ADC IP BLOCK DESIGN AND VERIFICATION

Major Project Report

Submitted in fulfillment of the requirements for the degree of

Master of Technology
in
Electronics & Communication Engineering
(VLSI Design)

By

Sachin Darji

(22MECV04)

NIRMA UNIVERSITY
INSTITUTE OF TECHNOLOGY
NAAC ACCREDITED 'A+' GRADE

Electronics & Communication Engineering Department
Institute of Technology
Nirma University
Ahmedabad-382 481

December 2023

ADC IP BLOCK DESIGN AND VERIFICATION

Major Project Report

Submitted in partial fulfillment of the

requirements for the degree of

Master of Technology

in

Electronics & Communication Engineering

By

Sachin Darji (22MECV04)

Under the guidance of

Internal Project Guide:

Dr. Jayesh Patel

Asso. Professor, EC Department,

Institute of Technology,

Nirma University, Ahmedabad.

Electronics & Communication Engineering Department

Institute of Technology-Nirma University

Ahmedabad-382 481

December 2023

Declaration

This is to certify that

a. The thesis comprises my original work towards the degree of Master of Technology in VLSI Design at Nirma University and has not been submitted elsewhere for a degree.

b. Due acknowledgment has been made in the text to all other material used.

- Sachin Darji

(22MECV04)

NIRMA UNIVERSITY

INSTITUTE OF TECHNOLOGY

NAAC ACCREDITED 'A+' GRADE

Certificate

This is to certify that the Major Project entitled **ADC IP Block Design and Verification submitted by Sachin Darji (22MECV04)**, towards the partial fulfillment of the requirements for the degree of Master of Technology in VLSI Design, Nirma University, Ahmedabad is the record of work carried out by him under our supervision and guidance. In our opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of our knowledge, haven't been submitted to any other university or institution for award of any degree.

Date:

Place: Ahmedabad

Prof. Jayesh Patel
Internal Guide,
Assistant Professor,
Institute of Technology,
Nirma University, Ahmedabad

Dr. Usha mehta
Program Coordinator,
Professor,
Institute of Technology,
Nirma University, Ahmedabad

Dr. Usha Mehta
Professor and Head,
EC Department,
Institute of Technology,
Nirma University, Ahmedabad.

Dr. Rajesh Patel
Director,
Institute of Technology,
Nirma University, Ahmedabad

Acknowledgments

I am immensely grateful to Dr. Usha Mehta, the PG Coordinator for M.Tech VLSI Design, for her invaluable guidance during the review process.

I want to extend my heartfelt thanks to Dr. Jayesh Patel, who served as my mentor for the in-house project, for his exceptional support, continuous monitoring, and unwavering motivation.

- Sachin Darji

(22MECV04)

ABSTRACT

This thesis navigates the realm of VLSI design with a focal point on Analog-to-Digital Converter (ADC) Intellectual Property (IP) blocks. Beginning with exploring the realm of semiconductor intellectual property, the research highlights the evolution of IP cores and their pivotal role in VLSI chip production. Delving deeper, the thesis elucidates the imperative need for IP designs in VLSI, emphasizing their reusability, legal implications, and substantial impact on reducing time-to-market for products.

The investigation revolves around the design lifecycle in VLSI, focusing on ADC IP blocks as pivotal components. Exploring the Flash ADC architecture, the study dives into the intricate analog front-end design, emphasizing the Threshold Inverter Quantization (TIQ) methodology, its principles, advantages, and challenges. Moreover, the digital back-end design phase scrutinizes the priority encoder implementation, leveraging Verilog/SystemVerilog for efficient translation of analog data to digital formats.

Critical to this study is the exploration of mixed-signal simulations, meticulously integrating analog and digital domains using Cadence Spectre and AMS tools. Detailed analyses encompass transient simulations, DC/AC analysis, Monte Carlo simulations, and sensitivity analyses, ensuring robustness and accuracy across the entire ADC system.

The thesis converges on the amalgamation of analog mixed-signal verification and integration phases, stressing functional, timing, noise, and power consumption verifications. Challenges spanning precision, power, layout complexities, and technology scaling are meticulously navigated to optimize ADC IP block performance and reliability.

This study unravels the complexities of ADC IP block design within the VLSI domain, highlighting the intricate interplay between analog and digital realms, underscoring the significance of IP cores in advancing VLSI semiconductor designs, and addressing critical challenges in achieving high-performance ADC IP blocks.

LIST OF FIGURES

Chapter 1:

Introduction

1.1 Introduction

In the modern semiconductor industry IP (Intellectual Property) based VLSI design is trending. This in the Semiconductor industry has become important to enhance VLSI chip production. These designs are largely reusable in terms of logic function and design layout. It with Verification capability is termed as VIPs.

In this thesis, we will try to understand the design life cycle in VLSI chip design. A semiconductor intellectual property (IP) core is most widely used by the VLSI Chip Design engineers in their product designs,

It has various aspects, In VLSI the IP is produced or being used based on the role of the design engineer.

Let us understand why we need IP in VLSI, its brief history, what are the different IP cores, What can be the different IP's in a design, the life cycle, and its usage across the semiconductor industry.

1.2 Introduction to IP design in VLSI

If we go to the early 1970s, the standard cells were being considered as reusable blocks as IP in VLSI. You can say the first IP design started with standard cells. Fixed height row cells and standard placement and routing algorithms were followed for these standard cells. The main purpose was to automate the design process of ASIC by these standard cells.

That led to an increase in the design costs compared to the manufacturing costs. A technique that trades between design and manufacturing was needed by having optimized low-cost re-usable standard cell designs were expected.

In today's semiconductor industry, we have IPs ranging from **standard cells** to I/O cells and processors (CPUs). The requirement for the VLSI chip engineers is increasing enormously which includes the complex level of integrations. We started expecting the VLSI IC chips to be working as systems such as System on Chips (SOCs).

1.3 Need for an IP design in VLSI

1. Designing a SOC (system on chip) mainly involves the ability of a VLSI architect who can identify a good combination of various components to place on the chip.

2. It is very important to consider these components which are open standard and must be an IP provider licensed who are great in the market.

3. It is difficult to identify and design and IP for product manufacture. He may not have the ability to design low-level standard IPs or he has to focus more on the product outcome by using existing standard reusable IPs. In case he can design an IP but can face legal problems and also patent violations.

4. Usually the current generation VLSI chip from the current generation becomes the IP core for the next generation as the technology is getting advanced very fast. Hence In VLSI, an IP core design is crucial in the VLSI semiconductor world to advance and meet the pace.

5. Current VLSI semiconductor world expects chip designers to have one or more CPUs and the cache in it to get the purpose fulfilled. It is very important to have reusable IPs in the design instead of starting from scratch every time to reduce the time to market the product.

6. Multicore processors (for example octa-core CPUs from Qualcomm) came into the market and they are dominating the IC and microprocessor world. The situation makes sense if the processors are replicated on the die itself.

1.4 Role of IP design providers in semiconductor devices

We know that the need for IP designs in today's VLSI semiconductor integrated chip designs, integrated devices relying on IPs as re-usable building blocks in the overall product design.

It is very time-consuming and not standard to create an IP in-house, which also allows us to forget thinking about main design requirements and overall system perspectives.

1. IP provider companies provide custom IP services and IP verification services which make the task easier

2. They take care of whether or not our required IP makes use of standard well-defined re-usable such as SPI interfaces or ASIC macros or Custom ADCs etc.

3. In VLSI IP providers will have design engineers who are well expertise and have complete domain knowledge who can help to achieve the desired requirements to reach a complete product.

4. IP service will also capable of providing the resources and silicon expertise needed to create an IP by using various EDA tools to analyze, simulate and verify the IP designs.

1.5 IP designs in various fields

In VLSI IP can be created in various fields of semiconductor industries to speed the design process.
A few of the IP block examples which are being spread into different fields are

1. IP blocks in High-resolution temperature sensors
2. Biomedical sensors are making use of IP blocks.
3. In the analog field such as SAR-ADCs Successive Approximation, Analog to Digital Converters used IP blocks. Several designs such as Pipeline ADC's, and Digital to Analog Converters (DAC) will include IP.
4. Voltage circuits such as Voltage references and Voltage regulators use IP.
5. Real-time clock circuits, Radio Frequency Identification (RFID) circuits, and IoT components make use of IPs.
6. Core VLSI Industries use IPs for reusing in Custom Standard cell libraries, Current Mode Logic (CML) for high-speed applications, and various CMOS, LVDS, ECL I/O cells.

1.6 Specification

This mixed-signal circuitry is designed and simulated using 90 nm CMOS technology. The 8-bit flash ADC only employs 255 comparators. The applied input clock is 80 MHz, with the input voltage ranging from 0 V to 0.8 V. The comparator block outputs 255 bits of thermometer code and sends them to the encoder.

Chapter 2:

Literature Review

Analog Design

The selection of the Flash ADC architecture for the design and verification of the ADC IP was predicated upon a comprehensive analysis of the established requirements and the inherent characteristics of various ADC architectures.

1. Speed and Simplicity: Flash ADCs are renowned for their exceptional speed due to their parallel architecture, enabling high-speed conversions in a single clock cycle. This aligns with the stringent requirement for fast sampling rates and real-time data acquisition in the intended application.

2. Resolution and Linearity: The Flash ADC architecture offers inherent high resolution owing to its wide analog input range divided into multiple comparators, ensuring excellent linearity across the entire input range. This aligns with the necessity for high precision and accuracy in the acquired data.

3. Verification and Design Complexity: While Flash ADCs present challenges in terms of power consumption and silicon area due to their requirement for an extensive number of comparators, the inherent simplicity of the architecture simplifies the design and verification processes. This reduces the overall verification effort and accelerates the design cycle, crucial in meeting project timelines.

4. Power Consumption Trade-offs: Although Flash ADCs can consume higher power compared to other architectures, the instantaneous nature of the conversion minimizes active time, allowing for potential power-saving strategies during idle periods. This

consideration was balanced against the need for real-time operation and justified within the context of the application's power constraints.

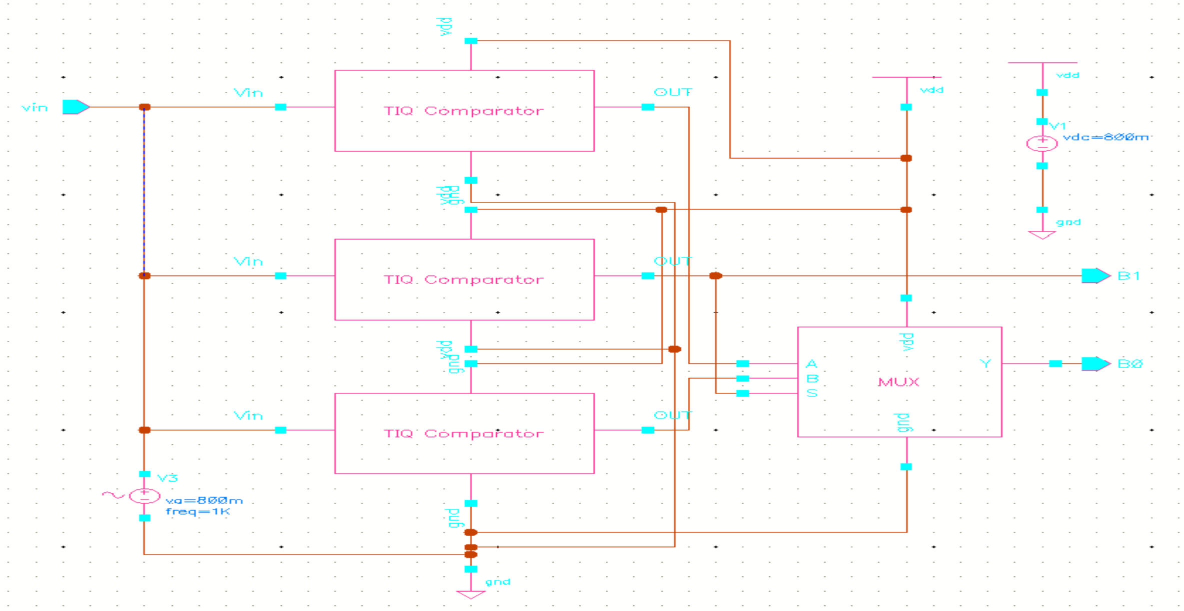


Figure 1 Flash ADC Architecture

2.1 FLASH ADC:

Analog to Digital Comparator is most probably the second most widely used electronic components after operational amplifiers in this world. Flash ADCs are most probably most widely used ADCs components in this world and for that reason they are higher speed and lower power dissipation. In the Flash ADCs conversion process, it is necessary to first Quantization the input. This Quantization signal is then applied to a combination of comparators to determine the digital equivalent of the Analog signal. The conversion speed of comparator is limited by the Register ladder. Apart from that, Fla are also can be found in many other applications like Data storage, Optical receiver, Digital multimedia application, and others. The basic functionality of a Flash ADCs is used to find out whether a signal is covert to Digital signal with respect to input signal. The schematic symbol and basic operation of a Flash ADCs are shown in Figure 1.

2.2 TIQ Comparator

Threshold Inverter Quantization (TIQ) is a methodology used in high-speed CMOS Flash Analog-to-Digital Converters (ADCs) to establish reference voltages for comparators. This technique plays a crucial role in achieving accurate analog-to-digital conversion by setting precise thresholds for the comparators within the ADC architecture.

2.2.1 Principles of TIQ:

1. Inverter-Based Thresholding: TIQ employs a network of CMOS inverters to generate reference voltages for the comparators. Each inverter stage in the network creates a specific voltage threshold.

2. Quantization Levels: The voltage levels created by the inverters form a ladder-like structure, establishing multiple quantization levels across the input range of the ADC. Each level corresponds to a specific comparator within the ADC.

3. Thermometer-to-Binary Conversion: TIQ converts the analog input voltage into a thermometer code, where each bit represents whether the input voltage exceeds a specific threshold. This thermometer code is later encoded into a binary representation.

2.2.2 Design Considerations for TIQ:

1. Resolution and Accuracy: The number of inverters in the TIQ network determines the resolution of the ADC. More inverters lead to finer quantization levels, enhancing the ADC's resolution and accuracy.

2. Speed and Power Trade-offs: The speed of the ADC is influenced by the number of inverters in the TIQ network. While more inverters improve resolution, they can also increase power consumption and introduce delays, impacting the ADC's speed.

3. Comparator Threshold Matching: Precise matching of threshold voltages across the omparators is essential for maintaining linearity and accuracy. Design techniques such as layout symmetry and voltage trimming may be employed to achieve this.

2.2.3 Advantages of TIQ:

1. High Speed: TIQ-based ADCs are known for their high-speed operation due to the parallel comparison of the input voltage against multiple thresholds simultaneously.

2. Reduced Complexity: Compared to other reference generation techniques, TIQ can offer a simpler architecture by utilizing a ladder-like structure of inverters to establish thresholds.

3. Improved Linearity: The precise voltage levels set by the TIQ network contribute to improved linearity and reduced quantization errors in the ADC's output.

2.2.4 Challenges and Optimization:

1. Power Consumption: As the number of inverters increases to enhance resolution, power consumption can rise. Design optimization techniques, such as power gating or advanced CMOS technologies, may be employed to mitigate this.

2. Layout and Matching: Achieving consistent performance across the inverters and comparators requires careful layout design and matching techniques to minimize variations and ensure accurate threshold levels.

2.3 SystemVerilog for Verification

Testbench or Verification Environment is used to check the functional correctness of the Design Under Test (DUT) by generating and driving a predefined input sequence to a design, capturing the design output and comparing with-respect-to expected output.

TestBench or Verification environment is a group of classes or components. where each component is performing a specific operation. i.e, generating stimulus, driving, monitoring, etc. and those classes will be named based on the operation.

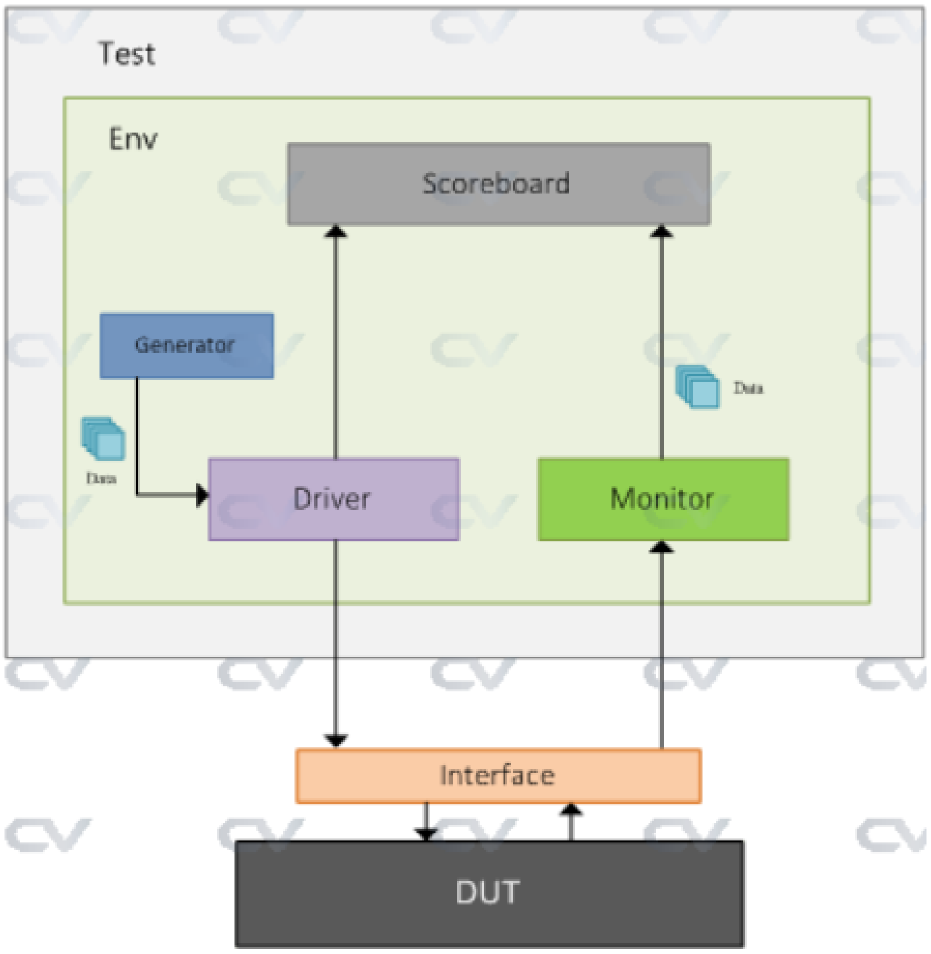


Figure 2 Systemverilog Testbench architecture

Component	Description
Generator	Generates different input stimulus to be driven to DUT
Interface	Contains design signals that can be driven or monitored
Driver	Drives the generated stimulus to the design
Monitor	Monitor the design input-output ports to capture design activity
Scoreboard	Checks output from the design with expected behavior
Environment	Contains all the verification components mentioned above
Test	Contains the environment that can be tweaked with different configuration settings

2.4 Mixed signal simulation

Modern System-on-Chip (SoC) designs involve complex interactions between analog and digital parts, making traditional design and verification methods insufficient. This complexity demands a mixed-signal verification approach, starting with simulating analog behaviors. However, merging analog and digital components causes delays in verification using fast analog circuit solvers. Cadence's mixed-signal verification method combines analog and digital sides, using models and assertions to balance speed and precision based on design needs. It offers a single environment, uniting analog and digital tools for accurate yet swift verification. This approach allows trade-offs between speed and accuracy, offers metric-driven verification, and supports power-aware verification. Cadence's advanced digital verification methodologies now apply to analog designs, ensuring comprehensive verification across the SoC. This integrated approach ensures reliability in verifying complex mixed-signal designs.

2.4.1 Steps in Mixed-Signal Simulation using Cadence Tools:

1. Schematic Entry:

- Design the mixed-signal circuit using Cadence Virtuoso for the analog part and RTL or digital design tools for digital components.

2. Mixed-Signal Simulation Setup:

- Configure simulations using tools like Spectre or AMS for mixed-signal simulations.
- Define analog and digital testbench stimuli and interactions.

3. Co-Simulation and Interfacing:

- Establish interfaces between analog and digital domains to ensure proper signal interaction.
- Use appropriate models and interfaces to bridge the gap between analog and digital worlds.

4. Verification and Analysis:

- Run mixed-signal simulations to verify functionality, accuracy, timing, noise, and power consumption across the entire system.
- Analyze simulation results to identify potential issues or areas for improvement.

2.4.2 Challenges in Mixed-Signal Simulation:

- Inter-Domain Interaction: Ensuring accurate representation and interaction between analog and digital domains.

- Model Abstraction: Managing different levels of abstraction between analog (transistor-level) and digital (RTL, gate-level) representations.

- Convergence: Achieving convergence in simulations, especially in complex mixed-signal designs.

- Simulation Time: Mixed-signal simulations might be time-consuming, especially when simulating large-scale systems.

Cadence tools provide a comprehensive environment for mixed-signal simulation, enabling engineers to analyze, verify, and optimize designs encompassing both analog and digital components. These simulations are crucial for ensuring the functionality, performance, and reliability of complex mixed-signal systems.

2.4.3 Mixed Signal Verification

Functional Verification: Ensure the ADC operates as intended under different input conditions.

Timing Verification: Validate timing requirements, like settling time, conversion time, and clock speed.

Noise Analysis: Check for noise sources and their impact on the ADC's performance.

Power Consumption Verification: Analyze power consumption to meet design constraints.

Corner Case Analysis: Test under extreme conditions (temperature, voltage, process variations).

Monte Carlo Simulation: Analyze performance variations due to manufacturing process deviations.

2.5 Documentation

Maintain detailed documentation of the entire design process, including specifications, schematics, code, simulation results, and verification procedures.

Documentation: Create comprehensive documentation specifying usage guidelines, interfaces, and characteristics.

Integrate the ADC IP into the intended application or system and validate its performance in the real-world context. Ensure it aligns with the initial specifications and requirements.

Chapter 3:

Methodology

3.1 Analog Design

3.1.1 TIQ comparator

A comparator is a fundamental component in electronic circuits used to compare two input voltages and provide an output based on their relative magnitudes. In a scenario where the width (Wp) of the p-type metal-oxide-semiconductor (PMOS) transistor and the width (Wn) of the n-type metal-oxide-semiconductor (NMOS) transistor within the comparator are both set to 120 nanometers, and the switching threshold voltage (Vswitching) is determined to be 336.69 mV, it signifies that the comparator will transition its output state when the difference between the two input voltages crosses this threshold. This specific configuration and switching voltage underline the critical role of transistor dimensions and reference voltages in determining the behavior and performance of comparators in modern integrated circuits.

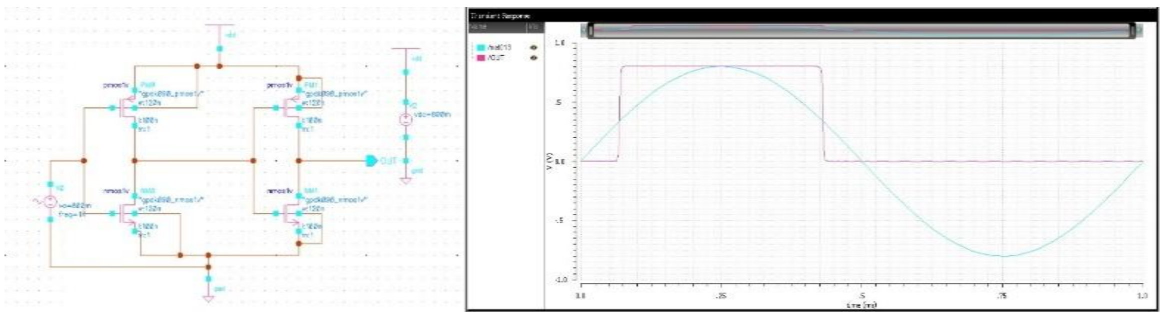


Figure 3 (a)Threshold Inverter comparator Design (b) Simulation result of Threshold Inverter comparator

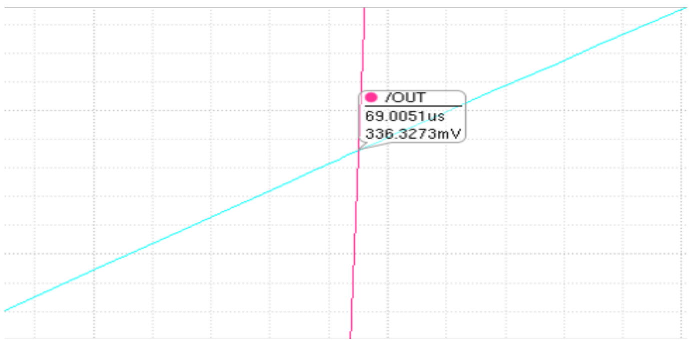


Figure 4 Switching Voltage value when width of nmos and pmos are 120nm

$$Vth = \frac{Vdd - |Vtp| + Vtn\sqrt{\frac{Kn}{Kp}}}{1 + \sqrt{\frac{Kn}{Kp}}}$$

The switching voltage formula for a threshold-independent quantizer (TIQ) comparator can be expressed as the difference between the input signal voltage and the reference voltage divided by the gain of the comparator. This formula defines the threshold at which the comparator transitions between its two output states, typically high and low. The comparator's decision boundary is determined by this switching voltage, allowing it to make binary decisions based on whether the input voltage is above or below this threshold.

3.1.2 One small example with 2-bit Flash ADC

A 2-bit Flash Analog-to-Digital Converter (ADC) employs a basic architecture utilizing comparators to rapidly convert analog input into a digital output. Here's the theory behind a 2-bit Flash ADC:

1. Comparator Network: The 2-bit Flash ADC consists of three comparators in a simple setup, comparing the input voltage against two reference levels.

2. Reference Voltages: The input voltage is compared against two reference voltages, dividing the input range into three segments. If the input voltage exceeds the first reference, the highest bit outputs '1'. If it's between the two references, the middle bit outputs '1'. If the input exceeds the second reference, the lowest bit outputs '1'.

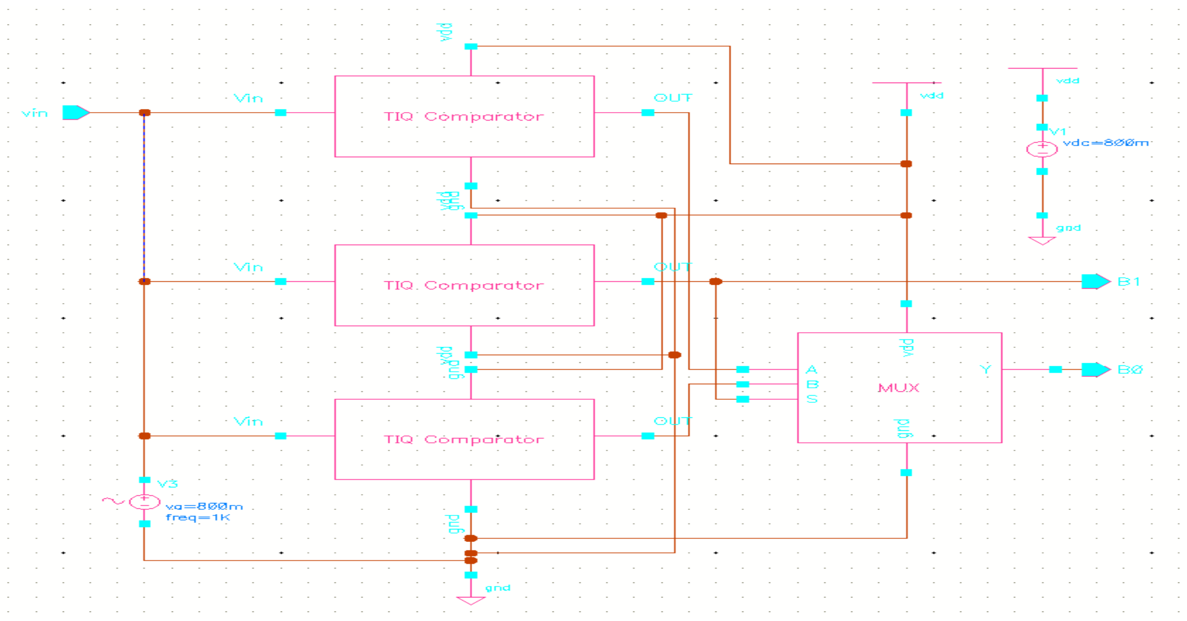


Figure 5 2 Bit flah adc

3. Digital Output: The outputs from these comparators form the 2-bit digital code, representing one of four possible levels within the input range based on where the input voltage falls concerning the reference levels.

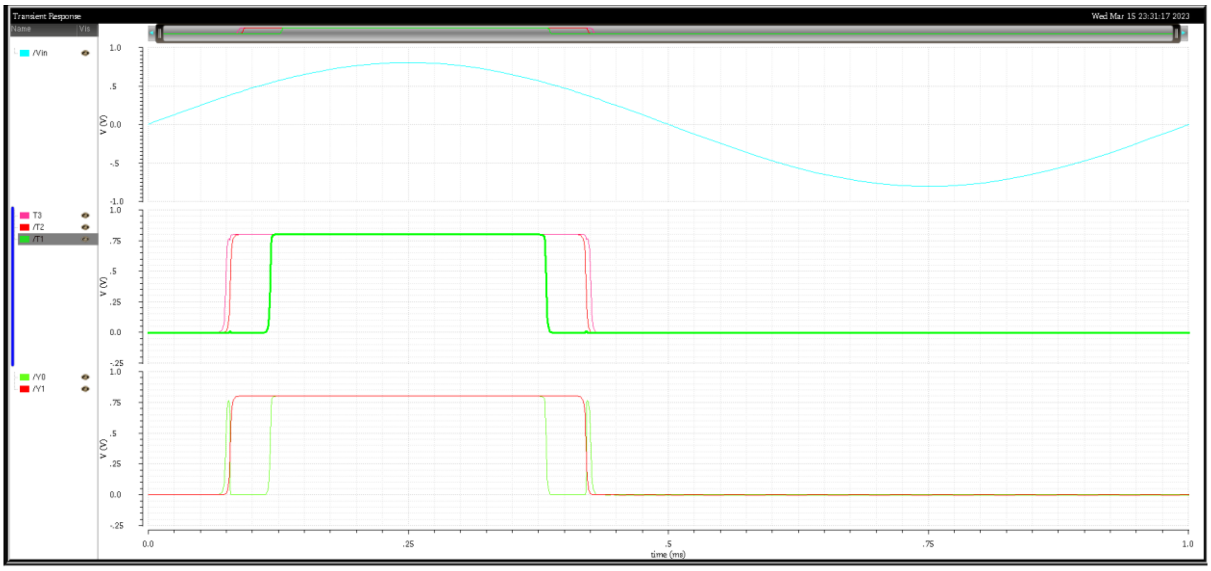


Figure 6 Simulation 2 bit flash adc

3.1.3 Analog design of 8 Bit flash ADC

1. Comparator Array: An 8-bit Flash ADC consists of 255 comparators arranged in a ladder network, comparing the input voltage against 255 reference levels.
2. Reference Voltages: The input voltage is compared simultaneously against multiple reference voltages, dividing the input range into 256 segments (2^8 levels for an 8-bit ADC). Each comparator checks if the input voltage exceeds a specific reference voltage level.
3. Digital Encoding: As the input voltage surpasses each reference level, corresponding comparators output '1.' These outputs create an 8-bit binary digital code that represents the highest reference voltage surpassed by the input signal, effectively encoding the analog signal into a digital format.

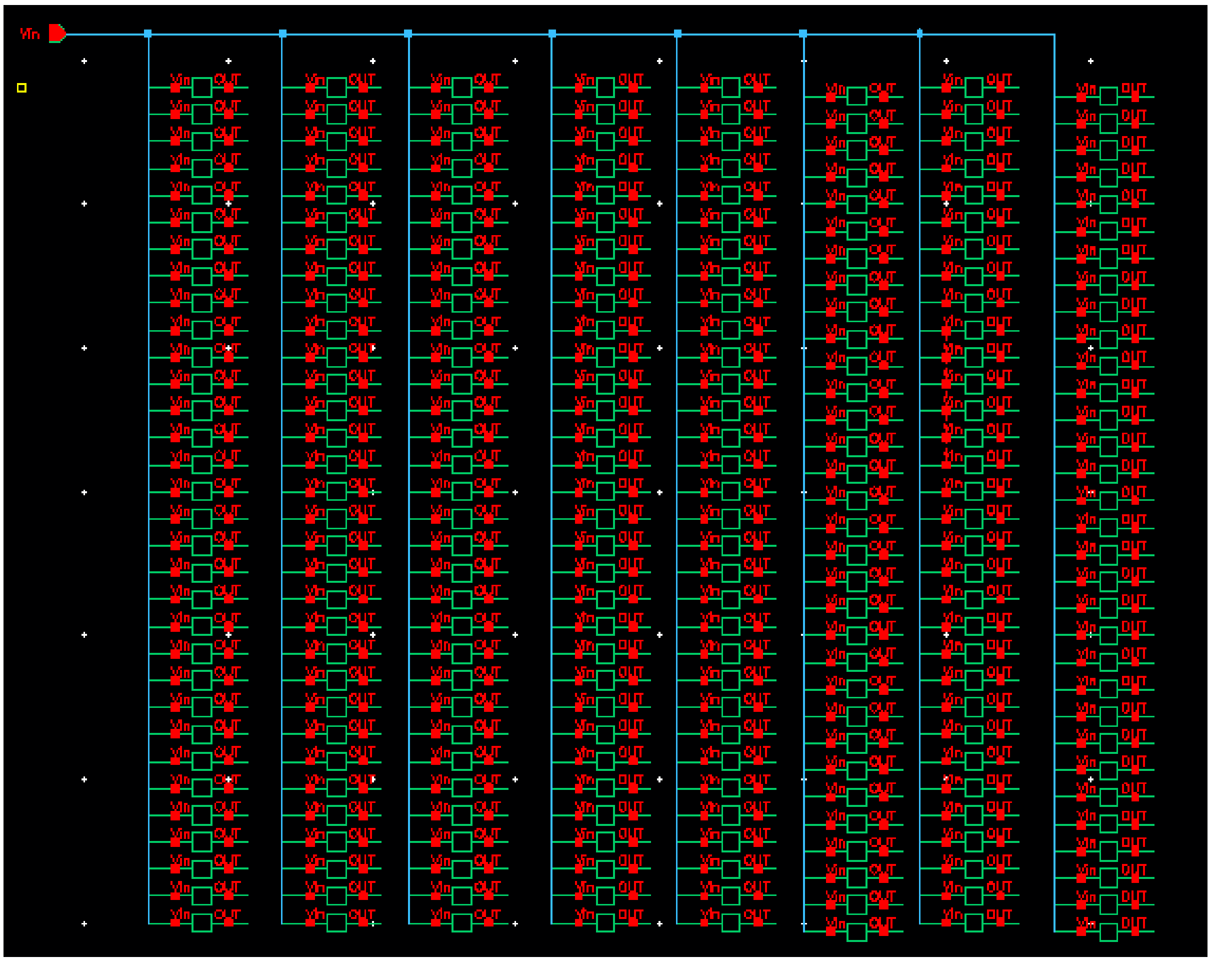


Figure 7 Analog design of the IP

Figure 6 Analog Design of the IP

3.1.4 Layout

1. Physical Representation: Layout theory in Cadence focuses on creating the physical representation of electronic circuits, where components, interconnections, and geometries are precisely designed to fabricate ICs.

2. Design Hierarchy: Cadence tools allow designers to create layouts hierarchically, breaking down complex designs into manageable blocks or cells. Each cell represents a specific circuit or module, facilitating ease of design and reuse.

7. DRC and LVS Checks: Design Rule Checks (DRC) and Layout versus Schematic (LVS) verification are essential steps. DRC ensures that the layout meets the specified design rules, while LVS verifies the consistency between the schematic and the layout.

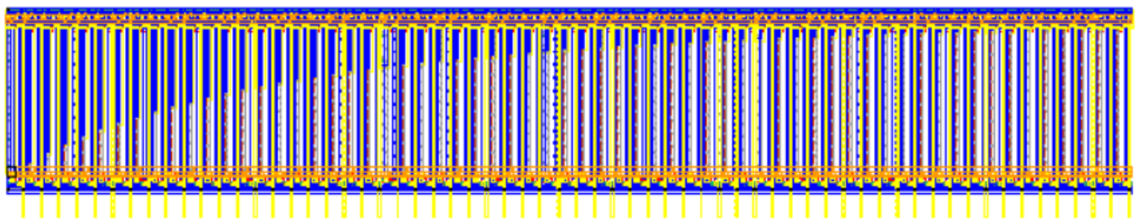


Figure 8 Layout of analog design

Figure 7 Layout of Analog Design of ADC IP

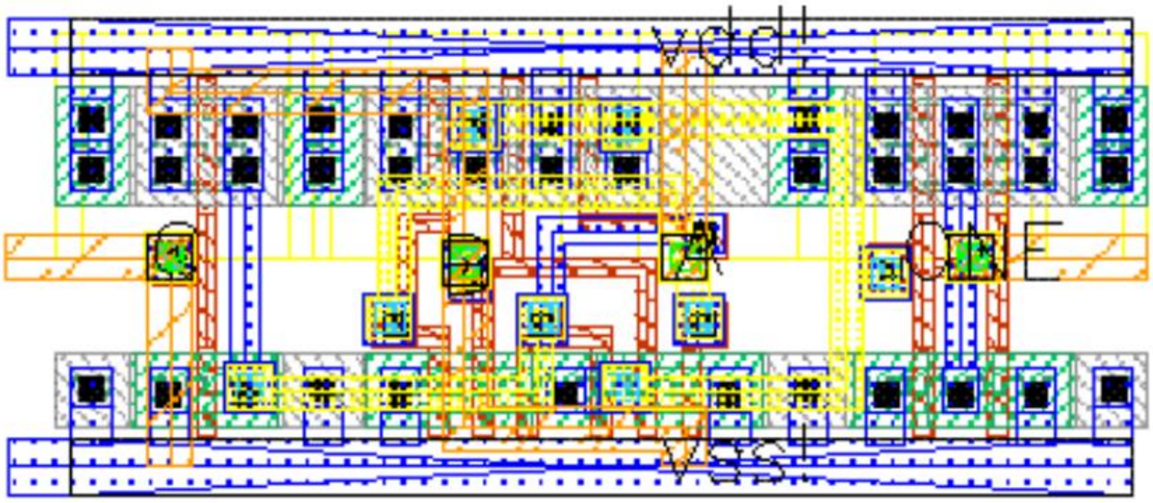


Figure 9 Layout of TIQ comparator

Figure 8 Layout of TIQ comparator

3.1.5 Testbench

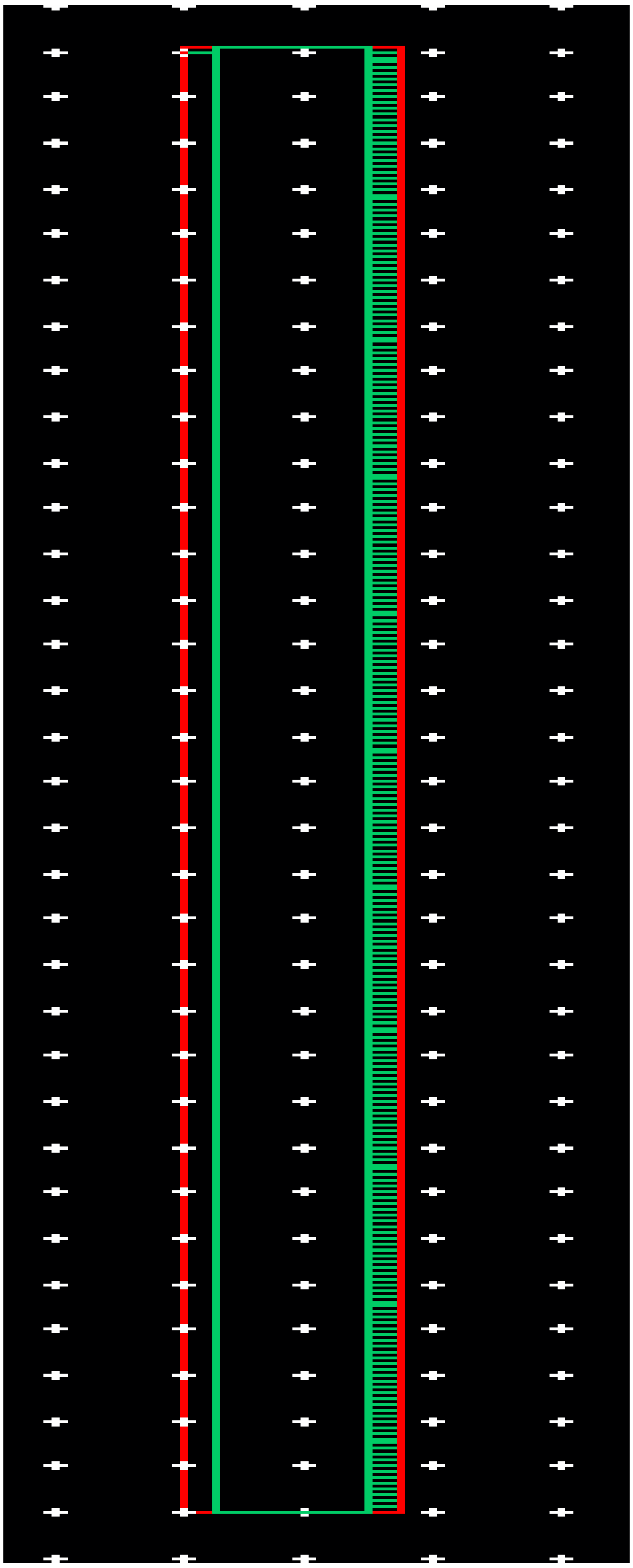


Figure 10 Full testbech of Analog Design

3.1.6 Analog Design Verification

1) Circuit Simulation:

Simulation

Analog verification involves simulating the behavior of analog circuits using tools like SPICE (Simulation Program with Integrated Circuit Emphasis). SPICE simulations help analyze the performance of analog components like transistors, resistors, capacitors, and other devices.

2) Layout vs. Schematic (LVS) Checks:

LVS checks ensure that the physical layout of the circuit matches the intended schematic, preventing errors that can occur during the layout phase.

3.2 Digital Design

The design of the digital back-end for an 8-bit Flash ADC involves the implementation of a priority encoder to efficiently convert the outputs from the TIQ comparators into an 8-bit digital representation of the analog input. This crucial stage employs Verilog/SystemVerilog in conjunction with tools like Cadence Incisive for digital simulation. The priority encoder assigns priority to the most significant bit set in the thermometer code generated by the comparators, enabling the conversion of the parallel comparator outputs into a serial 8-bit digital output. Through careful design and simulation, this phase ensures the accurate translation of analog information into a digital format, contributing significantly to the overall precision and functionality of the ADC.

3.2.1 Logic Behind a 255:8 Priority Encoder:

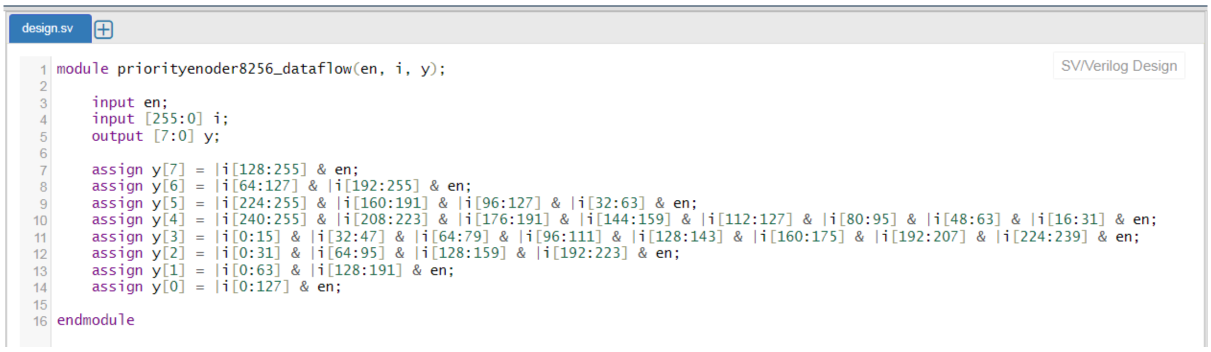
1. Thermometer-to-Binary Encoding: The output of the comparators in the analog front-end generates a thermometer code, where each bit indicates whether the input voltage exceeds a specific threshold. A priority encoder processes this thermometer code into an 8-bit binary representation.

2. Priority Encoding: The priority encoder identifies the highest-order (most significant) bit set in the thermometer code and encodes it into its binary equivalent. For instance, in a 255:8 priority encoder, the highest priority among 255 possible input lines is detected and encoded into an 8-bit binary output.

3. Handling Multiple Inputs: In the context of a Flash ADC, multiple comparators generate outputs representing different voltage thresholds. The priority encoder prioritizes these outputs, ensuring only the most significant bit set in the thermometer code is encoded into the binary output.

4. Output Encoding: The 255:8 priority encoder translates the active input line into an 8-bit binary output, where the position of the active input line is represented in binary form. For example, if the active input line corresponds to the 200th comparator, the output would represent "11001000" in binary (where the most significant bit denotes the 200th line).

3.2.2 Priority encoder Design



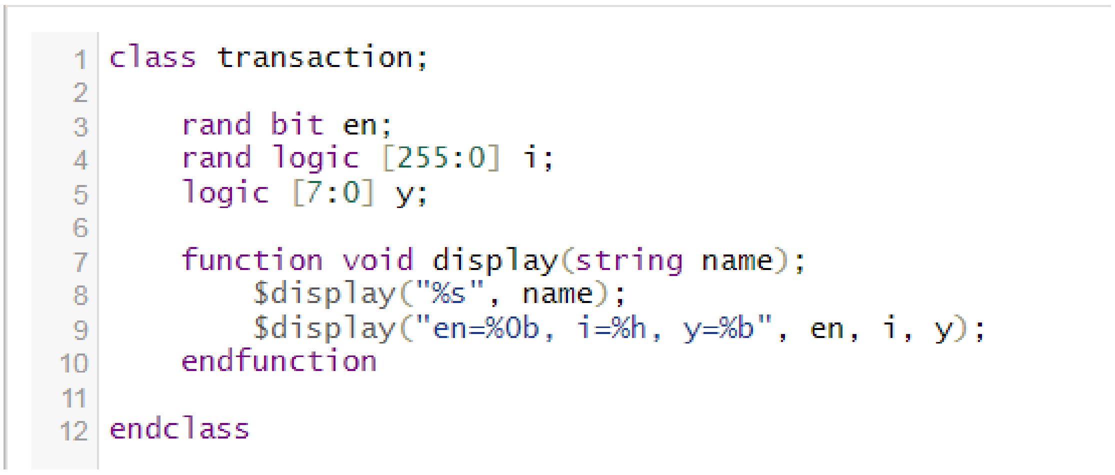
```verilog
module priorityenoder8256_dataflow(en, i, y);

    input en;
    input [255:0] i;
    output [7:0] y;

    assign y[7] = |i[128:255] & en;
    assign y[6] = |i[64:127] & |i[192:255] & en;
    assign y[5] = |i[224:255] & |i[160:191] & |i[96:127] & |i[32:63] & en;
    assign y[4] = |i[240:255] & |i[208:223] & |i[176:191] & |i[144:159] & |i[112:127] & |i[80:95] & |i[48:63] & |i[16:31] & en;
    assign y[3] = |i[0:15] & |i[32:47] & |i[64:79] & |i[96:111] & |i[128:143] & |i[160:175] & |i[192:207] & |i[224:239] & en;
    assign y[2] = |i[0:31] & |i[64:95] & |i[128:159] & |i[192:223] & en;
    assign y[1] = |i[0:63] & |i[128:191] & en;
    assign y[0] = |i[0:127] & en;

endmodule
```

Figure 11 Verilog code of Digital Design

3.2.3 Priority encoder Testbench

a) Transaction class

Defines the pin level activity generated by agent (to drive to DUT through the driver) or the activity has to be observed by agent (Placeholder for the activity monitored by the monitor on DUT signals)

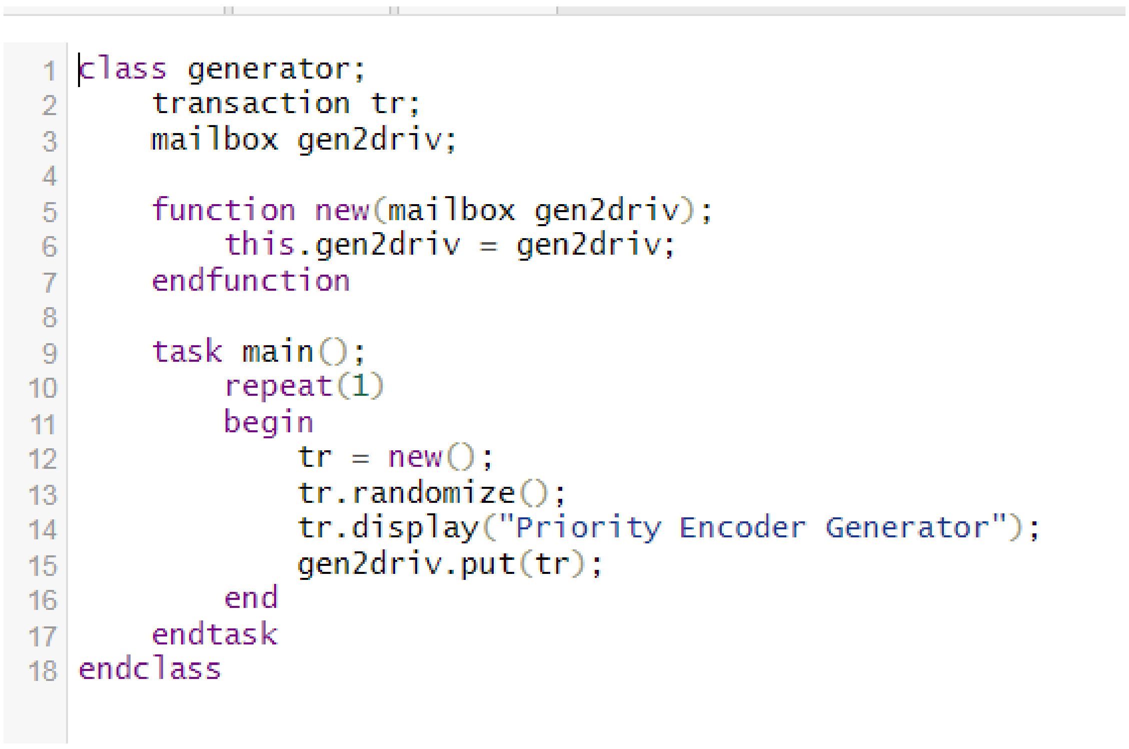


```systemverilog
class transaction;

    rand bit en;
    rand logic [255:0] i;
    logic [7:0] y;

    function void display(string name);
        $display("%s", name);
        $display("en=%0b, i=%h, y=%b", en, i, y);
    endfunction

endclass
```

Figure 12 Transaction class

B) Generator class

Generates the stimulus (create and randomize the transaction class) and send it to Driver

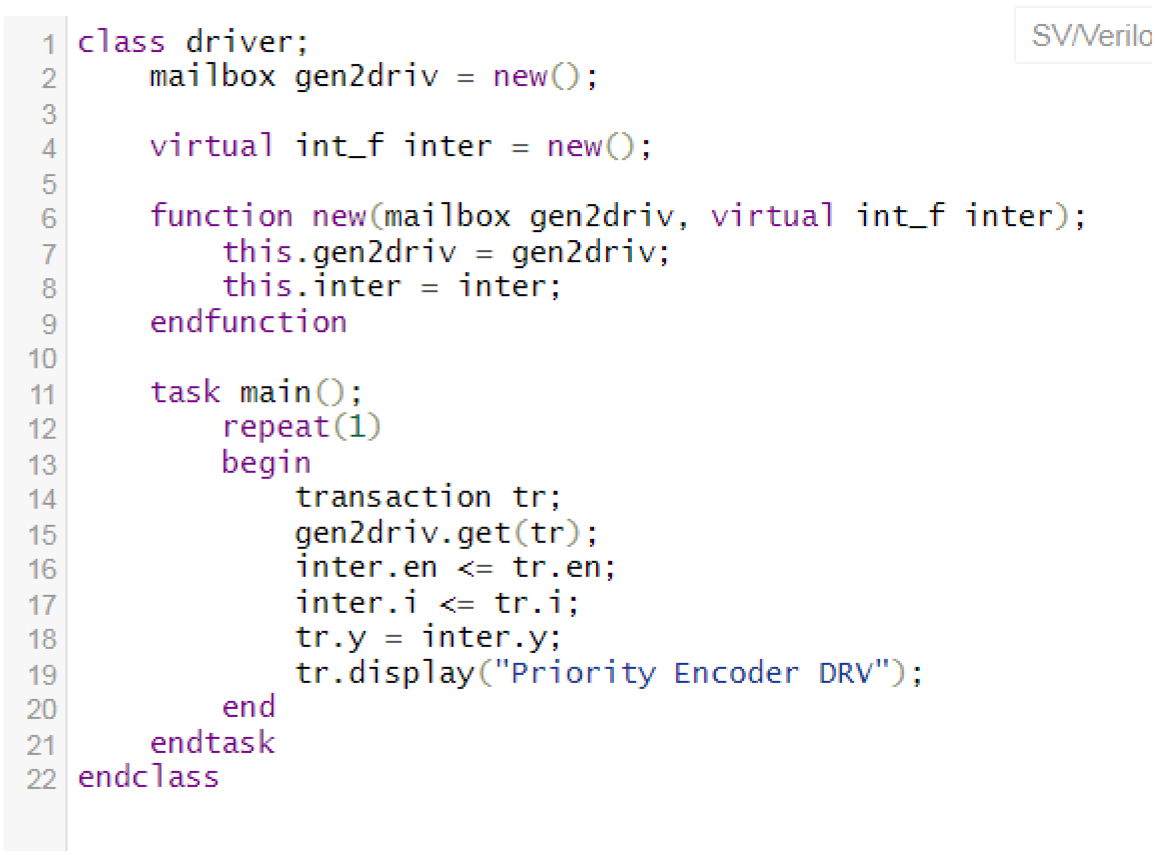


```
class generator;
    transaction tr;
    mailbox gen2driv;

    function new(mailbox gen2driv);
        this.gen2driv = gen2driv;
    endfunction

    task main();
        repeat(1)
        begin
            tr = new();
            tr.randomize();
            tr.display("Priority Encoder Generator");
            gen2driv.put(tr);
        end
    endtask
endclass
```

Figure 13 Generator class

C) Driver class

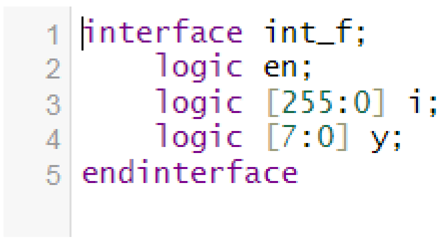
Receives the stimulus (transaction) from a generator and drives the packet level data inside the transaction into pin level.



```
class driver;
    mailbox gen2driv = new();

    virtual int_f inter = new();

    function new(mailbox gen2driv, virtual int_f inter);
        this.gen2driv = gen2driv;
        this.inter = inter;
    endfunction

    task main();
        repeat(1)
        begin
            transaction tr;
            gen2driv.get(tr);
            inter.en <= tr.en;
            inter.i <= tr.i;
            tr.y = inter.y;
            tr.display("Priority Encoder DRV");
        end
    endtask
endclass
```

Figure 14 Driver class

D) Interface class

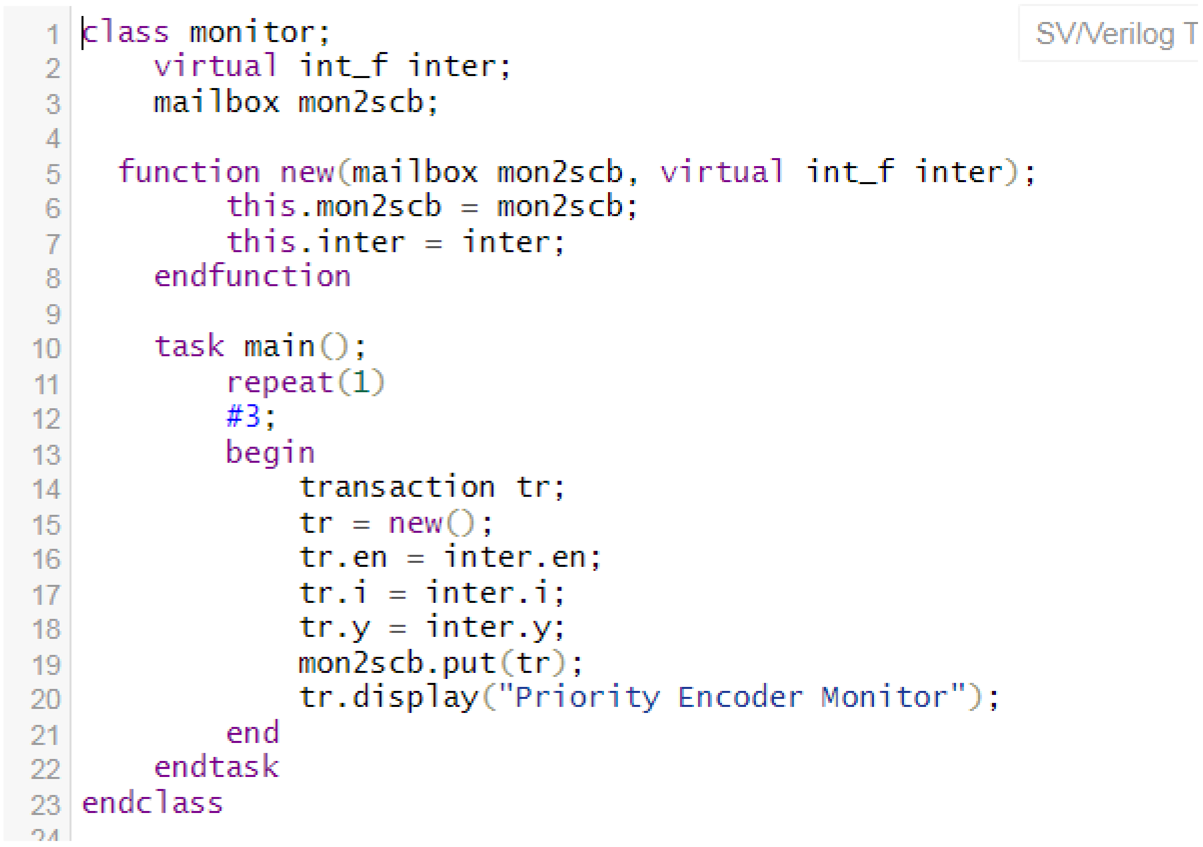


```
interface int_f;
    logic en;
    logic [255:0] i;
    logic [7:0] y;
endinterface
```

Figure 15 Interface class

E) Monitor class

Observes pin level activity on interface signals and converts into packet level which is sent to the components such as scoreboard

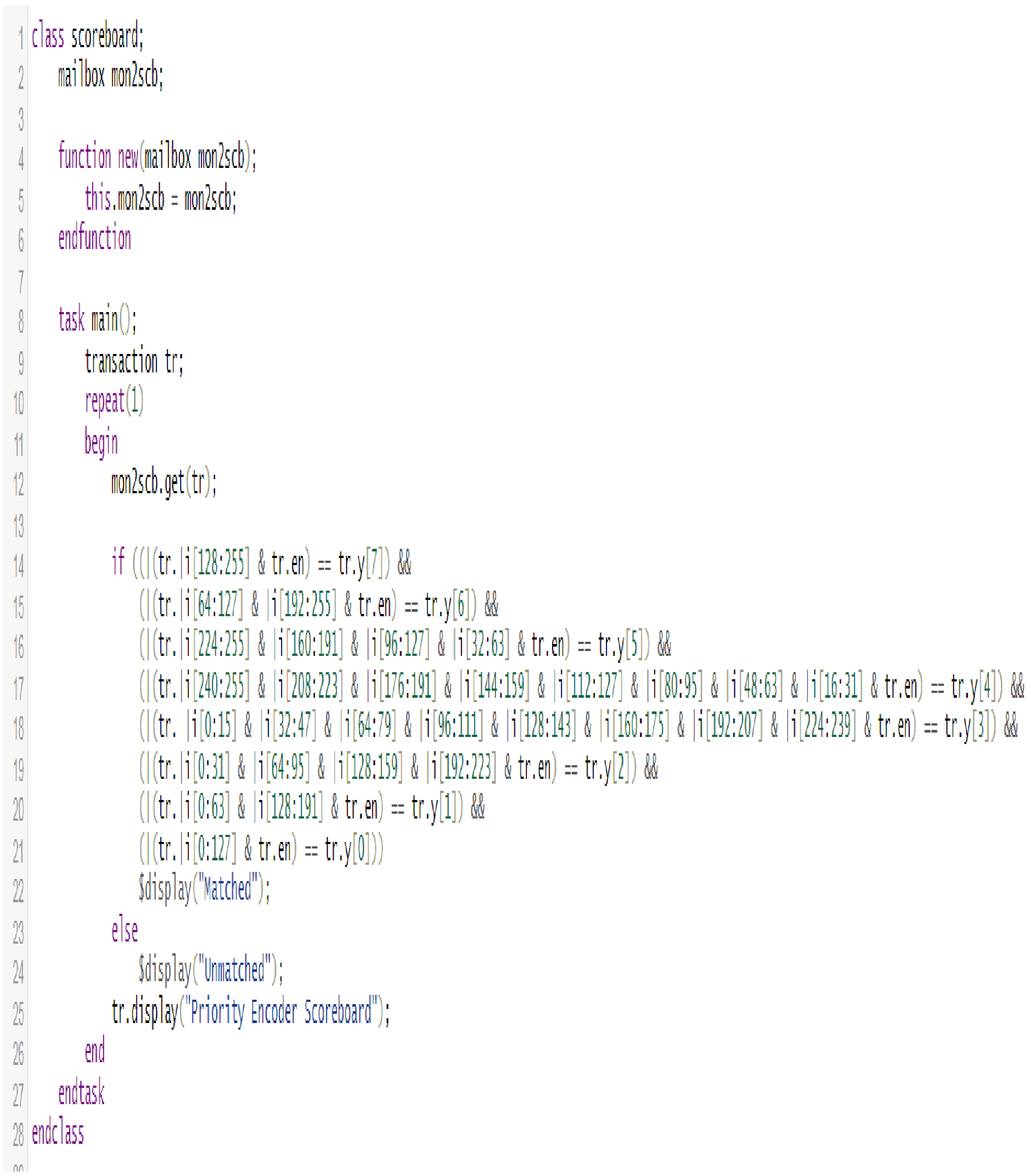


```
class monitor;
    virtual int_f inter;
    mailbox mon2scb;

  function new(mailbox mon2scb, virtual int_f inter);
        this.mon2scb = mon2scb;
        this.inter = inter;
    endfunction

    task main();
        repeat(1)
        #3;
        begin
            transaction tr;
            tr = new();
            tr.en = inter.en;
            tr.i = inter.i;
            tr.y = inter.y;
            mon2scb.put(tr);
            tr.display("Priority Encoder Monitor");
        end
    endtask
endclass
```

Figure 16 Monitor class

F) Scoreboard class

Receives data items from monitors and compares them with expected values. Expected values can be either golden reference values or generated from the reference model

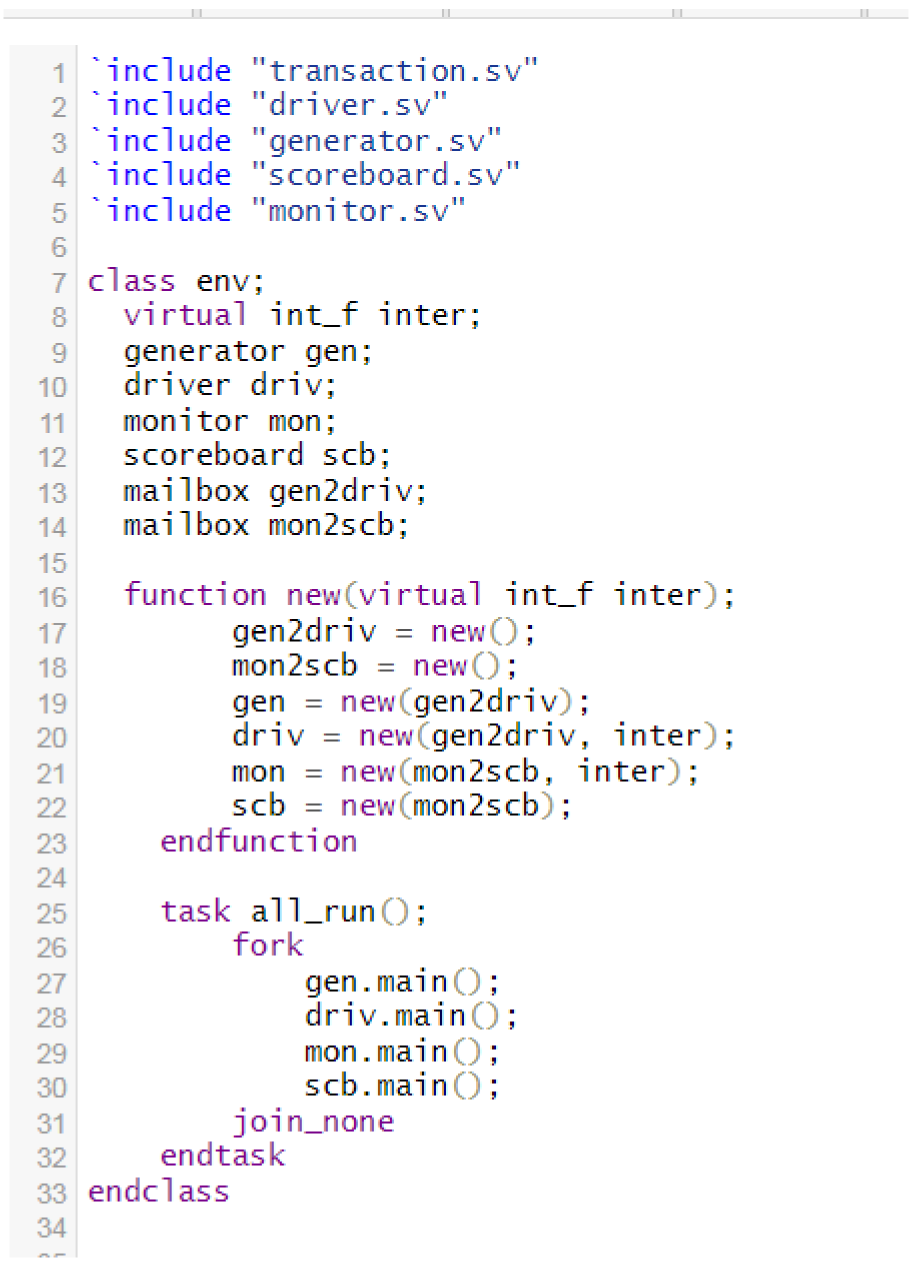


```
class scoreboard;
    mailbox mon2scb;

    function new(mailbox mon2scb);
        this.mon2scb = mon2scb;
    endfunction

    task main();
        transaction tr;
        repeat(1)
        begin
            mon2scb.get(tr);

            if (((|(tr.|i[128:255] & tr.en) == tr.y[7]) &&
                (|(tr.|i[64:127] & |i[192:255] & tr.en) == tr.y[6]) &&
                (|(tr.|i[224:255] & |i[160:191] & |i[96:127] & |i[32:63] & tr.en) == tr.y[5]) &&
                (|(tr.|i[240:255] & |i[208:223] & |i[176:191] & |i[144:159] & |i[112:127] & |i[80:95] & |i[48:63] & |i[16:31] & tr.en) == tr.y[4]) &&
                (|(tr. |i[0:15] & |i[32:47] & |i[64:79] & |i[96:111] & |i[128:143] & |i[160:175] & |i[192:207] & |i[224:239] & tr.en) == tr.y[3]) &&
                (|(tr.|i[0:31] & |i[64:95] & |i[128:159] & |i[192:223] & tr.en) == tr.y[2]) &&
                (|(tr.|i[0:63] & |i[128:191] & tr.en) == tr.y[1]) &&
                (|(tr.|i[0:127] & tr.en) == tr.y[0]))
                $display("Matched");
            else
                $display("Unmatched");
            tr.display("Priority Encoder Scoreboard");
        end
    endtask
endclass
```

Figure 17 Scoreboard class

G) Environment Class

The environment is a container class for grouping higher level components like agent's and scoreboard

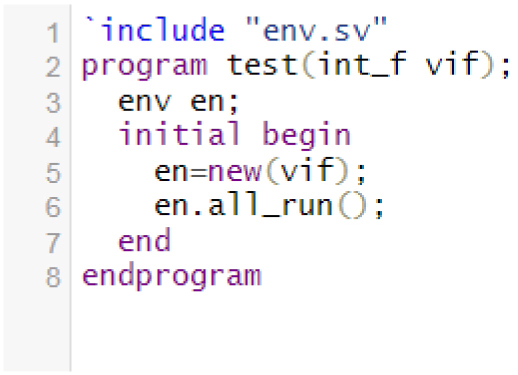


```
`include "transaction.sv"
`include "driver.sv"
`include "generator.sv"
`include "scoreboard.sv"
`include "monitor.sv"

class env;
  virtual int_f inter;
  generator gen;
  driver driv;
  monitor mon;
  scoreboard scb;
  mailbox gen2driv;
  mailbox mon2scb;

  function new(virtual int_f inter);
        gen2driv = new();
        mon2scb = new();
        gen = new(gen2driv);
        driv = new(gen2driv, inter);
        mon = new(mon2scb, inter);
        scb = new(mon2scb);
    endfunction

    task all_run();
        fork
            gen.main();
            driv.main();
            mon.main();
            scb.main();
        join_none
    endtask
endclass
```

Figure 18 Environment class

H) test Class

The test is responsible for, Configuring the testbench Ezoic Initiate the testbench components construction process Initiate the stimulus driving



```
`include "env.sv"
program test(int_f vif);
  env en;
  initial begin
    en=new(vif);
    en.all_run();
  end
endprogram
```

Figure 19 Testbench top class

1) Logic Simulation:

Digital verification primarily involves logic simulation, where the functionality of digital circuits is simulated using hardware description languages (HDLs).

2) Functional Verification:

Functional verification ensures that the digital circuit performs the intended logical functions correctly. Techniques such as constrained random testing, formal verification, and assertion-based verification are commonly used for functional verification.

3) Coverage Analysis:

Coverage analysis is a critical aspect of digital verification to ensure that all parts of the design have been exercised during simulation.

Code coverage, functional coverage, and assertion coverage are metrics used to measure the completeness of the verification process.

Testbench of Digital Design

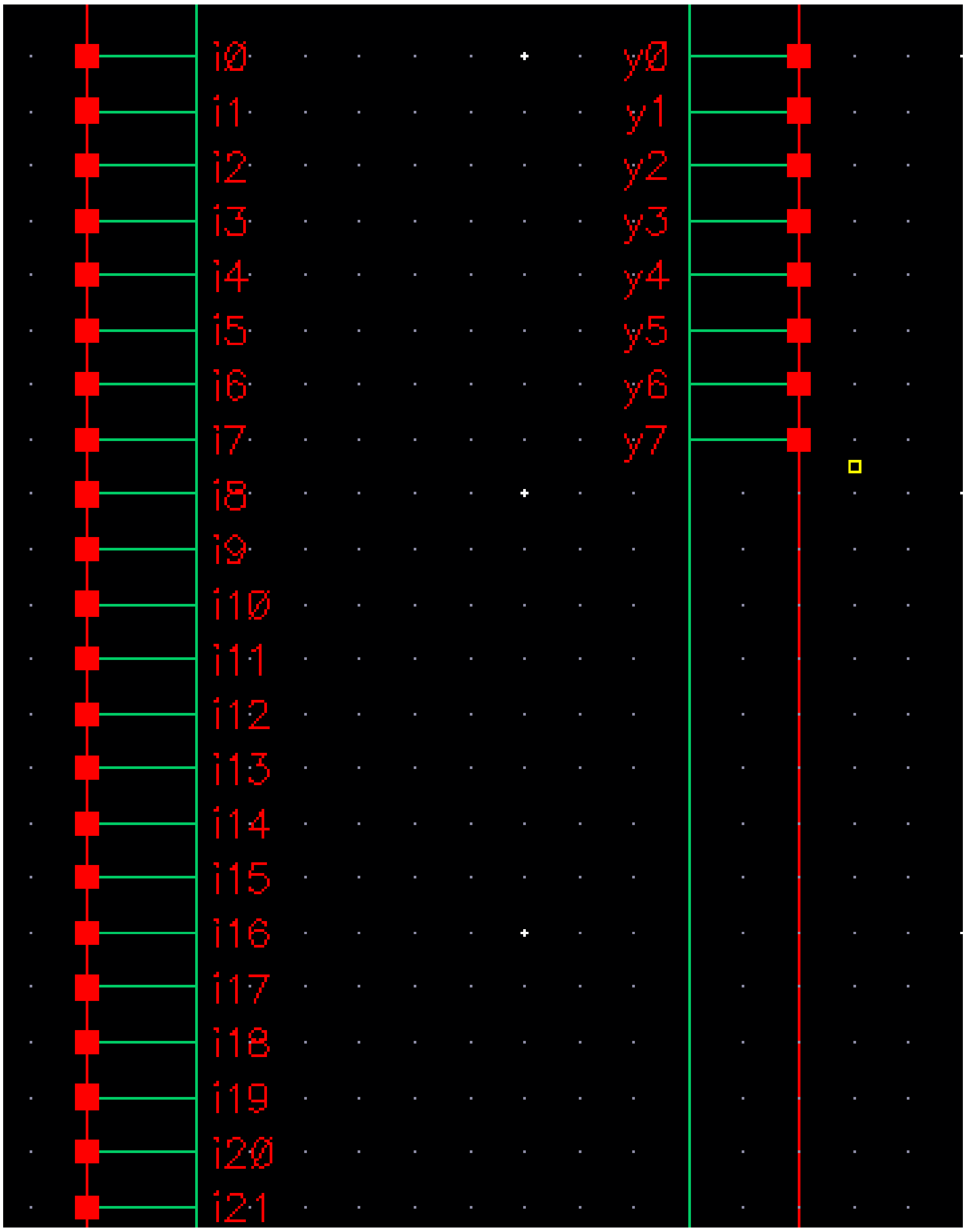


Figure 20 Testbench of Digital Design

3.3 Mixed signal simulation

Integrate the analog and digital components and conduct mixed-signal simulations to verify the ADC's performance across analog and digital domains. Tools like Cadence Spectre

Mixed-signal simulation involves the integration of both analog and digital components within a single simulation environment to analyze the interaction between these domains. Cadence provides tools like Spectre and AMS (Analog Mixed-Signal) that enable mixed-signal simulations. Here's an overview:

3.3.1 Analog and Digital Integration:

1. Analog Components:
  - Analog components like amplifiers, filters, and ADCs are represented using transistor-level models.

2. Digital Components:
  - Digital blocks, such as logic gates, processors, or memory units, are described using digital abstraction models (RTL, Verilog, VHDL).

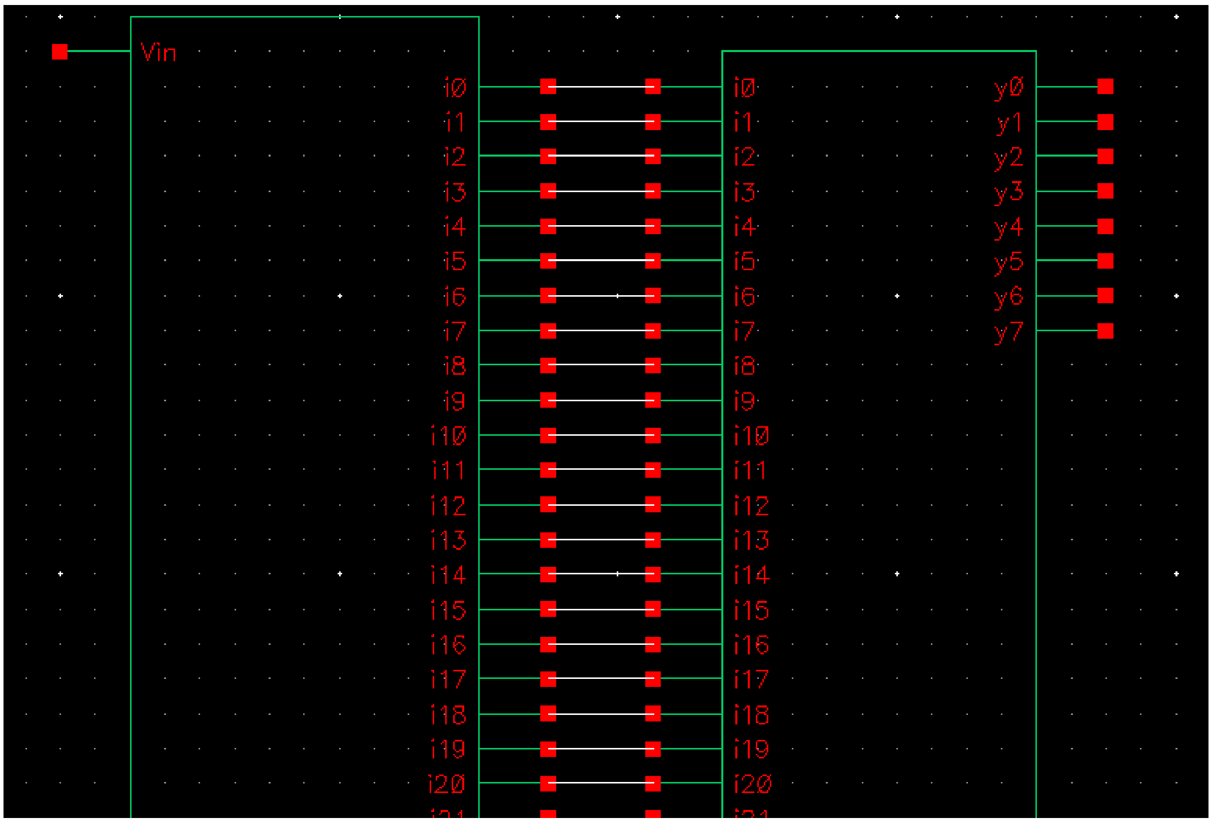


Figure 21 Mixed signal design testbench

3.3.2 Types of Mixed-Signal Simulations:

1. Transient Simulations:

- Simulating the dynamic behavior of the mixed-signal system over time. It captures interactions between analog and digital signals during various operational scenarios.

2. DC and AC Analysis:

- Analyzing the steady-state behavior (DC) and frequency response (AC) of mixed-signal systems.

Chapter 4

Results

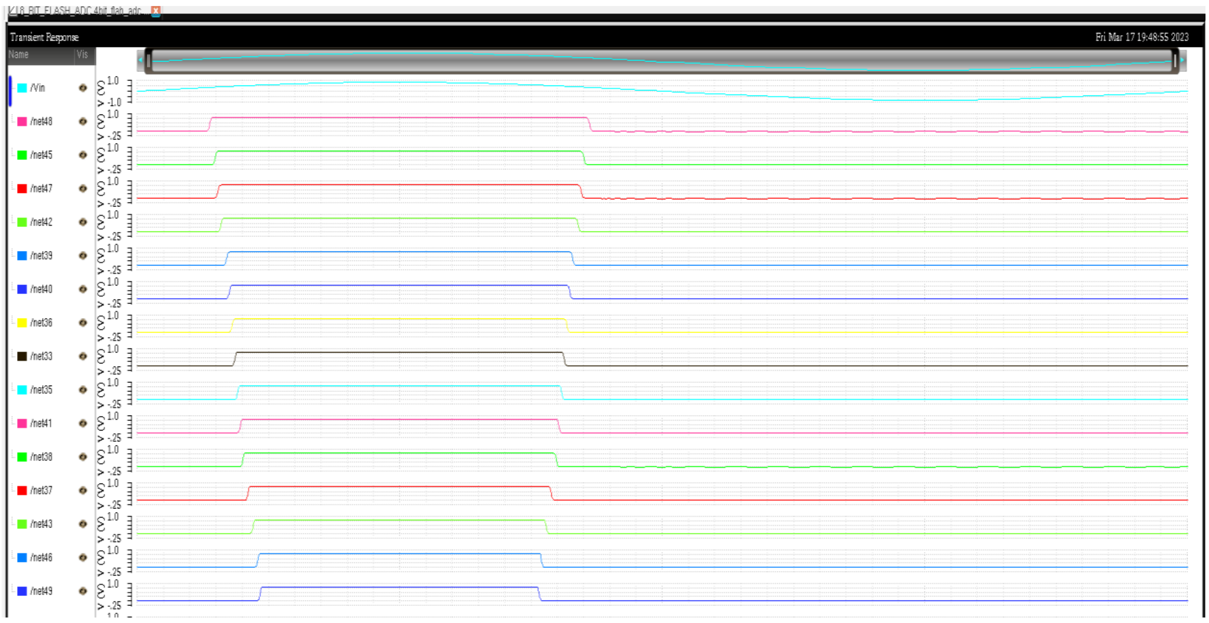
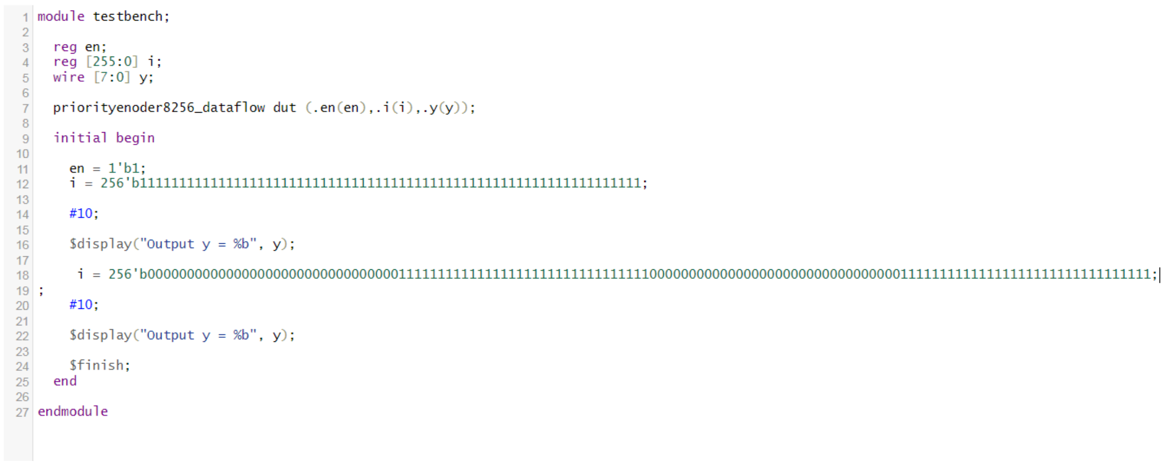


Figure 22 Simulation result of Analog Design

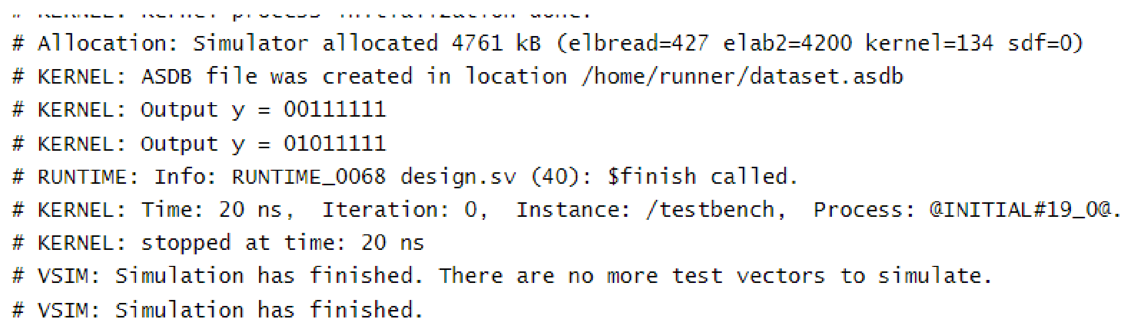
1) Logic Simulation:

Digital verification primarily involves logic simulation, where the functionality of digital circuits is simulated using hardware description languages (HDLs).



```verilog
module testbench;

  reg en;
  reg [255:0] i;
  wire [7:0] y;

  priorityenoder8256_dataflow dut (.en(en),.i(i),.y(y));

  initial begin

    en = 1'b1;
    i = 256'b1111111111111111111111111111111111111111111111111111111111111111;

    #10;

    $display("Output y = %b", y);

     i = 256'b00000000000000000000000000000000111111111111111111111111111111110000000000000000000000000000000011111111111111111111111111111111;
;
    #10;

    $display("Output y = %b", y);

    $finish;
  end

endmodule
```

Figure 23 Logic Simulation

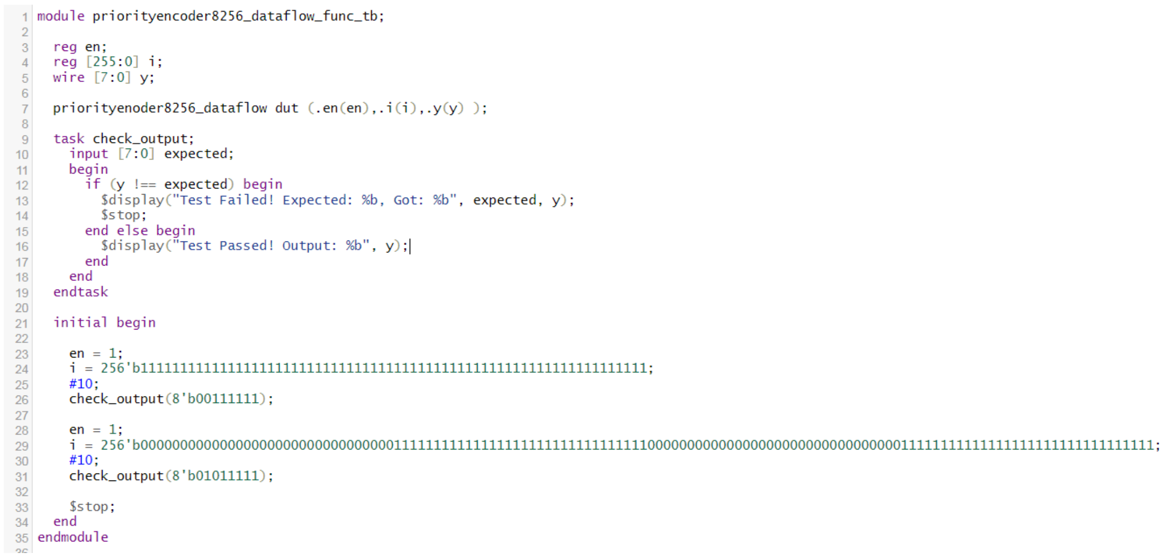


```
# Allocation: Simulator allocated 4761 kB (elbread=427 elab2=4200 kernel=134 sdf=0)
# KERNEL: ASDB file was created in location /home/runner/dataset.asdb
# KERNEL: Output y = 00111111
# KERNEL: Output y = 01011111
# RUNTIME: Info: RUNTIME_0068 design.sv (40): $finish called.
# KERNEL: Time: 20 ns,  Iteration: 0,  Instance: /testbench,  Process: @INITIAL#19_0@.
# KERNEL: stopped at time: 20 ns
# VSIM: Simulation has finished. There are no more test vectors to simulate.
# VSIM: Simulation has finished.
```

Figure 24 Result of logic simulation

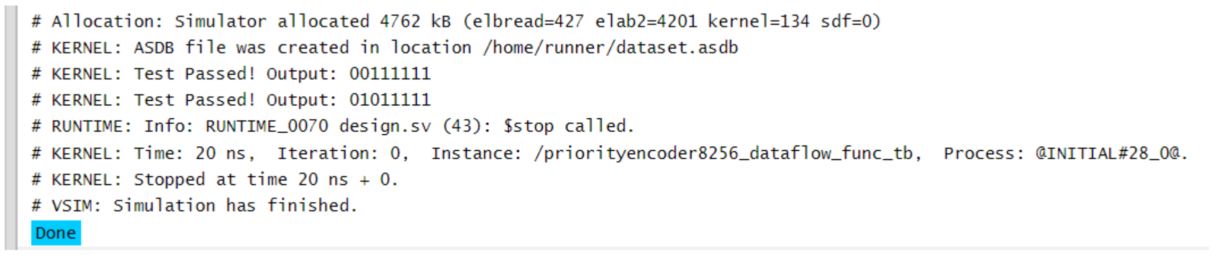
2) Functional Verification:

Functional verification ensures that the digital circuit performs the intended logical functions correctly. Techniques such as constrained random testing, formal verification, and assertion-based verification are commonly used for functional verification.



```verilog
module priorityencoder8256_dataflow_func_tb;

  reg en;
  reg [255:0] i;
  wire [7:0] y;

  priorityenoder8256_dataflow dut (.en(en),.i(i),.y(y) );

  task check_output;
    input [7:0] expected;
    begin
      if (y !== expected) begin
        $display("Test Failed! Expected: %b, Got: %b", expected, y);
        $stop;
      end else begin
        $display("Test Passed! Output: %b", y);
      end
    end
  endtask

  initial begin

    en = 1;
    i = 256'b1111111111111111111111111111111111111111111111111111111111111111;
    #10;
    check_output(8'b00111111);

    en = 1;
    i = 256'b00000000000000000000000000000000111111111111111111111111111111110000000000000000000000000000000011111111111111111111111111111111;
    #10;
    check_output(8'b01011111);

    $stop;
  end
endmodule
```

Figure 25 Function verification



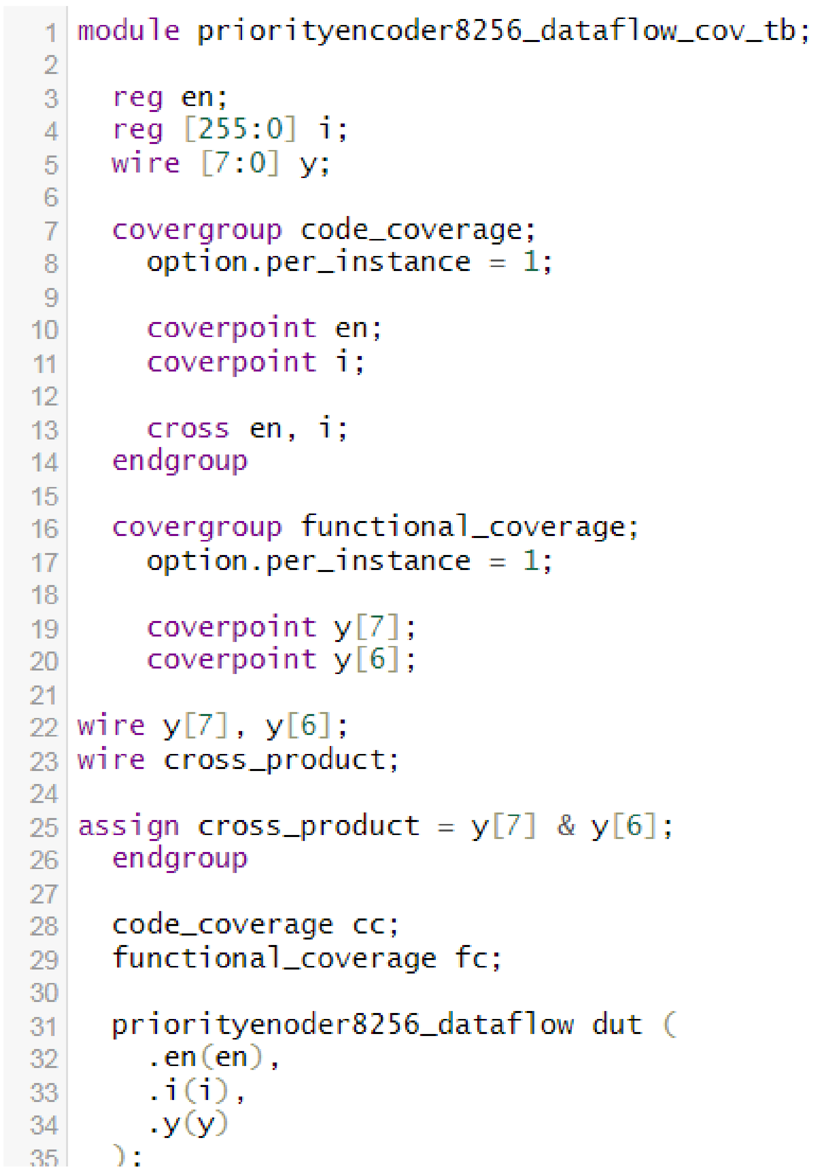
```
# Allocation: Simulator allocated 4762 kB (elbread=427 elab2=4201 kernel=134 sdf=0)
# KERNEL: ASDB file was created in location /home/runner/dataset.asdb
# KERNEL: Test Passed! Output: 00111111
# KERNEL: Test Passed! Output: 01011111
# RUNTIME: Info: RUNTIME_0070 design.sv (43): $stop called.
# KERNEL: Time: 20 ns,  Iteration: 0,  Instance: /priorityencoder8256_dataflow_func_tb,  Process: @INITIAL#28_0@.
# KERNEL: Stopped at time 20 ns + 0.
# VSIM: Simulation has finished.
Done
```

Figure 26 Result of function simulation

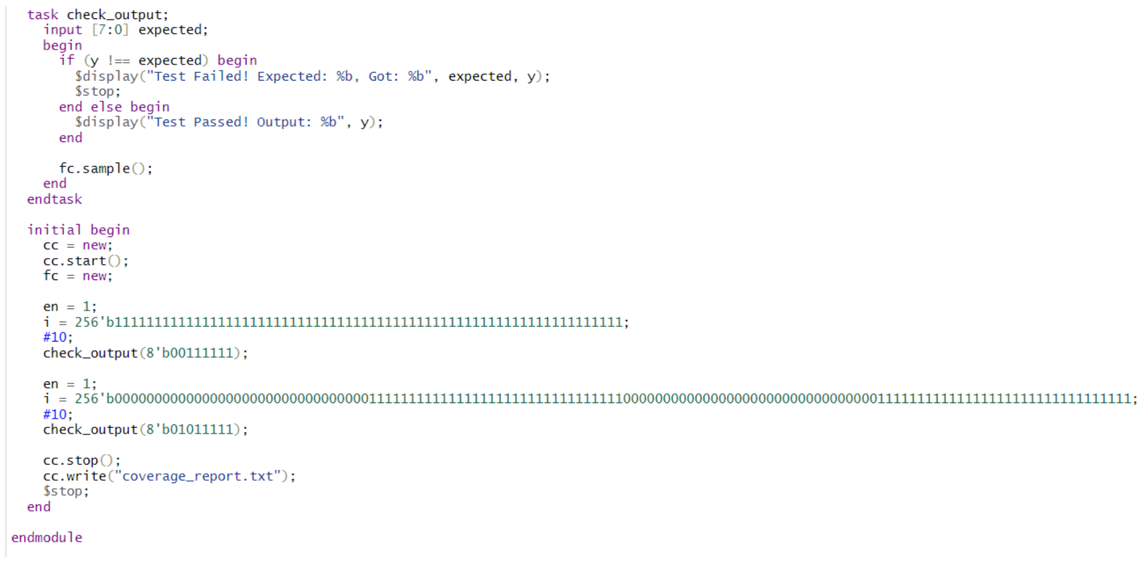
3) Coverage Analysis:

Coverage analysis is a critical aspect of digital verification to ensure that all parts of the design have been exercised during simulation.

Code coverage, functional coverage, and assertion coverage are metrics used to measure the completeness of the verification process.



```
module priorityencoder8256_dataflow_cov_tb;

  reg en;
  reg [255:0] i;
  wire [7:0] y;

  covergroup code_coverage;
    option.per_instance = 1;

    coverpoint en;
    coverpoint i;

    cross en, i;
  endgroup

  covergroup functional_coverage;
    option.per_instance = 1;

    coverpoint y[7];
    coverpoint y[6];

wire y[7], y[6];
wire cross_product;

assign cross_product = y[7] & y[6];
  endgroup

  code_coverage cc;
  functional_coverage fc;

  priorityenoder8256_dataflow dut (
    .en(en),
    .i(i),
    .y(y)
  );
```



```
  task check_output;
    input [7:0] expected;
    begin
      if (y !== expected) begin
        $display("Test Failed! Expected: %b, Got: %b", expected, y);
        $stop;
      end else begin
        $display("Test Passed! Output: %b", y);
      end

      fc.sample();
    end
  endtask

  initial begin
    cc = new;
    cc.start();
    fc = new;

    en = 1;
    i = 256'b1111111111111111111111111111111111111111111111111111111111111111;
    #10;
    check_output(8'b00111111);

    en = 1;
    i = 256'b00000000000000000000000000000000111111111111111111111111111111110000000000000000000000000000000011111111111111111111111111111111;
    #10;
    check_output(8'b01011111);

    cc.stop();
    cc.write("coverage_report.txt");
    $stop;
  end

endmodule
```

Figure 27 Coverage Analysis

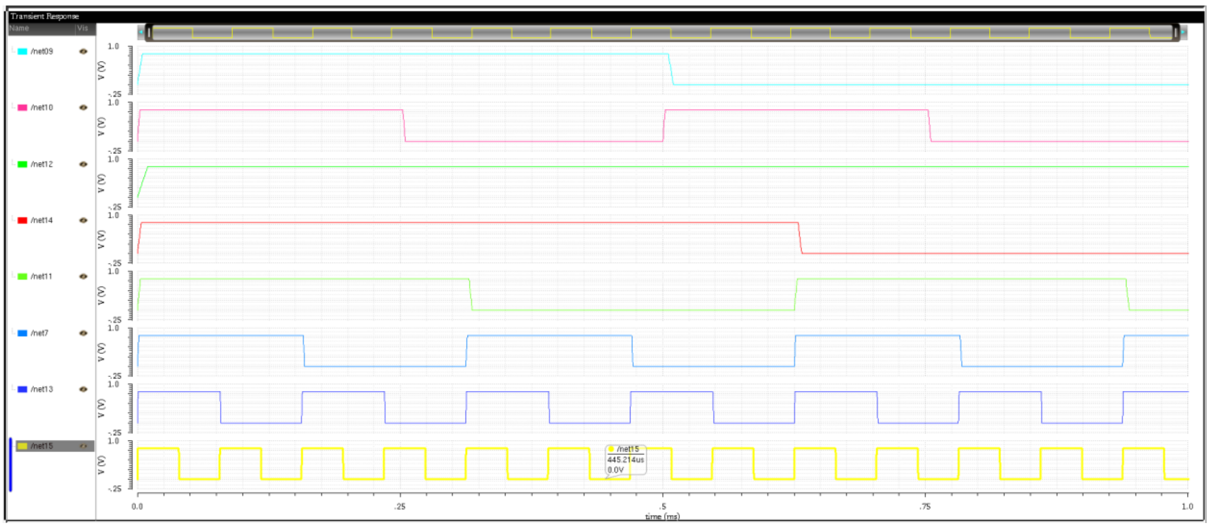


Figure 28 MIxed Simulation result output

Parameter	**FLASH ADC IP**	**DELTA SIGMA IP [1]**
Resolution	8 Bit	8 Bit
Input Range	0.8 Volt.	0.8 Volt
Technology	90nm	90nm
Area	3 mm^2	2.6 mm^2
Speed	2 GS/sec	Not mention
Power Consumption	6.45 microWatt	26.56 microwatt

Comparative analysis of two prominent Analog-to-Digital Converter (ADC) IPs, namely the FLASH ADC and DELTA SIGMA ADC, at an 8-bit resolution within a 0.8-volt input range and utilizing 90nm technology, reveals distinctive trade-offs in performance metrics. While the FLASH ADC occupies a larger silicon area at 3 mm^2, it boasts a lower power consumption of 6.45 microWatt compared to the more compact DELTA SIGMA ADC, which occupies 2.6 mm^2 but consumes 26.56 microWatt. Notably, the DELTA SIGMA ADC's speed remains undisclosed, signifying a potential variable that may impact its overall comparison against the FLASH ADC. These characteristics highlight the nuanced considerations in selecting ADC IPs, emphasizing the interplay between area efficiency, power consumption, and performance in integrated circuit design.

Chapter 5

Conclusion & Future Work

5.1 Conclusion

In conclusion, the development of the ADC IP block encompassed a meticulous process that culminated in a robust and high-performance solution. The specifications were comprehensively met, with a resolution of 8-bit, achieving a speed of 2 GS/sec while maintaining a low power consumption of 6.45 microWatt. The input range was 0.8 Volt. By selecting the appropriate ADC architecture and meticulously designing both the analog front-end and digital back-end components, the functionality and performance were rigorously validated through mixed-signal simulations, ensuring seamless integration and synchronization between analog and digital domains. The thorough documentation process captured the entire design journey, including, providing a comprehensive record of the development process. Continuous iteration and optimization further enhanced the ADC IP block's reliability and performance, making it a dependable solution for real-world applications.

5.2 Reference

[1] D. Lee, J. Yo and Clooi, Design method and automation of comoaarator generation for A/D converters, International Symosin on Quality Electronic Design, Pages 138-142, 2002

[2] K. Uyttenhove and M. Steyaert, 1.8 V 6-bit 1.3 GHz CMOS ADC in 0.45 micrometer CMOS, European Solid State Circuit Conference , Pages 455-458, 2002

[3] Andrea Agnes, Verilog Powered SAR A/D Converter for Instrumentation Application, UniversitA Degli Studi Dipavia, Department of Electronics, Pages 57-59, 2009

[4] Shubhara Yewale, Design of Low Power CMOS Comparator for A/D Converter. Sri JS Institue of Technology and Science Indore India.

[5] Neil H.E Weste, CMOS VLSI Design, Pearson Education Inc, India.

[6] Nikoozadeh and B. Murmann, "An Analysis of Latch Comparator Offset Due to Load Capacitor Mismatch," IEEE Trans. Circuits Syst. II: Exp. Briefs, vol. 53, no. 12, pp. 1398-1402, Dec. 2006

[7] Pedro M.Figueiredo, Joao C.Vital, "Kickback Noise Reduction Techniques for CMOS Latched Comparator", IEEE Transactions on Circuits and Systems, vol.53, no.7, pp.541-545, July 2006.

[8] Ramesh, J., and K. Gunavathi. "A 8-Bit TIQ Based 780 MSPS CMOS Flash A/D Converter." In International Conference on Computational Intelligence and Multimedia Applications (ICCIMA 2007), vol. 2, pp. 201-208. IEEE, 2007.

[9] Tiwari, Suruchi, and Abhishek Kumar. "Reconfigurable Flash ADC using TIQ technique." In 2018 4th International Conference on Computing Sciences (ICCS), pp. 204-208. IEEE, 2018.

[10] Ozdemir, Ali, Mshabab Alrizah, and Kyusun Choi. "Optimization of comparator selection algorithm for TIQ flash ADC using dynamic programming approach." In 2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), pp. 495-500. IEEE, 2019.

[11] Priya, M. Sowmya, M. Senthil Sivakumar, and Sruthi Pulya. "Comparative analysis of the CMOS 180nm technology-based flash ADC designs using dynamic comparator and TIQ comparator." In 2019 2nd International Conference on Power and Embedded Drive Control (ICPEDC), pp. 111-115. IEEE, 2019.

[12] Swarupa, B. N., Dr Vijaya Prakash AM, and K. V. Kumaraswamy. "Implementation of a 3-bit Flash ADC using TIQ Modified Comparator Circuit and NOR-ROM based Encoder." International Journal of Innovative Research in Computer and Communication Engineering 5, no. 5 (2016).

[13] Park, Jun Hyuk, Soobum Kwon, and Kyusun Choi. "Designing Algorithm for the High Speed TIQ ADC, with Improved Accuracy." In 2018 31st IEEE International System-on-Chip Conference (SOCC), pp. 233-237. IEEE, 2018.

[14] Razavi Behzad, "Design of Analog CMOS Integrated Circuits", New York, NY: McGraw-Hill, Inc., 2000

[15] Patel, Nirali Hemant. "Power efficient 4-bit flash ADC using Cadence Virtuoso." Int J Eng Res Technol (IJERT) 10, no. 03 (2021): 2278-0181.

[16] Kalyani, Nayana, and M. Monica. "Design and analysis of high speed and low power 6-bit flash ADC." In 2018 2nd International Conference on Inventive Systems and Control (ICISC), pp. 742-747. IEEE, 2018.

[17] Sireesha, Ranam, and Abhishek Kumar. "Design of low power 0.8 V Flash ADC using TIQ in 90nm technology." In 2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM), pp. 406-410. IEEE, 2015.