary Lourde Regeena, “A novel algorithm for hardware trojan detection through reverse engineering”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (Voulme: 41, Issue :4, April 2022), 20 April 2021.

[32] Chris Nigh and Alex Orailoglu, “Ada Trust: Combinational Hardware Trojan detection Through Adaptive Test Pattern construction”, IEEE Transactions on Very Large Scale Integrated (VLSI) Systems (Volume:29, Issue:3, March 2021), 10 February 2021.

[33] Rajat Subhra Chakraborty and Swarup Bhunia, “Security against hardware trojan through a novel application of design obfuscation”, 2009 IEEE/ACM International Conference on Computer-Aided Design-Digest of Technical papers, 2 November 2009.

[34] N. Nalla Anand Kumar and Mohammad S. Hashmi, “Design, implementation and analysis of Efficient Hardware- Based Security Primitives”, 2020 IFIP/IEEE 28th International Conference on Very Large Scale Integration (VLSI-SOC), 5 October 2020.

[35] Koustav Dey and Malay Kule, “PUF based Hardware security: A Review”, 2021 International Symposium on Devices, Circuits and Systems (ISDCS), 3 March 2021.

[36] Charles Herder and Srinivas Devadas, “Physical Unclonable Functions and Applications: Tutorial”, 2014 proceedings of the IEEE (Volume:102, Issue:8, August 2014), 30 May 2014.

[37] Fahem Zerrouki and Samir Ouchani, “Quantifying security and Performance of Physical Unclonable Functions”, 2020 7th International Conference on Internet of Things (IOT): Systems, management, and security (IOTSMS), 2 February 2021.

[38] Mary Latha Ch. and L. Abhikshit, “Design and Implementation of a secure Physical Unclonable Function in FPGA”, 2020 2nd International Conference on Inventive Research in computing Applications, 15 July 2020.

[39] Axel Sikora and Achim Bittner, “Design, Simulation and Analysis of Physical Unclonable Functions with MEMS AIN Cantilevers”, 2022 IEEE Conference on Smart System Integration (SSI), 27 April 2022.

[40] Rivaldo Ludovicus Sembiring and Parman Sukarno, “Randomness, Uniqueness and Steadiness Evaluation of Physical Unclonable Functions”, 2021 9th IEEE International Conference on Information and Communication Technology (ICoICT), 5 August 2021.

[41] Durba Chatterjee and Aritra Hazra, “Formal Analysis of Physical Unclonable Functions”, 2021 IEEE/IFIP 29th International Conference on Very Large-Scale Integration (VLSI-SOC) ,4 October 2021.

[42] Josef Biba and Walter Sanch, “Measurement Setup for Physical Unclonable Functions”, 2021 6th International Conference on Integrated Circuits and Microsystems (ICICM), 22 October 2021.

[43] Onur Gunlu and Rafael F. Schaefer, “Low Complexity and Reliable Transforms for Physical Unclonable Functions”, ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 4 May 2020.

[44] Pooja Lokhande and A. M. Shah, “Strong Authentication and Encryption Modelling using Physical Unclonable Function based on FPGA”, 2021 6th International Conference on Communication and Electronics Systems (ICCES), 08 July 2021.

[45] Frederic Rizk and Ashok Kumar, “A cost efficient Reversible Based Configurable Ring Oscillator Physical Unclonable Function”, 2021 IEEE 34th International Conference on System-on-chip Conference (SOCC), 14 September 2021.

[46] Sandeep S. Kumar and Roel Maes, “Extended Abstract: The butterfly PUF protecting IP on every FPGA”, 2008 IEEE International Workshop on Hardware Oriented Security and Trust, 9 June 2008.