**HEALTH MONITORING SYSTEM**

Detailed report about the project for the course VLSI PHYSICAL DESIGN

SUBMITTED BY

# D. Divya Deekshitha - AP22110020103

# P. Sudeep Reddy -AP22110020104

UNDER THE GUIDANCE OF

Dr. SASWAT KUMAR RAM

Assistant Professor, ECE



DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

SRM UNIVERSITY , ANDHRA PRADESH

# CERTIFICATE A logo with text on it Description automatically generated

This is to certify that the following student has successfully completed the project titled

“HEALTH MONITORING SYSTEM”, being submitted for the award of the degree of Bachelor of Technology in the Department of Electronics and Communication Engineering at SRM University AP, as a bonafide work carried out by:

Submitted by

Name of the Student: D. DIVYA DEEKSHITHA

Registration Number: AP22110020103

Name of the Student: P. SUDEEP REDDY

Registration Number: AP22110020104

This project was completed to our satisfaction, and the student has shown keen interest and dedication throughout the project duration. We place on record our appreciation for his sincere efforts.

Under the guidance of

## Dr. Saswat Kumar Ram

Assistant Professor, Department of ECE SRM University – Andhra Pradesh

## Table of Contents :-

|  |  |  |
| --- | --- | --- |
| **S.No** | **Content** | **Pg.No** |
| 1. | Abstract | 4 |
| 2. | Introduction | 5 |
| 3. | Background | 6 |
| 4. | Methodology | 6 |
| 5. | Procedure | 7-11 |
| 6. | Code Implementation | 11-20 |
| 7. | RTL and Functional Simulation | 20 - 24 |
| 8. | Code Coverage ,Linting | 24 - 25 |
| 9. | DFT | 26 - 28 |
| 10. | LEC | 29 - 30 |
| 11. | ATPG | 30 - 32 |
| 12. | PD (Physical design) | 32 - 38 |
| 13. | STA(Static Timing Analysis) | 40 |
| 14. | Conclusion | 40 |

ABSTRACT:-

In the modern era of digital communication and data exchange, security is one of the primary concerns in system design. Among the most widely adopted security mechanisms is password-based user authentication. However, most implementations rely on software-level verification, which can be prone to vulnerabilities such as bruteforce attacks, malware exploitation, and timing-based side-channel attacks. This project aims to address these limitations by implementing a hardware-based secure login system using an ASIC (Application-Specific Integrated Circuit) design methodology, ensuring high speed, improved security, and minimal power consumption.

The proposed system accepts a user-entered password and compares it against a predefined value stored in hardware. If the credentials match, access is granted through a secure enable signal. The design is modelled using Verilog HDL at the Register Transfer Level (RTL) and rigorously verified through testbenches and simulation. The project adopts a full-fledged VLSI design flow using Cadence EDA tools, covering multiple stages:

* RTL Design and Simulation
* Linting and Functional Verification
* Code Coverage Analysis for completeness
* Logic Synthesis using Cadence Genus
* DFT Insertion and ATPG using Cadence Modus
* Logic Equivalence Check (LEC)
* Physical Design (PD) using Cadence Innovus, covering floorplanning, placement, clock tree synthesis, and routing
* Static Timing Analysis (STA) for timing closure and performance validation

The final GDSII file generation confirms that the ASIC meets all design specifications including timing, area, and power constraints. This hardware solution offers a secure, efficient, and robust alternative to conventional software login systems, and demonstrates the integration of VLSI design concepts with real-world applications.

This project not only strengthens the understanding of digital logic design and ASIC implementation but also highlights the importance of hardware-level security in embedded systems and IoT applications.

### INTRODUCTION

1.1 Overview

In today's rapidly evolving world of digital health and wearable technology, the need for compact, real-time, and hardware-efficient health monitoring systems is more important than ever. This project, titled **"HEALTH MONITORING SYSTEM"**, focuses on the implementation of a simple yet effective health monitoring module using Verilog HDL. The design simulates the estimation of several critical health metrics that are commonly used in medical and fitness evaluations.

The module, named health\_monitor, processes biometric input parameters such as weight, height, age, waist circumference, and sex to compute five essential health indicators: Body Mass Index (BMI), Body Fat Percentage (BFP), Relative Fat Mass (RFM), Body Mineral Density (BMD), and Basal Metabolic Rate (BMR). These calculations are carried out synchronously with a clock signal, making the design suitable for integration into real-time FPGA-based systems.

The system applies basic arithmetic and conditional operations to estimate each parameter using simplified formulas derived from standard health assessment models. The design also incorporates safeguards against invalid operations, such as division by zero, and ensures all outputs are clamped to an 8-bit range to match hardware constraints.

This project demonstrates how digital hardware design can be used to build a foundational block for portable and low-power health monitoring solutions, potentially useful in IoT healthcare devices, fitness trackers, and embedded systems.

#### **1.2 Objective**

The main objective of this project is to design, verify, and implement a password-based secure login system using ASIC flow, ensuring:

* Secure and efficient user authentication at the hardware level.
* Complete implementation of ASIC design flow using industry-standard EDA tools from Cadence.
* Functional correctness through simulation and linting.
* Optimization of area, power, and performance via synthesis and physical design.
* Implementation of DFT (Design for Testability) to ensure high fault coverage during testing.

Through this, the project aims to develop an end-to-end secure hardware system that mimics a real-world application of embedded login systems, suitable for secure embedded processors, IoT devices, and consumer electronics.

**Background**

### As health concerns and lifestyle diseases continue to rise globally, accurate and efficient health monitoring systems have become more important than ever. While software-based systems for health tracking are widely available, they often rely on vulnerable computing environments and can be prone to security risks, errors, or inaccuracies. A hardware-based health monitoring system can provide a more reliable and tamper-resistant solution by performing the computations directly in silicon, ensuring both accuracy and robustness.

### ASICs (Application-Specific Integrated Circuits) offer a tailored solution for such tasks, allowing designers to create highly specialized circuits optimized for specific functions such as health parameter computation. These specialized chips provide advantages in terms of speed, power consumption, and area optimization, which are essential for wearable health devices or embedded systems.

### By designing a health monitoring system in an ASIC, key parameters like Body Mass Index (BMI), Basal Metabolic Rate (BMR), Body Fat Percentage (BFP), Relative Fat Mass (RFM), and Bone Mineral Density (BMD) can be computed quickly and accurately without the overhead of software processing, while minimizing power consumption—critical for battery-powered devices.

### Furthermore, the use of VLSI design flows and EDA tools such as those from Cadence ensures that the design process is streamlined, from RTL (Register Transfer Level) coding and simulation to physical implementation and testing. These tools not only verify the functionality of the system but also ensure manufacturability and testability, guaranteeing a reliable and robust ASIC for real-world health monitoring applications.

**Methodology**

The design and implementation of the ASIC for the Health Monitoring System follows a structured and systematic approach based on the standard VLSI design flow using Cadence tools. This methodology ensures that the final chip is optimized for performance, power, area, and reliability, making it suitable for embedded health monitoring applications.

1. Specification Phase: The design process begins with defining the functional requirements of the health monitoring system, which include parameters like Body Mass Index (BMI), Basal Metabolic Rate (BMR), Body Fat Percentage (BFP), and Bone Mineral Density (BMD). The system's inputs, such as sensor data (e.g., heart rate, weight, height, etc.), and outputs like computed health metrics, are specified along with control signals for data management and user interaction.
2. RTL Design: The design is described at the Register Transfer Level (RTL) using Verilog HDL. The core logic for computing health metrics, managing input data, and producing output results is developed. This includes logic for each calculation—BMI, BMR, BFP, RFM, and BMD—and the corresponding control mechanisms to handle sensor data and user inputs.
3. Functional Verification: The RTL design undergoes thorough functional verification through simulation to ensure that the logic behaves correctly under all possible input conditions. A comprehensive testbench is created to verify the correctness of the design. Linting and static checks are applied to ensure code quality by detecting issues like unreachable code, uninitialized variables, and inconsistent signal assignments. Code coverage analysis is conducted to ensure that all logical paths, including edge cases and special conditions, are thoroughly tested.
4. Synthesis: Once the functional correctness is established, the design is synthesized into a gate-level netlist using Cadence Genus. During this phase, the design is optimized for area, power, and timing, ensuring it meets the specified performance and power consumption targets for the health monitoring system. The synthesis process also includes any necessary design transformations, such as logic optimizations and technology mapping.
5. Design for Testability (DFT): The Design for Testability (DFT) phase ensures that the fabricated chip can be thoroughly tested. Automatic Test Pattern Generation (ATPG) using Cadence Modus generates test vectors to detect faults in the design. Additionally, Logic Equivalence Check (LEC) using Cadence Conformal verifies that the synthesized gate-level design is functionally equivalent to the original RTL.
6. Physical Design: The design moves to physical implementation using Cadence Innovus, where key tasks such as floorplanning, standard cell placement, clock tree synthesis (CTS), and routing are performed. The physical design also includes optimizations to reduce power consumption and ensure high performance in a compact area. The final output is a GDSII file, which is ready for fabrication.
7. Static Timing Analysis (STA): Static Timing Analy30sis (STA) is performed throughout the physical design process to ensure that the design meets setup and hold time constraints. STA ensures that all critical paths meet timing requirements, preventing any timing violations that could affect the functionality or reliability of the health monitoring system.

**PROCEDURE :-**

TOOLS USED FOR ASIC FLOW:-

INCISIVE:- Used for functional simulation of designs

GENUS:-Handles synthesis and pre-layout timing analysis of the design.

INNOVUS:-Used for physical design, including placement, routing and optimization

Ensure that the licensing server is active and the client machine is properly connected to the server.

Navigate to the counter directory ,right-click and select “Open in terminal”

To open the necessary cadence tools, type the following command in the terminal :

* Csh
* Source /home/installs/cshrc

1. Creating RTL Code:-

* Open terminal in Linux environment.
* Type gedit and press Enter to launch the text editor.
* Create and save the RTL file as Health\_monitor.v or Health\_monitor\_tb.v.
* Write the Verilog code to implement the password checking logic.
* Again, open gedit to create the testbench file.
* Save it as pc1\_tb.v.
* Write Verilog testbench code to apply inputs and verify outputs.
* Ensure separate files for design and testbench to maintain clarity and ease of simulation.

1. Functional Simulation
   1. Launching Incisive Tool

* Open terminal and run:

nclaunch -new

* Select "Multiple Step", then click "Create cds.lib" to configure the library.
* Save the .lib file and select don’t include any libraries.

* 1. Simulating the Design

Simulation process includes three stages:

* Compile: Verilog/RTL and testbench files are compiled.
* Elaborate: Design and testbench are elaborated and snapshots are generated.
* Simulate: Testbench is run to simulate the top-level module.

* 1. Using nclaunch GUI
* In the nclaunch window:

Select the Verilog files.

Compile them using the Worklib. o Launch the Elaborator.

* After successful elaboration, launch the Simulator from snapshots.

View waveforms, debug logic, and verify functional correctness.

3.Code Coverage Analysis

Command:

ncsim one.v try\_tb.v -access +rwc -coverage all -gui

Points:

* This command enables full coverage (line, toggle, condition) during simulation.
* Use IMC to generate and review the coverage report. 4. Synthesis using Cadence Genus
* Command : genus
* Converts RTL code (.v/.vhdl) into a gate-level netlist.
* Uses Design Constraints (SDC) and library files (.lib) to guide the synthesis.
* Outputs include gate-level netlist, area, timing, and power reports.
* Provides schematic view of synthesized design for verification.
* Verifies the synthesized netlist using a linting report.
* Command for Linting: report\_timing -lint
* Checks design guidelines adherence and catches syntax/logical errors early.
* Ensures better synthesis quality and design reliability.

5. Physical Design – Key Points

* Takes gate-level netlist as input from synthesis.
* Requires additional files: SDC, .lib (library), and LEF (Layer Exchange Format).
* Performed using Cadence Innovus tool.
* Outputs a GDS II file used for chip fabrication.
* Command : Innovus

**Procedure:- 1.** Import design

1. Floor planning
2. Power planning
3. Placement
4. CTS (Clock tree Synthesis)
5. Routing

1) Click on File → Click on Import design → Click on auto assign → Click on 3 dots → Click on two arrows → Select the netlist and add → Close

2) Select the LEF files → Three dots → Click on two arrows → Click on down arrow → home/installs/foundry/digital/90nm/dig/lef → Click on last two files → Add and Close Power Power net: VDD Ground net: VSS

3) Click on create analysis configuration → Click on MMMC → Multiple mode multiple corners → Select library sets → Right click → New → Slow type → ADD → Click on two

arrows → Click on down arrow→ dig/90nm/digital/foundry/installs/home→ lib/slow.lib , Same process of fast.lib

1. Click on QRC technology file → Click on

/home/Installs/foundry/digital/90nm/dig/qrc\_90

1. Select Delay corners → Type max → Select RC corner → Library set: slow → Click on OK
2. Select Delay corners → Type min → Select RC corner → Library set: fast → Click on OK
3. Click on Constraints modes → Type: Constraint → Add → Click on down arrow → Click on Constraints block.tcl → Add → Close → Apply → OK
4. Click on Analysis views → Type: best case → OK ← Delay corner (min)
5. Click on Analysis views → Type: worst case → OK ← Delay corner (max)
6. Click on Setup analysis views → Click on best case 11) Click on Hold analysis views → Click on worst case 12) Save and Close .
7. Click on floorplan, specify floorplan • Basic • Size • Ratio: 1 • Core: 0.6 • Core to left: 6 • Core to top: 6 • Core to right: 6 • Core to bottom: 6
8. Power → Power planning → Connect Global nets Pin name(s): VDD To global net: VDD → Add to list → Remove VDD & type VSS Apply
9. Power → Power planning → Add pin nets → Click on file and select VDD & VSS → OK
10. Top: Metal 9 (a) M • Bottom: Metal 9 (a) M • Left: Metal 8 (a) V • Right: Metal 8 (a) V Click on Offset → (Enter Pro Channel) → Update → OK
11. Power → Power planning → Adding Nets → Click on folder and select VDD & VSS → OK
12. Layer: Metal 9 (9) → Update
13. Power → Power planning → Add Ring → Nets → Click on folder & select VDD & VSS → OK
14. Layer: Metal 8 (8) Verified CTC
15. Clock → Accept clock tree debugging → Select Unit Delay Mode → OK and click on Yes
16. Timing → Debug Timing → Setup → Hold → OK
17. Timing → Open the command window → Minimize it
18. Timing → Extract RC → Save SPEF (.loc) → Click OK Routing
19. Route → Nano Route → Route → Click (on Timing driven) and SI driven → OK
20. Verify → Verify DRC → OK
21. Open the command window and verify the DRC (Take Screenshot)
22. Verify → Connectivity → OK
23. Right side and layers → Remove (All ticks) and take screenshot Chip design & standard cell
24. File → Save Design → Innovus →File Name: Health\_monitor.innovus → Click on OK File → Save → GDS/OASIS →Output file: bit4\_full\_adder.gds → OK → Close and open the gds file.

# CODE IMPLEMENTATION:-

# DESIGN CODE:-

`timescale 1ns / 1ps

module health\_monitor (

input wire clk,

input wire [7:0] w,

input wire [7:0] h,

input wire [7:0] age,

input wire [7:0] waist,

input wire a,

output reg [7:0] bmi,

output reg [7:0] bfp,

output reg [7:0] rfm,

output reg [7:0] bmd,

output reg [7:0] bmr

);

reg [15:0] bmi\_temp;

reg [15:0] bfp\_temp;

reg [15:0] rfm\_temp;

reg [15:0] height\_waist\_ratio;

reg [15:0] bmr\_temp;

always @(posedge clk) begin

if (h != 0) begin

bmi\_temp = (w \* 10000) / (h \* h);

bmi = (bmi\_temp > 255) ? 255 : bmi\_temp[7:0];

end else begin

bmi = 0;

end

bfp\_temp = (120 \* bmi + 23 \* age - 1080 \* a - 540) / 100;

bfp = (bfp\_temp > 255) ? 255 : (bfp\_temp < 0) ? 0 : bfp\_temp[7:0];

if (waist != 0) begin

height\_waist\_ratio = (h \* 20) / waist;

if (a == 1) begin

rfm\_temp = 64 - height\_waist\_ratio;

end else begin

rfm\_temp = 76 - height\_waist\_ratio;

end

rfm = (rfm\_temp > 255) ? 255 : (rfm\_temp < 0) ? 0 : rfm\_temp[7:0];

end else begin

rfm = 0;

end

bmd = (w \* 3) / 10;

bmr\_temp = (10 \* w + 625 \* h / 100 - 5 \* age + (a ? 5 : -161)) / 10;

bmr = (bmr\_temp > 255) ? 255 : (bmr\_temp < 0) ? 0 : bmr\_temp[7:0];

end

endmodule

# TEST BENCH CODE :-

`timescale 1ns / 1ps

module health\_monitor\_tb;

// Inputs

reg [7:0] w;

reg [7:0] h;

reg [7:0] age;

reg [7:0] waist;

reg a;

reg clk; // Added clock

// Outputs

wire [7:0] bmi;

wire [7:0] bfp;

wire [7:0] rfm;

wire [7:0] bmd;

wire [7:0] bmr;

integer i; // Declare integer at module level

// Instantiate the Unit Under Test (UUT)

health\_monitor uut (

.clk(clk),

.w(w),

.h(h),

.age(age),

.waist(waist),

.a(a),

.bmi(bmi),

.bfp(bfp),

.rfm(rfm),

.bmd(bmd),

.bmr(bmr)

);

initial begin

clk = 0;

forever #5 clk = ~clk; // 10ns clock period

end

// Test procedure

initial begin

// Initialize Inputs

w = 0; h = 0; age = 0; waist = 0; a = 0;

#20;

// Test case 1: Zero inputs

w = 0; h = 0; age = 0; waist = 0; a = 0;

@(posedge clk);

#10;

$display("Test 1: w=%d, h=%d, age=%d, waist=%d, a=%b -> bmi=%d, bfp=%d, rfm=%d, bmd=%d, bmr=%d",

w, h, age, waist, a, bmi, bfp, rfm, bmd, bmr);

// Test case 2: Typical female

w = 60; h = 160; age = 30; waist = 80; a = 0;

@(posedge clk);

#10;

$display("Test 2: w=%d, h=%d, age=%d, waist=%d, a=%b -> bmi=%d, bfp=%d, rfm=%d, bmd=%d, bmr=%d",

w, h, age, waist, a, bmi, bfp, rfm, bmd, bmr);

// Test case 3: Typical male

w = 80; h = 175; age = 40; waist = 90; a = 1;

@(posedge clk);

#10;

$display("Test 3: w=%d, h=%d, age=%d, waist=%d, a=%b -> bmi=%d, bfp=%d, rfm=%d, bmd=%d, bmr=%d",

w, h, age, waist, a, bmi, bfp, rfm, bmd, bmr);

// Test case 4: Max inputs

w = 255; h = 255; age = 255; waist = 255; a = 1;

@(posedge clk);

#10;

$display("Test 4: w=%d, h=%d, age=%d, waist=%d, a=%b -> bmi=%d, bfp=%d, rfm=%d, bmd=%d, bmr=%d",

w, h, age, waist, a, bmi, bfp, rfm, bmd, bmr);

// Test case 5: Min non-zero inputs

w = 1; h = 1; age = 1; waist = 1; a = 0;

@(posedge clk);

#10;

$display("Test 5: w=%d, h=%d, age=%d, waist=%d, a=%b -> bmi=%d, bfp=%d, rfm=%d, bmd=%d, bmr=%d",

w, h, age, waist, a, bmi, bfp, rfm, bmd, bmr);

// Test case 6: h=0 edge case

w = 70; h = 0; age = 25; waist = 85; a = 0;

@(posedge clk);

#10;

$display("Test 6: w=%d, h=%d, age=%d, waist=%d, a=%b -> bmi=%d, bfp=%d, rfm=%d, bmd=%d, bmr=%d",

w, h, age, waist, a, bmi, bfp, rfm, bmd, bmr);

// Test case 7: waist=0 edge case

w = 70; h = 170; age = 25; waist = 0; a = 1;

@(posedge clk);

#10;

$display("Test 7: w=%d, h=%d, age=%d, waist=%d, a=%b -> bmi=%d, bfp=%d, rfm=%d, bmd=%d, bmr=%d",

w, h, age, waist, a, bmi, bfp, rfm, bmd, bmr);

// Test case 8: Negative BMR

w = 10; h = 100; age = 200; waist = 50; a = 0;

@(posedge clk);

#10;

$display("Test 8: w=%d, h=%d, age=%d, waist=%d, a=%b -> bmi=%d, bfp=%d, rfm=%d, bmd=%d, bmr=%d",

w, h, age, waist, a, bmi, bfp, rfm, bmd, bmr);

// Test case 9: BMI overflow

w = 100; h = 50; age = 30; waist = 80; a = 0;

@(posedge clk);

#10;

$display("Test 9: w=%d, h=%d, age=%d, waist=%d, a=%b -> bmi=%d, bfp=%d, rfm=%d, bmd=%d, bmr=%d",

w, h, age, waist, a, bmi, bfp, rfm, bmd, bmr);

// Test case 10: BFP negative

w = 40; h = 180; age = 20; waist = 70; a = 1;

@(posedge clk);

#10;

$display("Test 10: w=%d, h=%d, age=%d, waist=%d, a=%b -> bmi=%d, bfp=%d, rfm=%d, bmd=%d, bmr=%d",

w, h, age, waist, a, bmi, bfp, rfm, bmd, bmr);

// Test case 11: RFM overflow (male)

w = 80; h = 200; age = 30; waist = 10; a = 1;

@(posedge clk);

#10;

$display("Test 11: w=%d, h=%d, age=%d, waist=%d, a=%b -> bmi=%d, bfp=%d, rfm=%d, bmd=%d, bmr=%d",

w, h, age, waist, a, bmi, bfp, rfm, bmd, bmr);

// Test case 12: RFM negative (female)

w = 60; h = 150; age = 25; waist = 30; a = 0;

@(posedge clk);

#10;

$display("Test 12: w=%d, h=%d, age=%d, waist=%d, a=%b -> bmi=%d, bfp=%d, rfm=%d, bmd=%d, bmr=%d",

w, h, age, waist, a, bmi, bfp, rfm, bmd, bmr);

// Test case 13: BMD max

w = 255; h = 170; age = 30; waist = 80; a = 0;

@(posedge clk);

#10;

$display("Test 13: w=%d, h=%d, age=%d, waist=%d, a=%b -> bmi=%d, bfp=%d, rfm=%d, bmd=%d, bmr=%d",

w, h, age, waist, a, bmi, bfp, rfm, bmd, bmr);

// Test case 14: BMR overflow

w = 200; h = 200; age = 10; waist = 80; a = 1;

@(posedge clk);

#10;

$display("Test 14: w=%d, h=%d, age=%d, waist=%d, a=%b -> bmi=%d, bfp=%d, rfm=%d, bmd=%d, bmr=%d",

w, h, age, waist, a, bmi, bfp, rfm, bmd, bmr);

// Test case 15: Toggle all bits for each output

for (i = 0; i < 8; i = i + 1) begin

w = (1 << i) \* 10; h = 100; age = 20; waist = 50; a = 0;

@(posedge clk);

#10;

$display("Test 15-%d: w=%d, h=%d, age=%d, waist=%d, a=%b -> bmi=%d, bfp=%d, rfm=%d, bmd=%d, bmr=%d",

i, w, h, age, waist, a, bmi, bfp, rfm, bmd, bmr);

end

w = 255; h = 80; age = 20; waist = 50; a = 0; // Max BMI

@(posedge clk);

#10;

$display("Test 15-8: w=%d, h=%d, age=%d, waist=%d, a=%b -> bmi=%d, bfp=%d, rfm=%d, bmd=%d, bmr=%d",

w, h, age, waist, a, bmi, bfp, rfm, bmd, bmr);

// Test case 16: Toggle BFP, RFM, BMD, BMR bits

w = 40; h = 160; age = 1; waist = 80; a = 1; // BFP near 0

@(posedge clk);

#10;

$display("Test 16a: w=%d, h=%d, age=%d, waist=%d, a=%b -> bmi=%d, bfp=%d, rfm=%d, bmd=%d, bmr=%d",

w, h, age, waist, a, bmi, bfp, rfm, bmd, bmr);

w = 128; h = 128; age = 64; waist = 64; a = 0; // BFP, BMR mid-range

@(posedge clk);

#10;

$display("Test 16b: w=%d, h=%d, age=%d, waist=%d, a=%b -> bmi=%d, bfp=%d, rfm=%d, bmd=%d, bmr=%d",

w, h, age, waist, a, bmi, bfp, rfm, bmd, bmr);

w = 200; h = 100; age = 1; waist = 50; a = 1; // BFP, BMR near 255

@(posedge clk);

#10;

$display("Test 16c: w=%d, h=%d, age=%d, waist=%d, a=%b -> bmi=%d, bfp=%d, rfm=%d, bmd=%d, bmr=%d",

w, h, age, waist, a, bmi, bfp, rfm, bmd, bmr);

// Test case 17: RFM specific toggle

w = 70; h = 140; age = 30; waist = 70; a = 0; // RFM near 0

@(posedge clk);

#10;

$display("Test 17a: w=%d, h=%d, age=%d, waist=%d, a=%b -> bmi=%d, bfp=%d, rfm=%d, bmd=%d, bmr=%d",

w, h, age, waist, a, bmi, bfp, rfm, bmd, bmr);

w = 70; h = 140; age = 30; waist = 70; a = 1; // RFM near 0 (male)

@(posedge clk);

#10;

$display("Test 17b: w=%d, h=%d, age=%d, waist=%d, a=%b -> bmi=%d, bfp=%d, rfm=%d, bmd=%d, bmr=%d",

w, h, age, waist, a, bmi, bfp, rfm, bmd, bmr);

// Finish simulation

#10;

$finish;

end

// Dump coverage data

initial begin

$dumpfile("health\_monitor.vcd");

$dumpvars(0, health\_monitor\_tb);

// Enable coverage collection (for Cadence IES)

// $set\_coverage("all");

end

endmodule

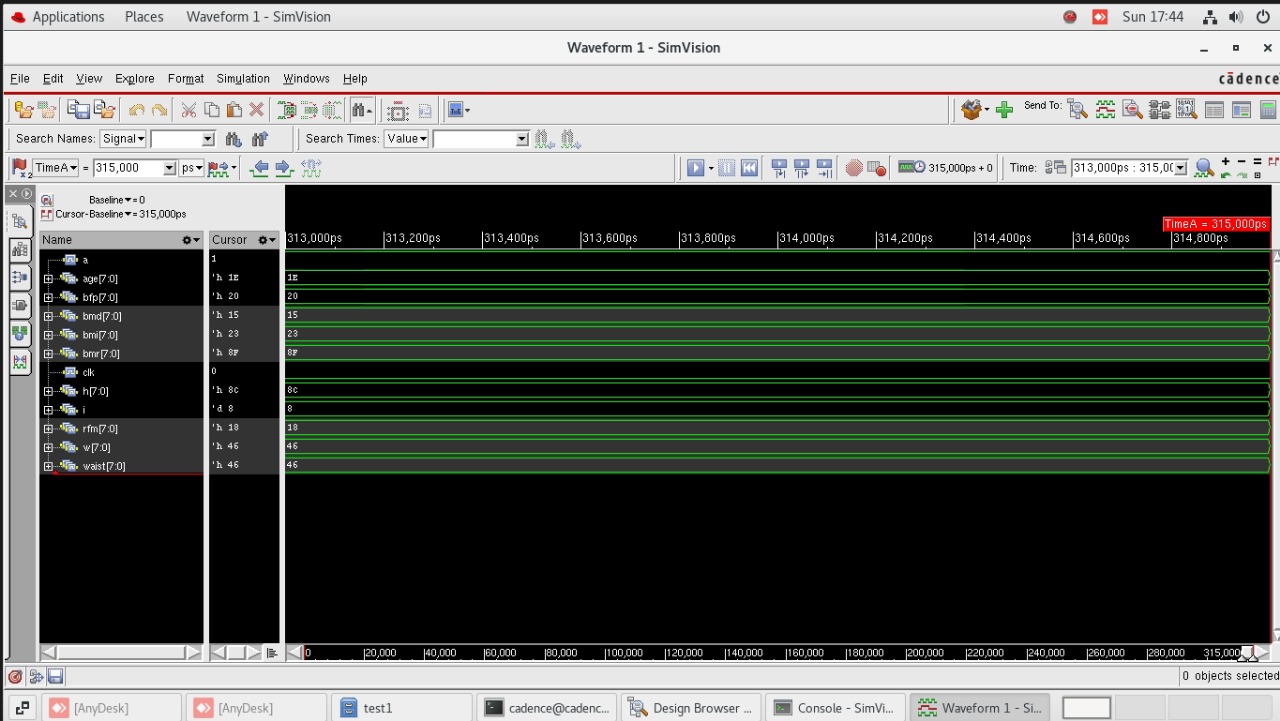
## RTL and Functional Simulation

For the ASIC design of the Health Monitoring System, functional simulation and synthesis were carried out using Cadence’s INCISIVE and GENUS tools.

The simulation was conducted to verify the RTL design and ensure that all functional requirements of the health monitoring system were met before moving to the physical design stage. Using the NC Launch tool from the Incisive suite, comprehensive testbenches were written to simulate different scenarios. This phase was crucial to validate:

* Correct functionality of input handling, sensor data processing, and system responses.
* Accurate outputs indicating the health status, alerts, or errors.
* Edge cases, including sensor failure and extreme values, were thoroughly tested to ensure robustness.

Once the RTL behavior was verified through simulation, the design proceeded to the synthesis phase using Cadence Genus, ensuring the design was optimized for power, area, and timing before moving on to the physical implementation.



**Schematic Design:**

The schematic design of the Health Monitoring System ASIC captures the high-level architecture of the chip and defines its functional blocks, input/output interfaces, and core RTL modules.

**Inputs and Outputs:**

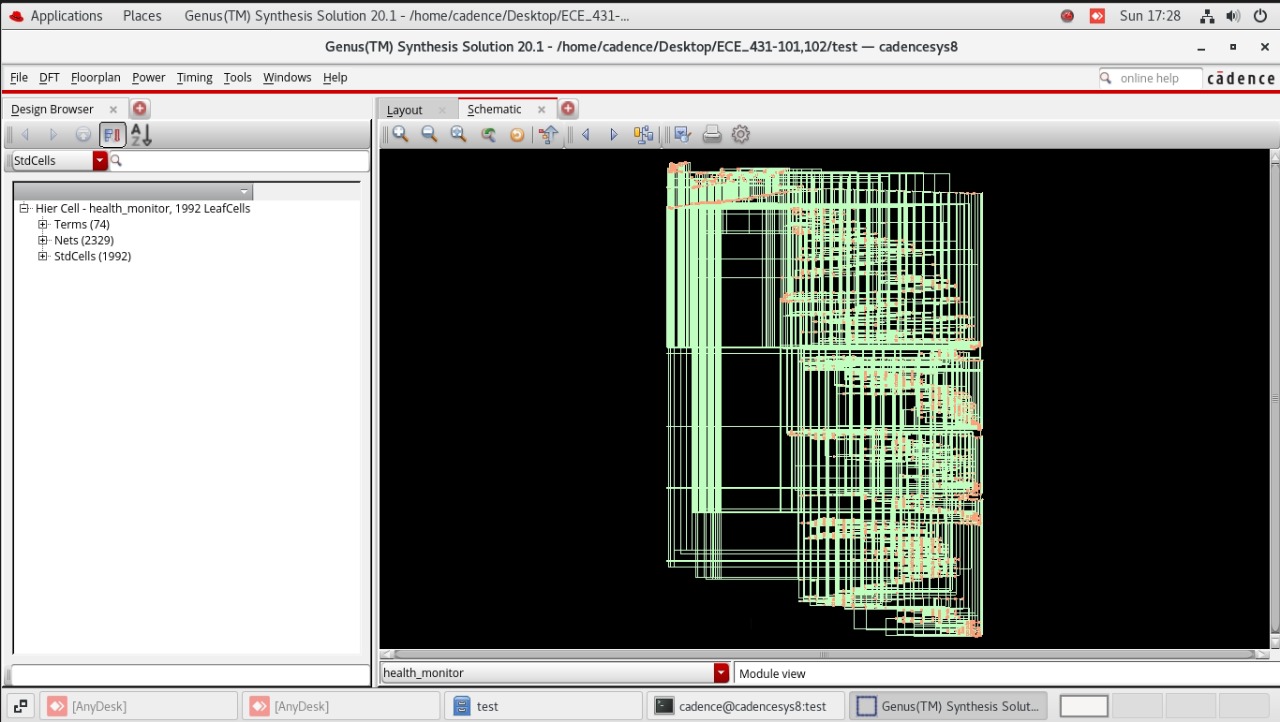
* Inputs: Clock, Reset, User Inputs (Weight, Height, Age, Gender)
* Outputs: Calculated Health Parameters (BMI, BMR, BFP, RFM, BMD)

**RTL Components:**

* Input Register Block: Captures and stores user input values for health parameter computation.
* Computation Units:
  + BMI (Body Mass Index) Calculator
  + BMR (Basal Metabolic Rate) Calculator
  + BFP (Body Fat Percentage) Calculator
  + RFM (Relative Fat Mass) Calculator
  + BMD (Bone Mass Density) Estimator
* FSM Controller: A finite state machine that sequences operations across all computation modules and controls output flow.
* Output Register Block: Holds the final computed values for output after calculations.

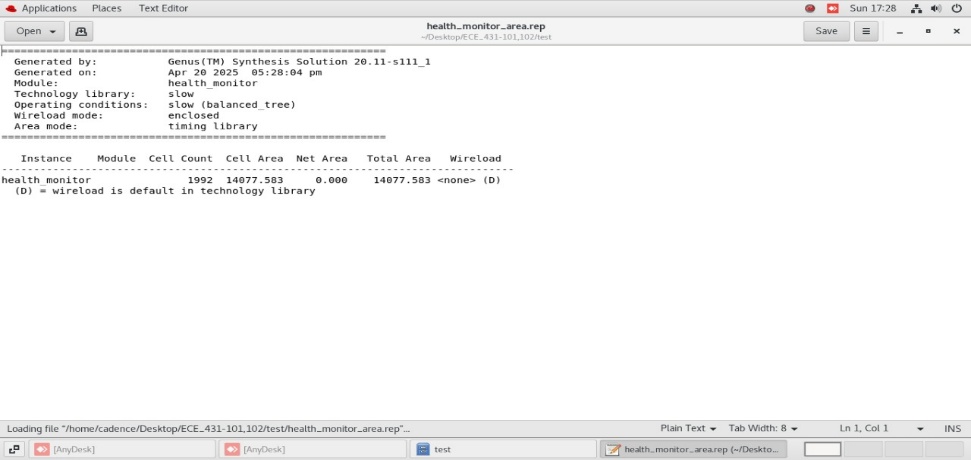
**Key Highlights of the Health Monitoring Design:**

* Implements multi-metric health analysis in a fully digital RTL architecture.
* Optimized for hardware implementation with pipelined stages and modular blocks.
* Designed to be scalable and power-efficient for ASIC integration.
* Simulated and verified using Cadence NC Launch to ensure accurate computation of all health metrics across valid and edge-case input ranges.
* Fully validated and prepared for downstream stages like synthesis, DFT, and physical design using Cadence Genus and Innovus.

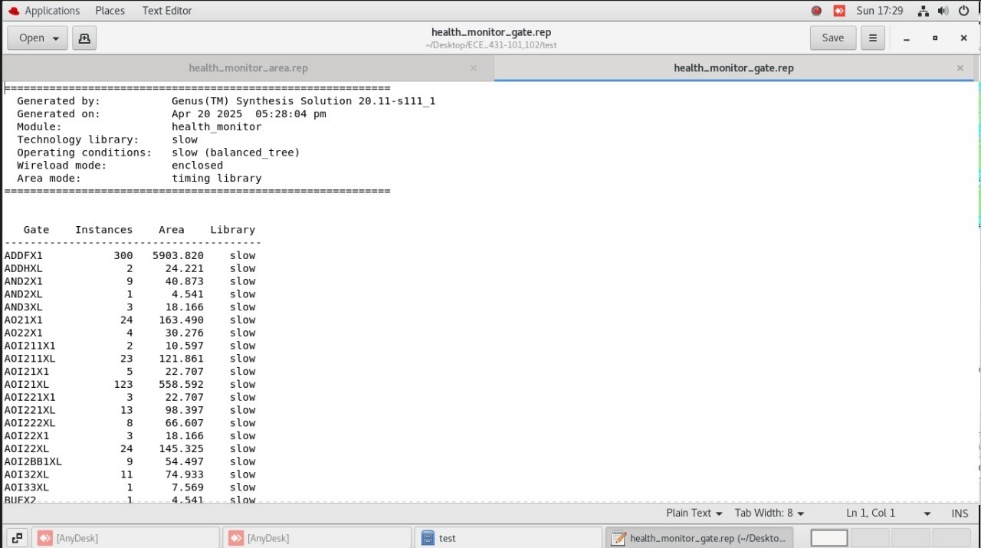


After genus reports will be generated :-

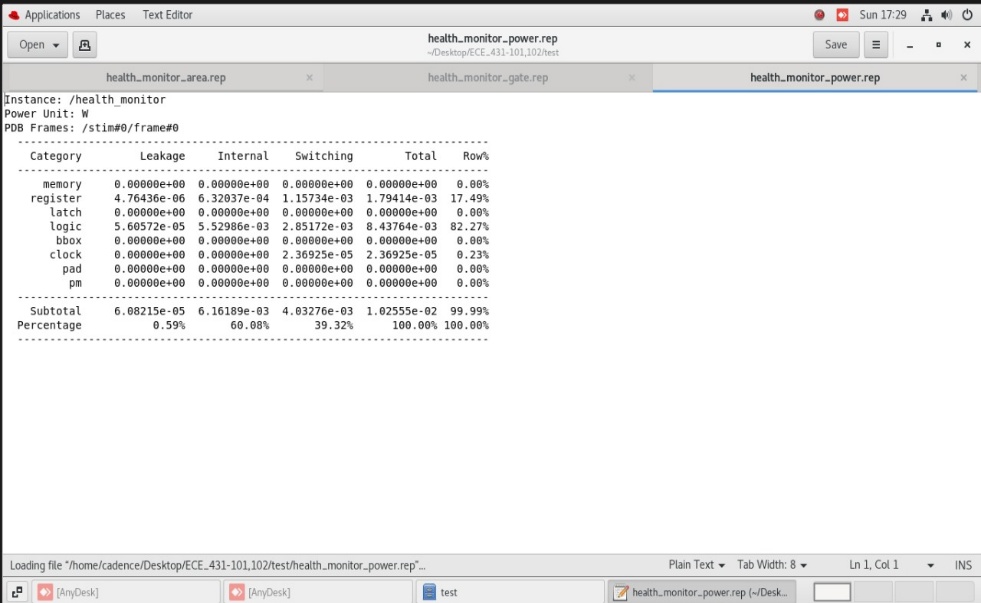
* Area report



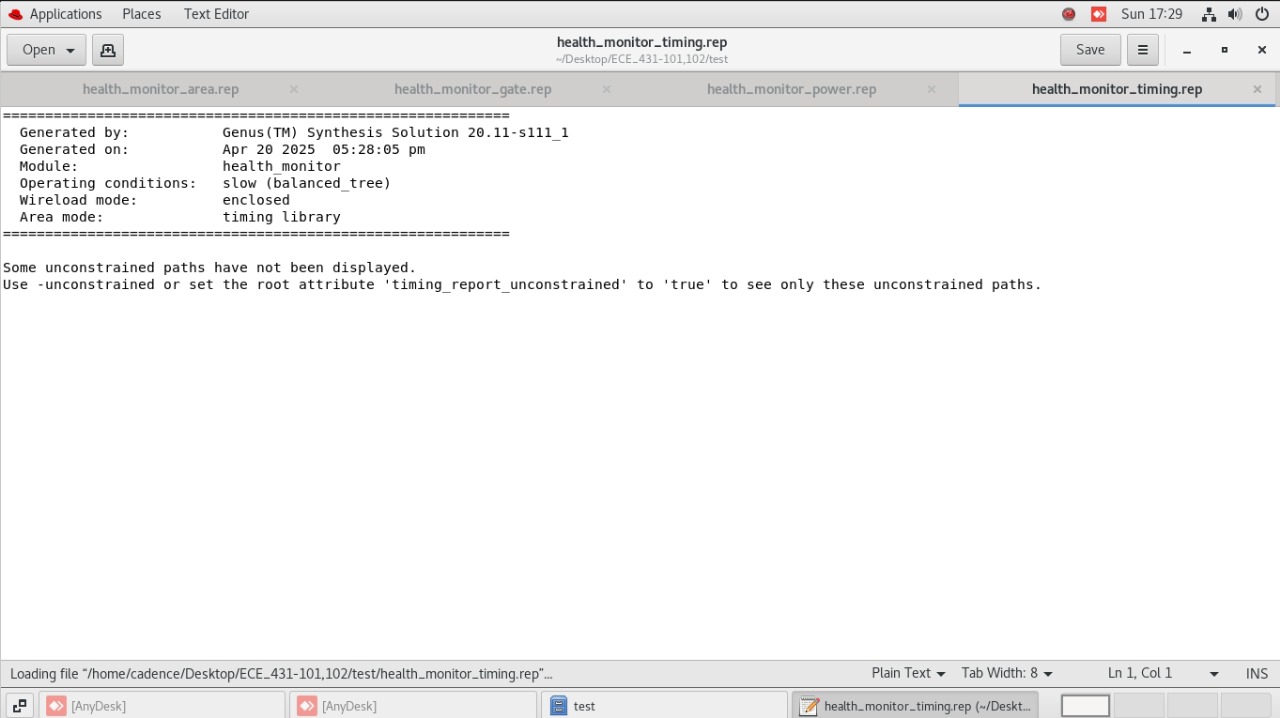
* Gate report



* Power Report



* Timing report



**Code Coverage:**

Code coverage analysis was conducted using the Cadence IMC (Integrated Metrics Center) tool to evaluate the thoroughness of the simulation for the Health Monitoring System ASIC. This step was essential to ensure that the RTL (Register Transfer Level) design was comprehensively exercised before moving on to synthesis and physical implementation.

**Purpose of Code Coverage in Health Monitoring ASIC:**

Code coverage measures how effectively simulation testbenches explore the RTL logic. It includes checking:

* Line coverage (executed lines of code)
* Toggle coverage (changing signal states)
* Condition and expression coverage
* FSM state and transition coverage

**Coverage Highlights:**

A code coverage score of over 95% was achieved, indicating a robust simulation strategy. The high coverage validates that key functional paths were exercised, including:

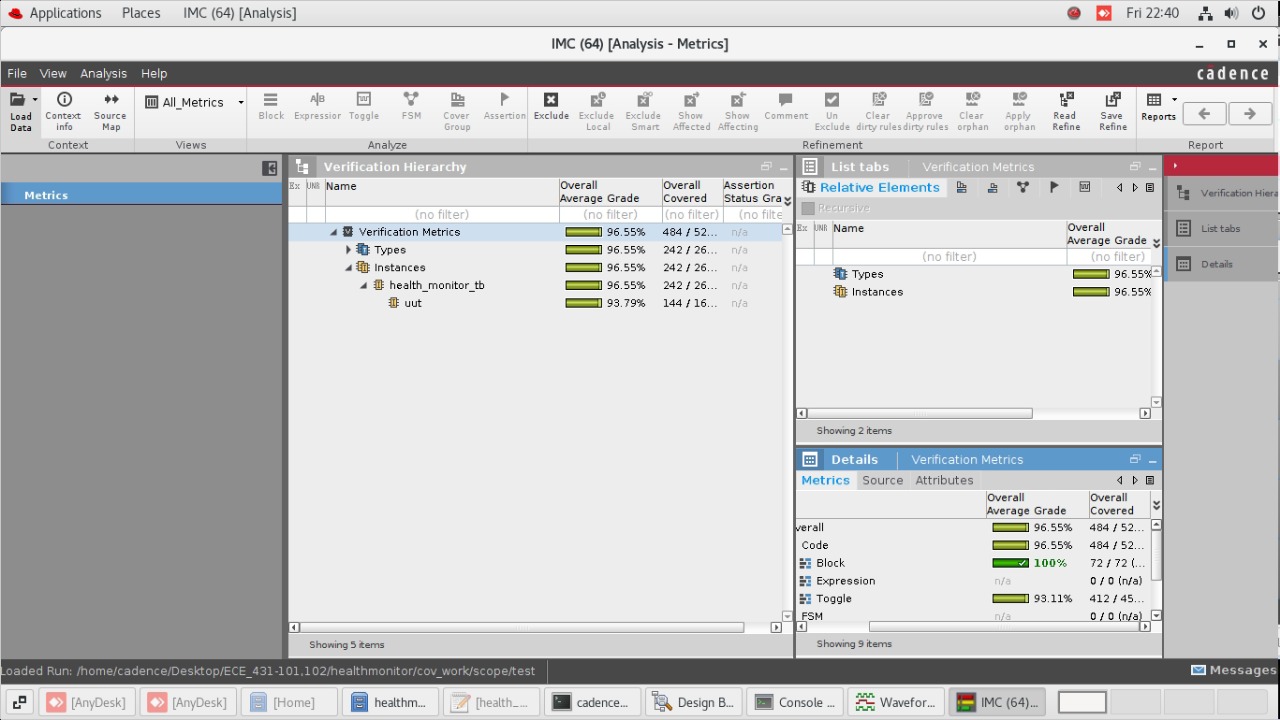
* All health parameter calculations: BMI, BMR, BFP, RFM, and BMD modules under different input conditions.
* Extreme values and boundary cases (e.g., minimum and maximum user weight, height, and age).
* Input validation and reset behavior for ensuring proper reinitialization.
* FSM transitions in control logic for triggering each module and output stage.

**Impact on Design Confidence:**

This level of coverage provided strong assurance that:

* Corner cases and realistic user scenarios were properly verified.
* The control FSM and all calculation modules were functionally complete.
* Unexpected behavior or dead code paths were identified and resolved early.

By achieving high code coverage, the Health Monitoring System ASIC was proven to be functionally reliable, reducing the risk of design bugs and ensuring a smoother transition into synthesis, DFT, and physical design stages.

****

**Linting:**

Linting is a form of static RTL analysis used to detect potential design issues early in the ASIC development flow. For the Health Monitoring System ASIC, linting was performed using the Cadence GENUS tool to verify the integrity and quality of the Verilog RTL code prior to synthesis.

**Purpose of Linting in Health Monitoring ASIC:**

**Identify and resolve common RTL issues like:**

* Undriven or unused signals
* Redundant logic constructs
* Incorrect bit-widths
* Conflicting signal assignments
* FSM state encoding errors

**Modules Analyzed:**

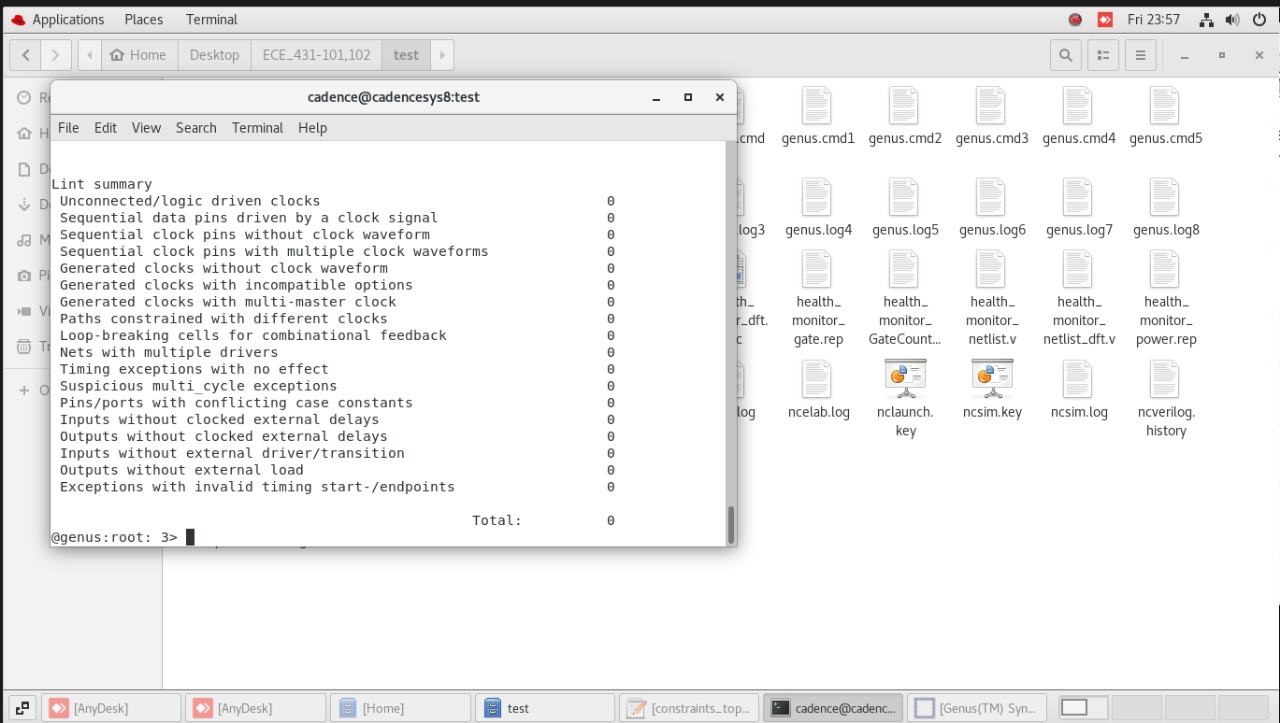
**Linting was applied across major functional blocks, including:**

* BMI, BFP, RFM, BMR, and BMD calculation units
* Control FSM for data flow and output logic
* Input handling and register interface

**Key Issues Detected and Resolved:**

* Unused wires and temporary signals in arithmetic calculation paths.
* Width mismatches between inputs/outputs of health parameter modules.
* Improperly connected clock/reset signals in the FSM and register banks.
* Unreachable FSM states or inconsistent encoding detected in the control logic.

By resolving these issues early in the flow, the RTL codebase was made synthesis-ready, robust, and compliant with design standards. This proactive verification step ensured reduced risk of simulation mismatches, improved overall design quality, and streamlined the downstream synthesis and physical implementation stages.



**Design for Test (DFT)**

Design for Test (DFT) is a vital process in the ASIC design flow that ensures the fabricated chip can be thoroughly tested for manufacturing defects. For the Health Monitoring System ASIC, DFT was implemented using the Cadence MODUS tool to insert scan chains and improve the observability and controllability of internal logic during testing.

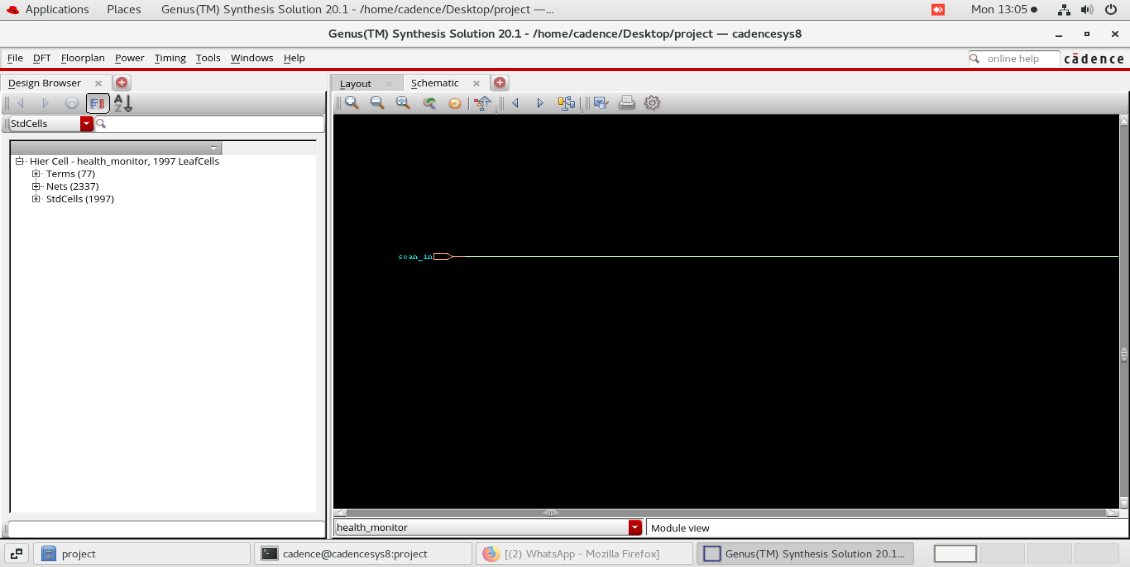
**Purpose of DFT in Health Monitoring System ASIC:**

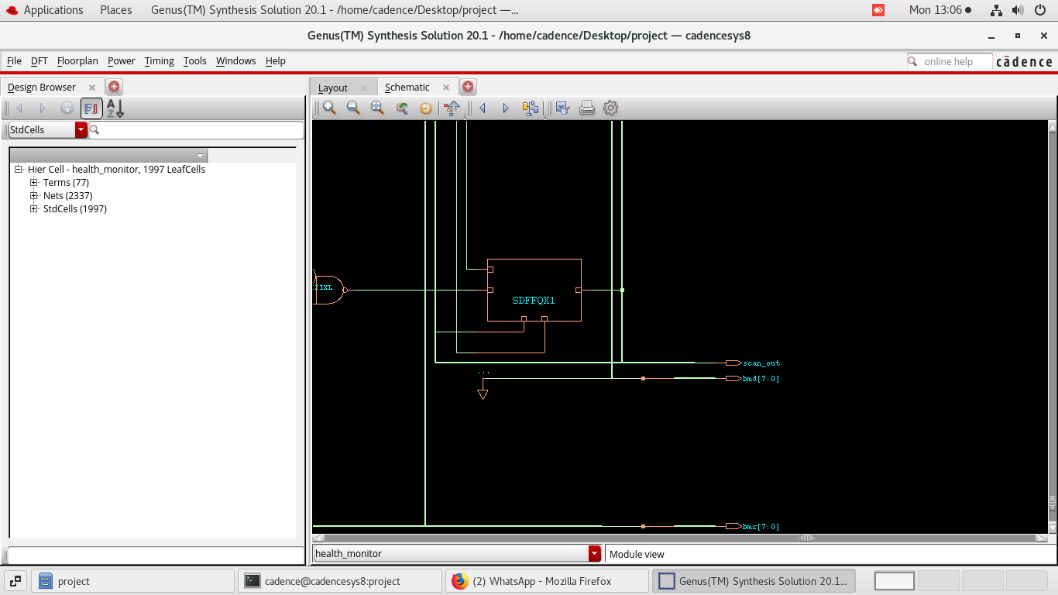
* Enable thorough testing of internal nodes within health metric modules such as BMI, BFP, RFM, BMR, and BMD calculators.
* Detect manufacturing defects like stuck-at and transition faults post-fabrication.
* Achieve high fault coverage to guarantee reliability and accuracy in health-related computations**.**

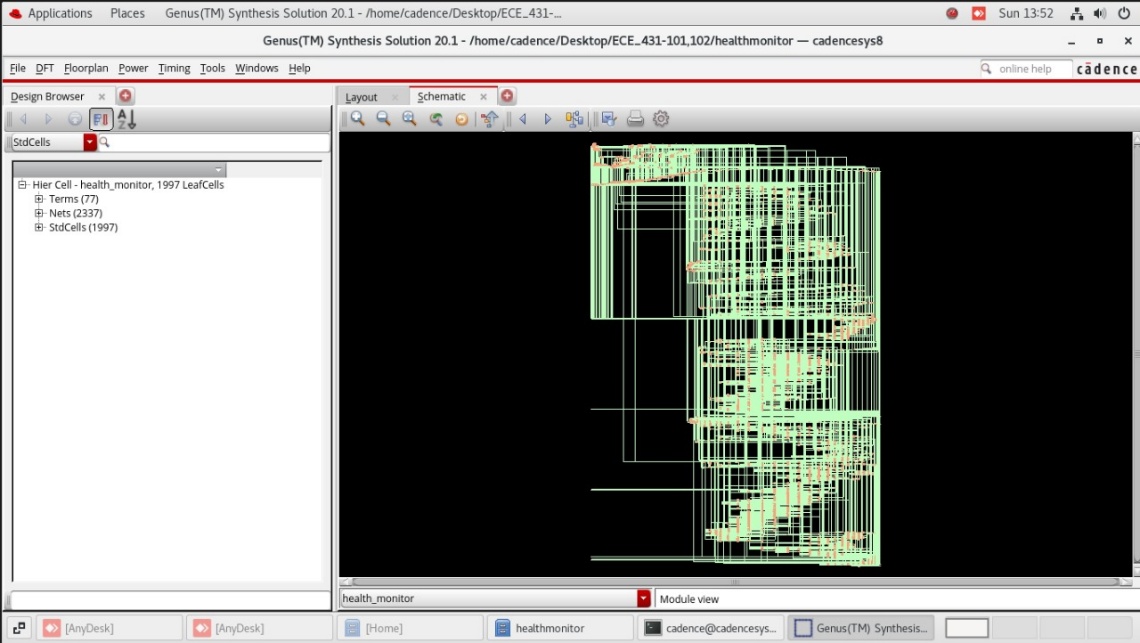
**Key Steps Performed in DFT Using MODUS:**

1. Scan Insertion:  
   All flip-flops across key computational blocks and control FSMs were converted into scan cells, forming scan chains that enable internal states to be shifted in and out for testing.
2. ATPG Pattern Generation:  
   Automatic Test Pattern Generation (ATPG) was used to create test vectors targeting stuck-at and transition faults, especially in the arithmetic and control logic of the health parameter modules.
3. Fault Simulation and Coverage Analysis:  
   The generated test patterns were simulated to measure fault coverage, which exceeded 95%, ensuring strong testability of the design.

By incorporating DFT, the Health Monitoring System ASIC was made robust against potential manufacturing defects, ensuring that every chip produced can be tested effectively without impacting the performance, area, or accuracy of its core health calculations. This is especially critical for systems where precision and dependability directly impact health assessments.







### LEC (Logic Equivalence Check)

Logic Equivalence Check (LEC) is a critical verification step in the ASIC design flow, ensuring that the synthesized netlist is functionally identical to the original RTL design. For the Health Monitoring System ASIC, LEC was carried out using Cadence CONFORMAL to validate that the synthesis process did not introduce any unintended functional changes to the health metric computation logic.

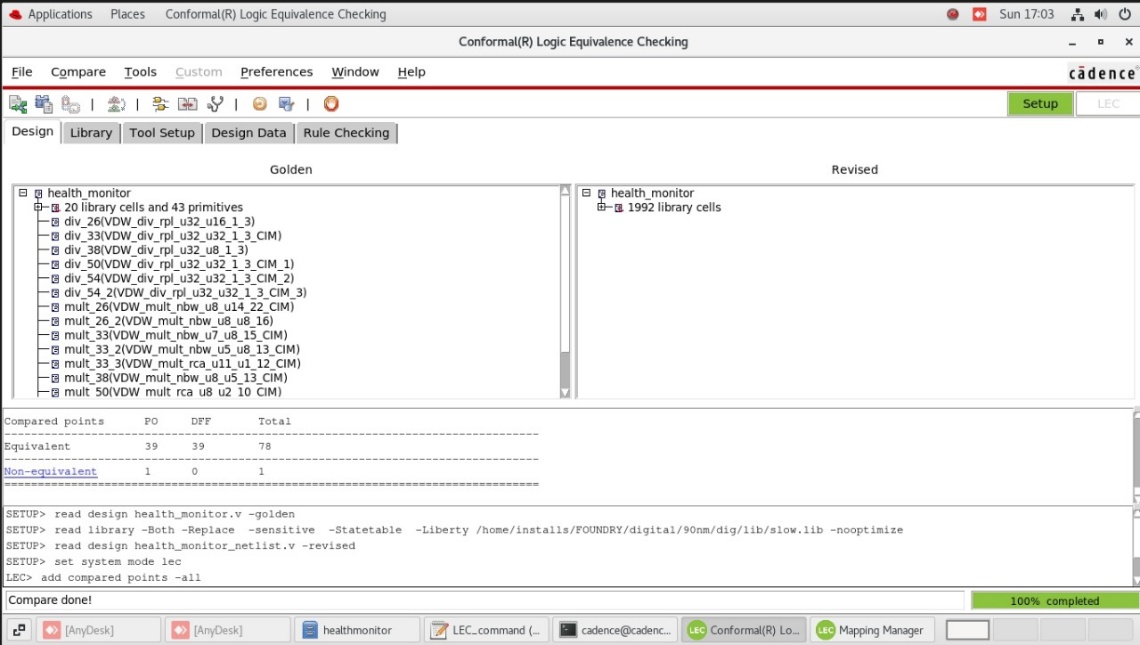
**Purpose of LEC in Health Monitoring System:**

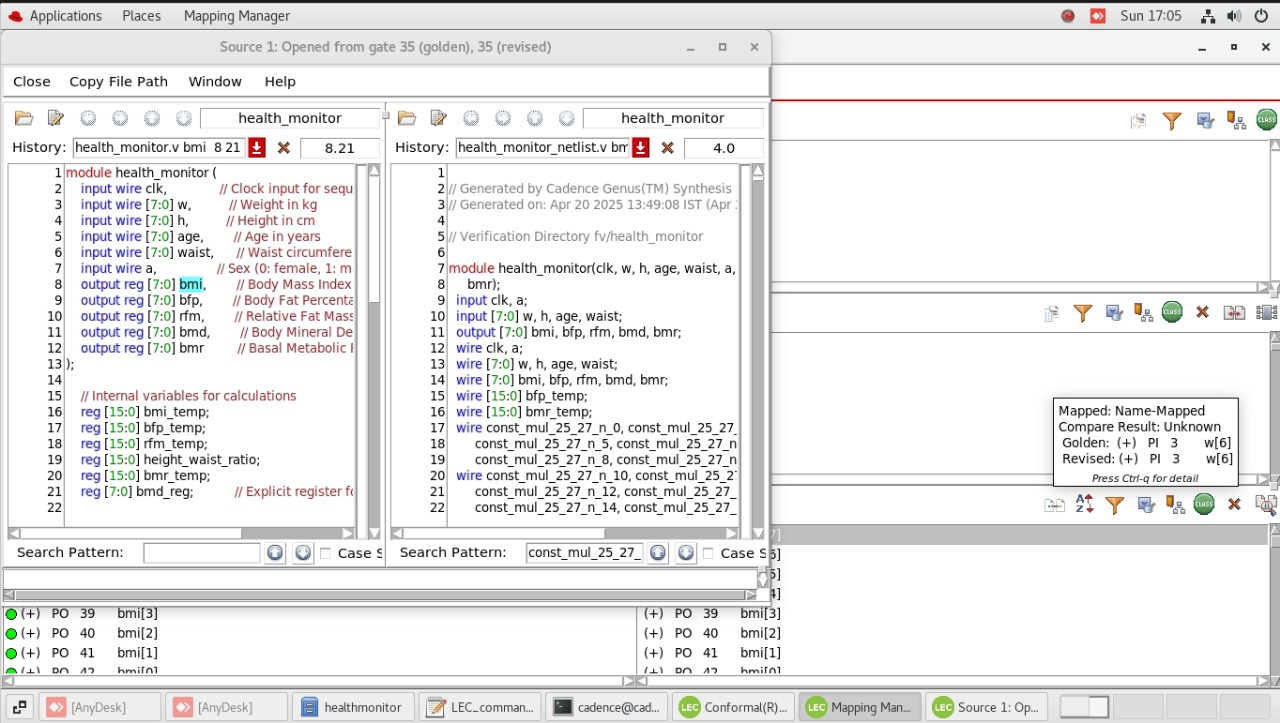
* Validate Functional Consistency:  
  Confirm that essential modules—such as BMI, BFP, RFM, BMR, and BMD calculation blocks—function identically before and after synthesis.
* Preserve Control Logic:  
  Ensure that sequential control governing data flow, health metric updates, and conditional operations (e.g., gender-based calculations) remained unaffected through logic optimizations.
* Secure Calculation Accuracy:  
  Verifying equivalence helps guarantee accuracy of health metrics like body fat percentage and basal metabolic rate, which are vital for reliable health monitoring.

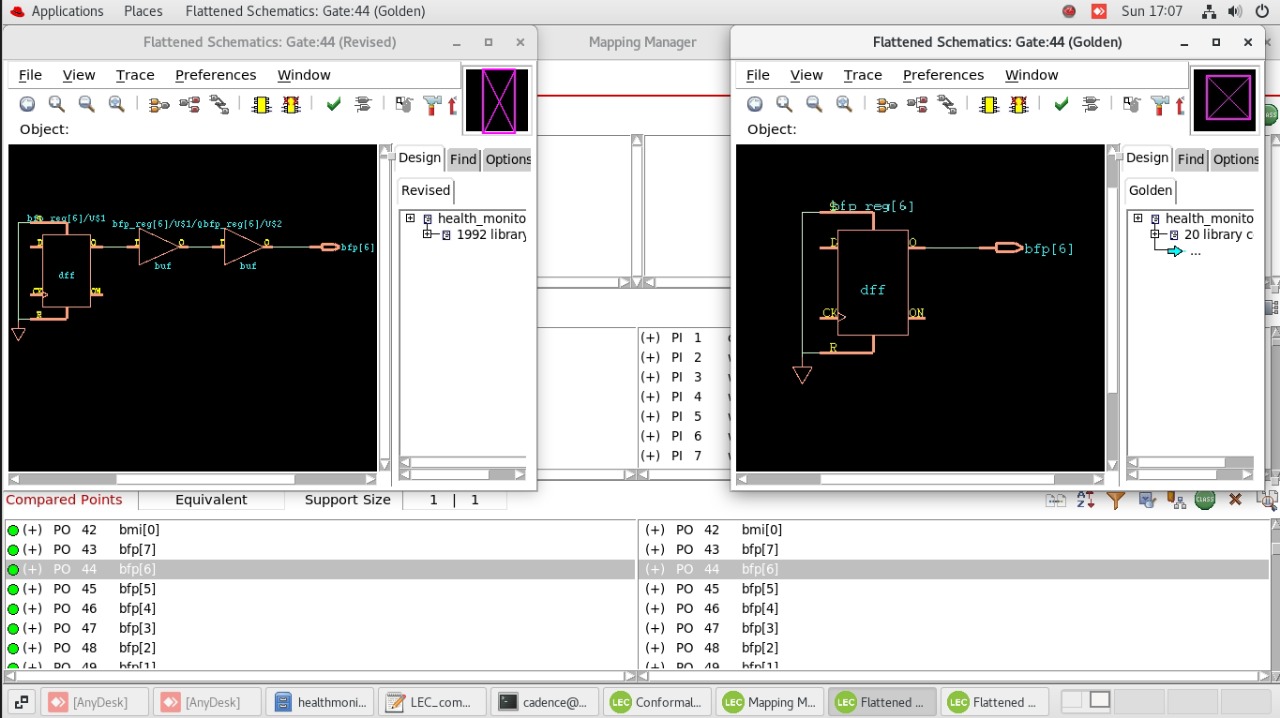
**Key Steps in LEC Using CONFORMAL:**

1. Load RTL and Gate-Level Netlists:  
   The RTL code (golden model) and the synthesized netlist (revised model) were imported into CONFORMAL.
2. Equivalence Mapping and Checks:  
   The tool automatically mapped corresponding modules and signals, then performed structural and functional checks to detect mismatches.
3. Verification Results:  
   CONFORMAL reported no mismatches, confirming that all logic transformations during synthesis preserved the original functionality.

By successfully completing LEC, the Health Monitoring System ASIC was proven to be functionally equivalent across both RTL and gate-level implementations. This assurance is crucial for a system that provides health-critical calculations, reinforcing the design's reliability, correctness, and readiness for fabrication.







### ATPG(Automatic Test Pattern Generation):

Automatic Test Pattern Generation (ATPG) is a vital phase in the ASIC design process, used to verify the manufacturability and testability of a digital circuit after fabrication. For the Health Monitoring System ASIC, ATPG was performed using the Cadence MODUS tool to ensure high fault coverage and enable effective post-fabrication testing.

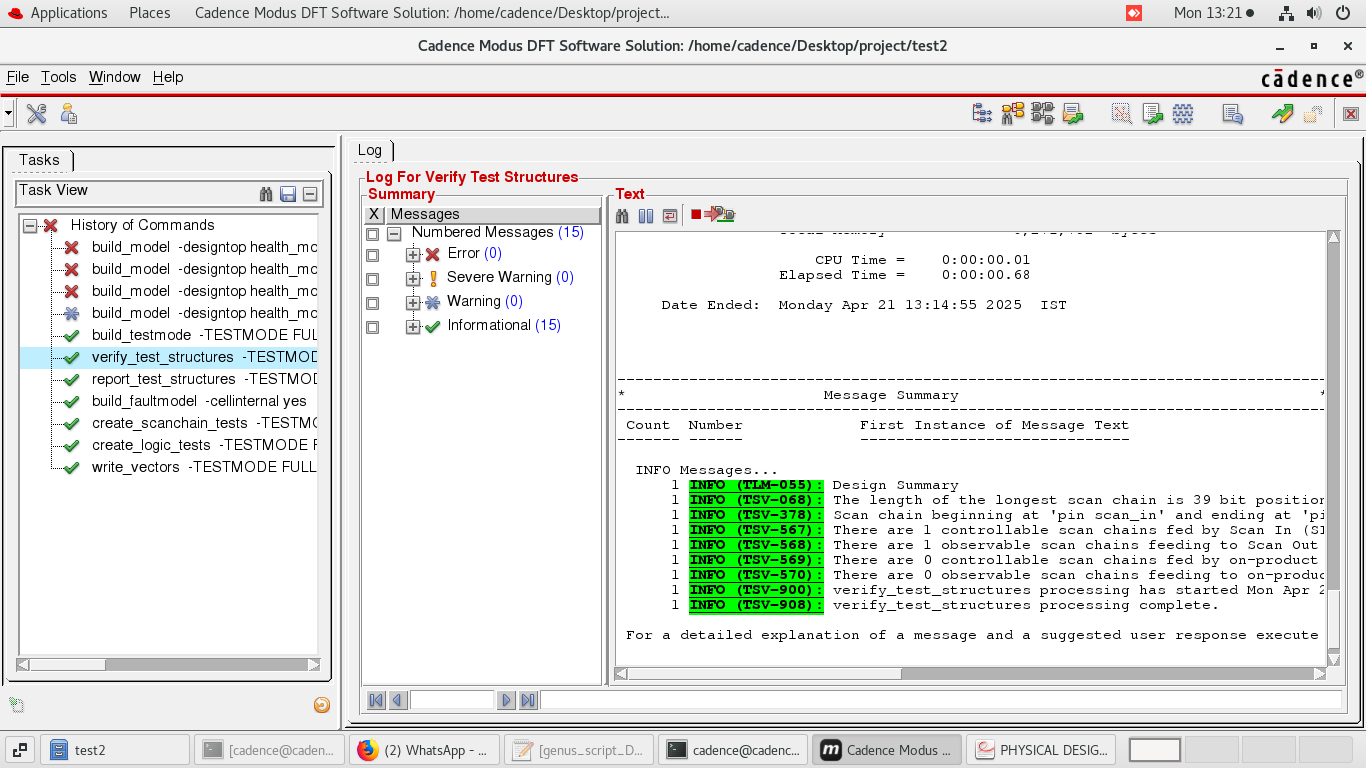
**Objectives of ATPG in Health Monitoring System:**

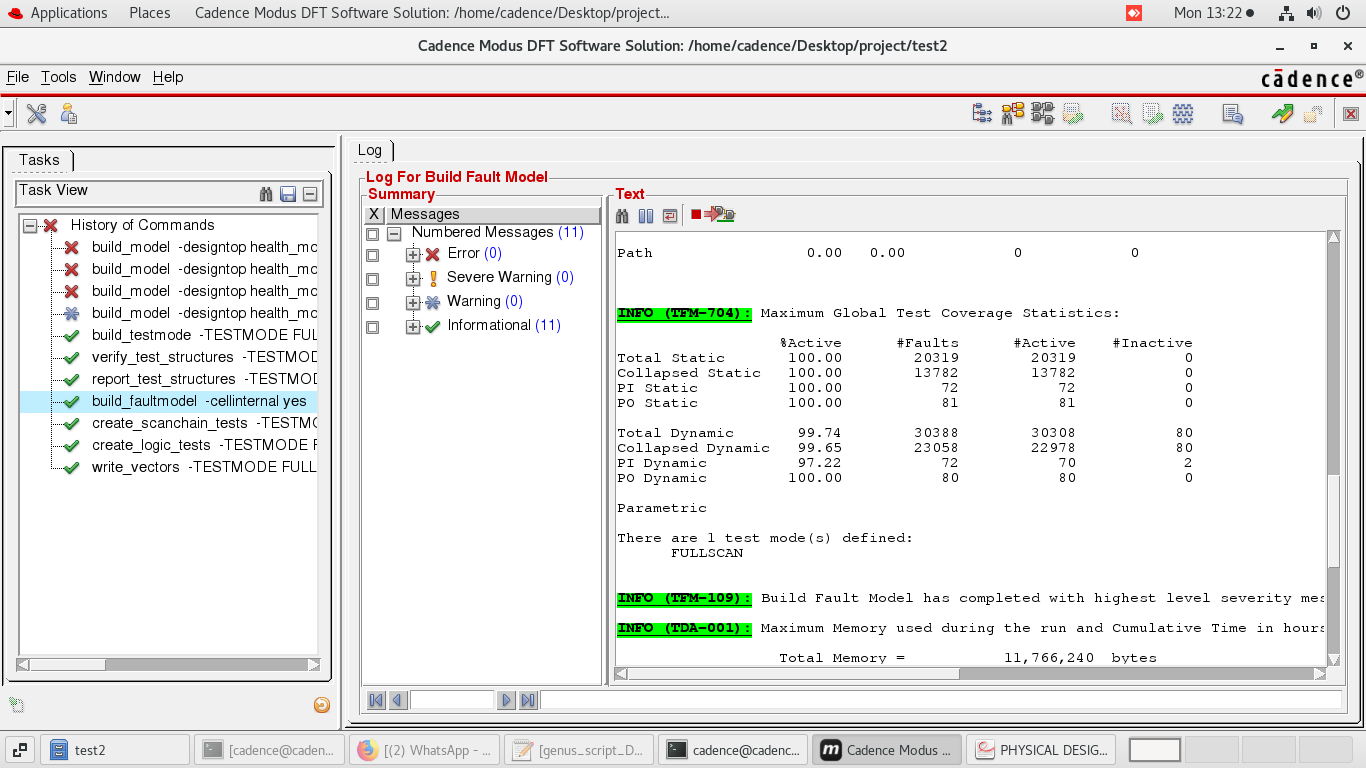
* Detect Manufacturing Defects:  
  ATPG was used to identify stuck-at faults and transition faults in key computation modules such as the BMI, BFP, and BMR calculation blocks, as well as control and data routing logic.
* Generate Efficient Test Vectors:  
  MODUS automatically generated test patterns that can be applied during production testing to verify that the chip operates correctly, helping to filter out defective units before deployment.
* Ensure Testability of Critical Paths:  
  The test strategy focused on paths involved in real-time health calculations, which are critical for system accuracy and reliability. Ensuring high testability for these blocks adds robustness to the design.

**Steps Involved in ATPG:**

1. Scan-Inserted Netlist Import:  
   The netlist containing Design-for-Test (DFT) features such as scan chains was loaded into the MODUS tool.
2. Fault Model Selection:  
   Standard fault models including stuck-at and transition faults were selected to target both static and dynamic faults.
3. Test Pattern Generation:  
   MODUS generated optimized test vectors aimed at maximizing fault detection while minimizing the number of patterns, which helps reduce test time and cost.
4. Fault Coverage Report:  
   The ATPG tool produced a detailed report showing high fault coverage (>95%), confirming that the design is highly testable and reliable for mass production.

By completing the ATPG process, the Health Monitoring System ASIC was validated for post-silicon testing, ensuring that any chip with manufacturing defects can be efficiently detected and discarded. This step enhances overall design reliability, especially for systems intended to monitor health metrics accurately and consistently.





### INNOVUS Implementation

**1. Floor Planning in Innovus for Health Monitoring System**

Floor planning is the foundational stage of physical design, where the physical organization of the chip is defined. For the Health Monitoring System ASIC, which includes key functional blocks such as arithmetic units for BMI, BFP, RFM, BMD, and BMR calculations, proper floor planning was essential to optimize area, timing, and power consumption.

**Key Floor Planning Tasks:**

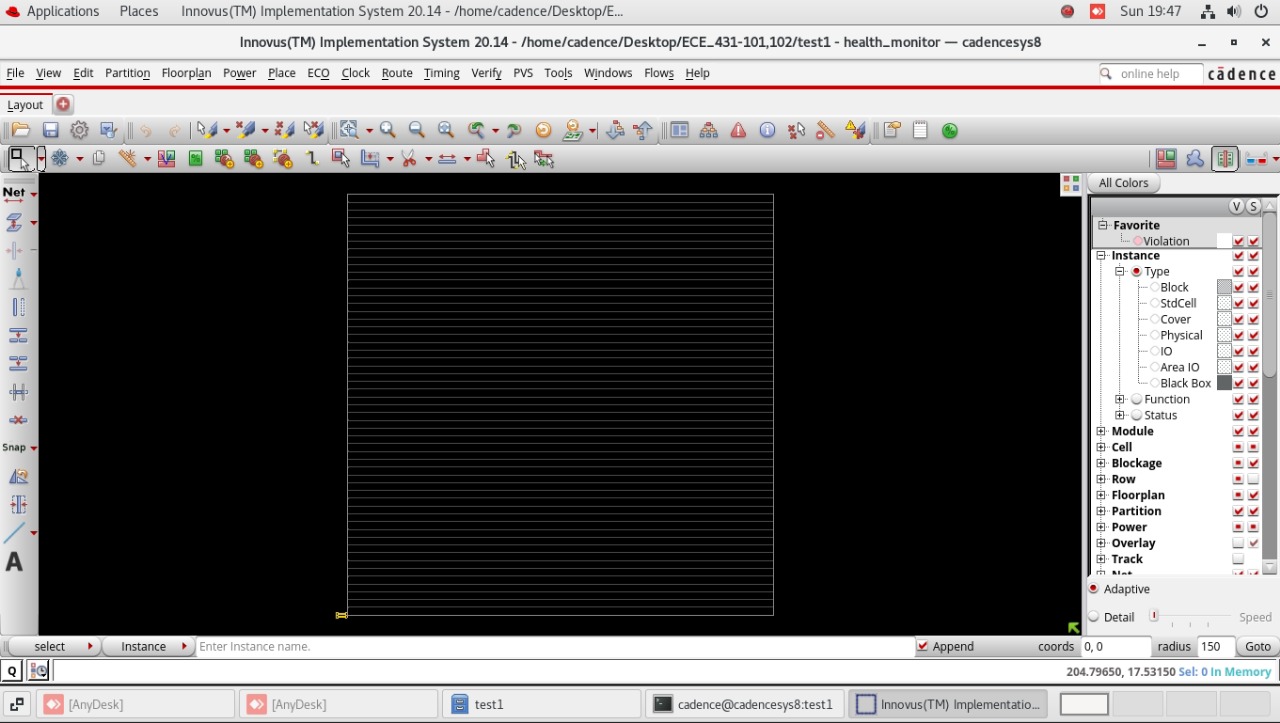
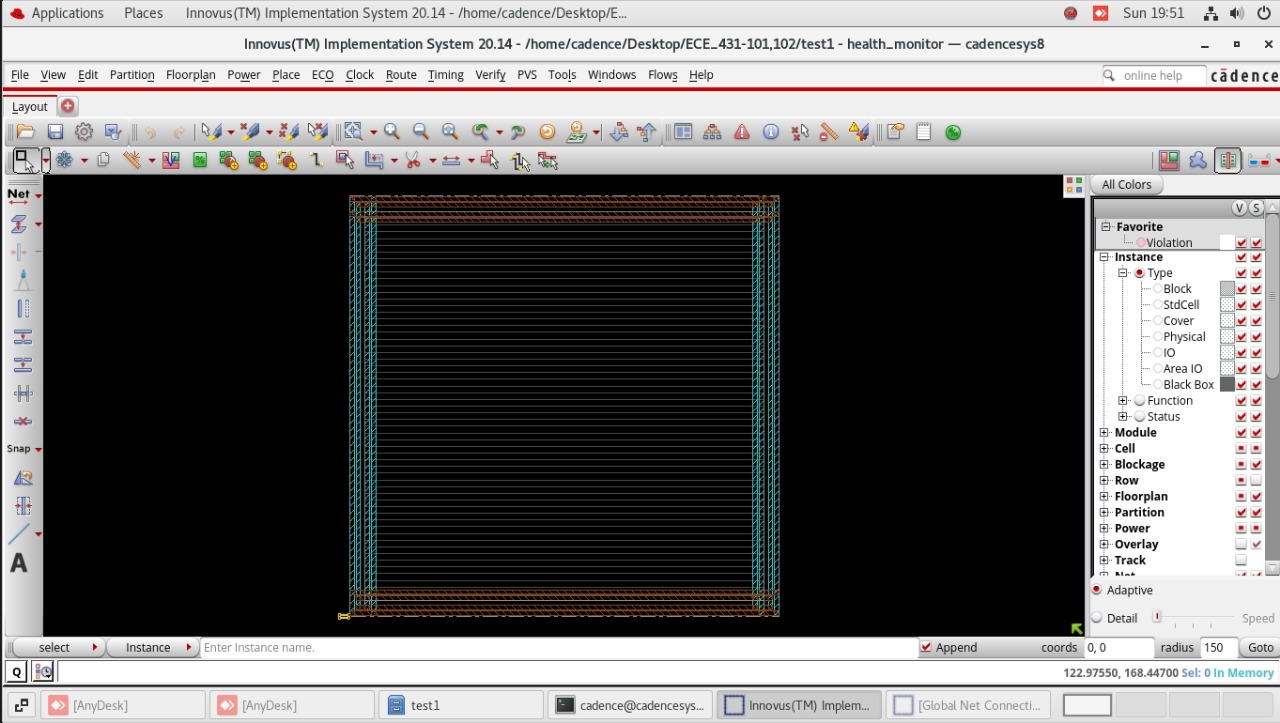
* Core Area Definition:  
  The core area was determined based on the logic complexity of the modules required for real-time health computations. This included computational blocks, control logic, and intermediate data storage.
* Placement of Functional Blocks:  
  The health metric calculation modules were placed logically and physically close to reduce wirelength and interconnect delay. This ensured fast data propagation between arithmetic blocks and minimized power consumption.
* I/O Pin Placement:  
  Pins for clock, input signals (weight, height, age, waist, and sex), and output signals (BMI, BFP, RFM, BMD, BMR) were placed strategically along the periphery. This minimized routing congestion and helped meet timing for high-speed input/output interfaces.
* Aspect Ratio and Block Allocation:  
  A balanced aspect ratio was selected to ensure compactness without causing routing congestion. Each module was allocated area proportional to its complexity and connectivity.
* Floorplan Constraints:  
  Blockages and keep-out zones were defined around sensitive computation areas to reserve space for routing and avoid timing-critical violations. Innovus was used to enforce these constraints and maintain a clean layout.

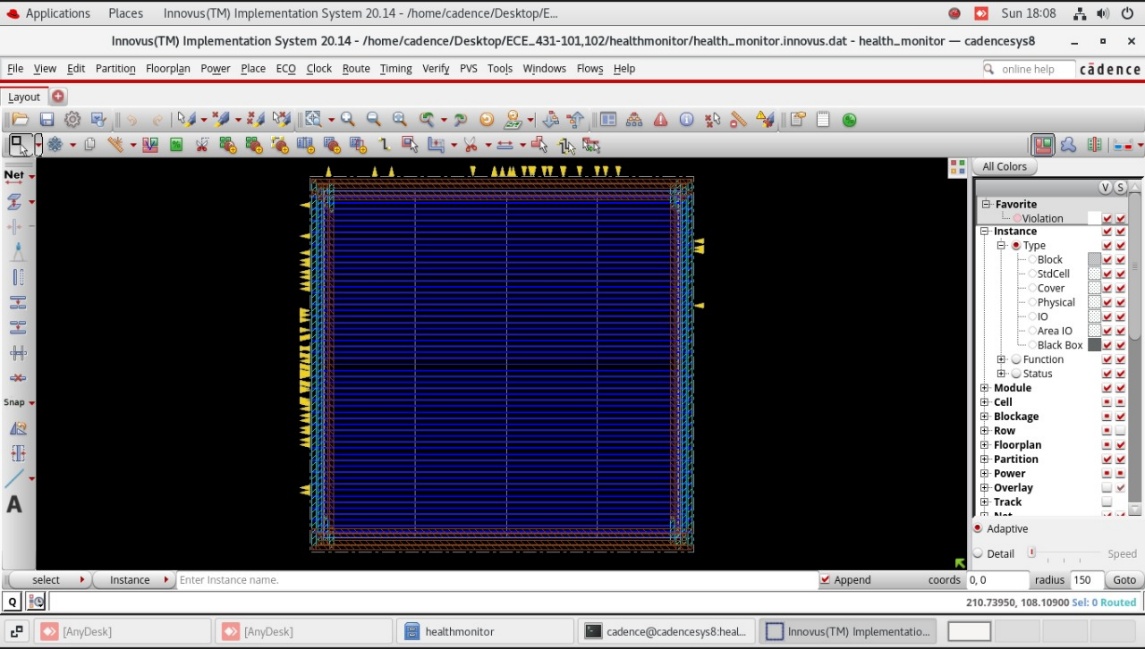
**2. Power Planning in Innovus for Health Monitoring System**

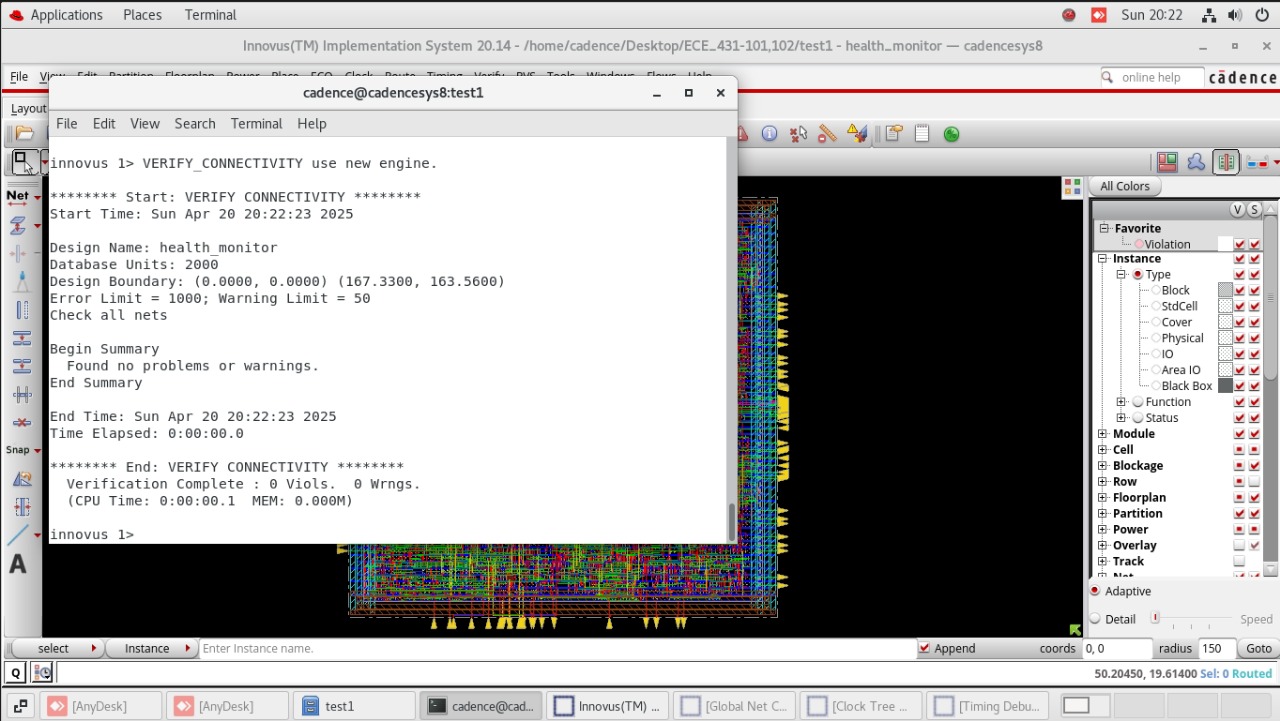
Power planning ensures robust and consistent delivery of power across the chip. Given the real-time nature of health monitoring applications, power integrity was a top priority to guarantee accurate and uninterrupted processing of health metrics.

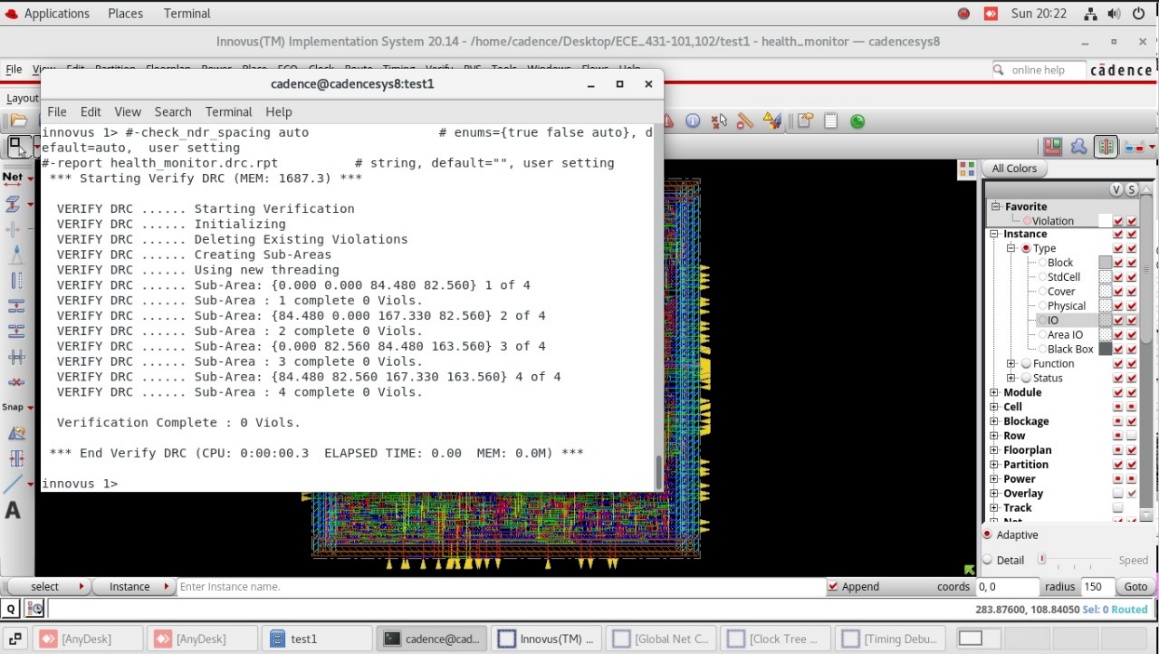
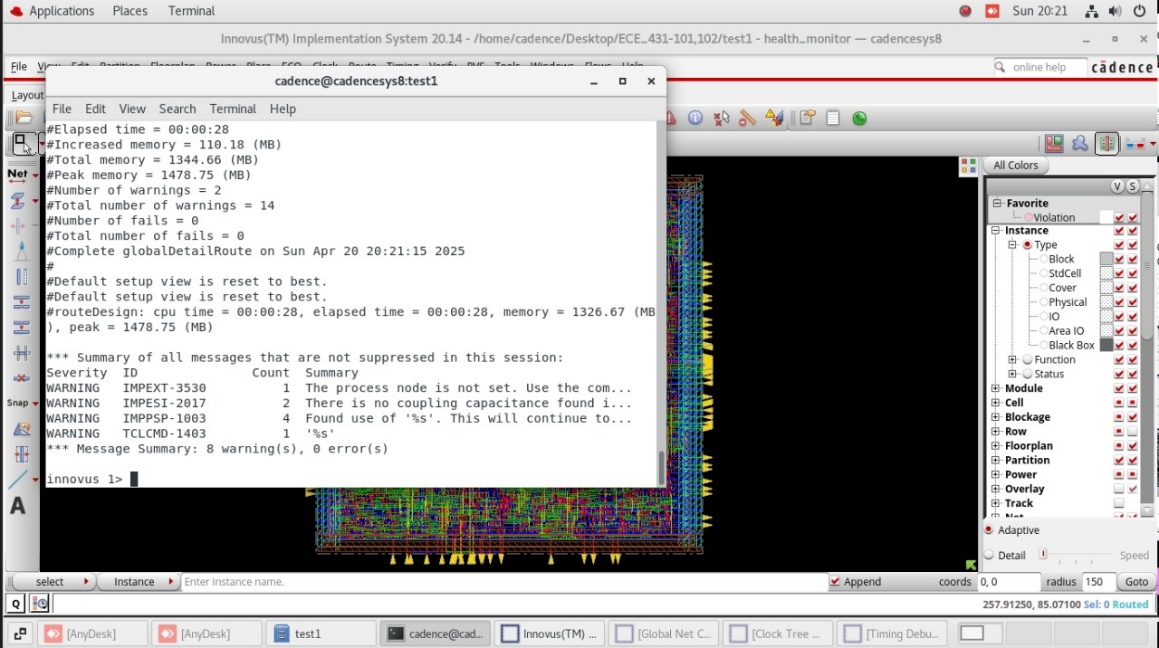
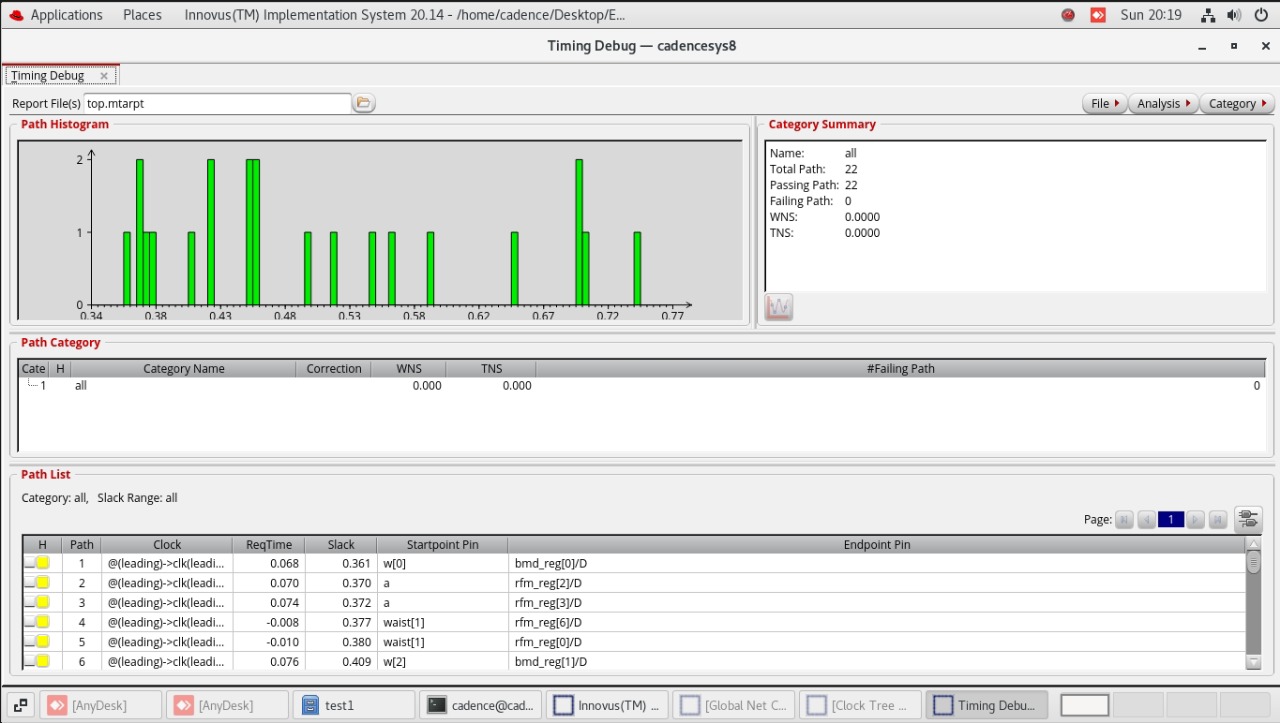
**Key Power Planning Tasks:**

* Power Grid Design:  
  Cadence Innovus was used to design a structured power grid that included power rings around the core and horizontal/vertical power stripes throughout the layout. This grid ensured even distribution of VDD and GND.
* Global Power Nets:  
  Dedicated global power nets were created to connect all standard cells and computation modules to VDD and GND. This was crucial for the stable operation of high-activity blocks like the BMI and BFP units.
* Insertion of Decap Cells:  
  Decoupling capacitors (decap cells) were inserted across the layout to mitigate noise during simultaneous switching of logic elements. These cells helped stabilize voltage supply, especially during peak computational load.
* IR Drop Analysis:  
  IR drop checks were conducted to confirm that all regions of the chip, particularly those with high current demand, received sufficient voltage. This analysis ensured power integrity and reduced risk of logic failure.
* Power Mesh Optimization:  
  A multi-layer power mesh was optimized for low resistance and high current-handling capacity. This provided a robust infrastructure to meet the power demands of the health monitoring calculations with minimal voltage droop.



**Clock Tree Synthesis (CTS) for the Health Monitoring System**

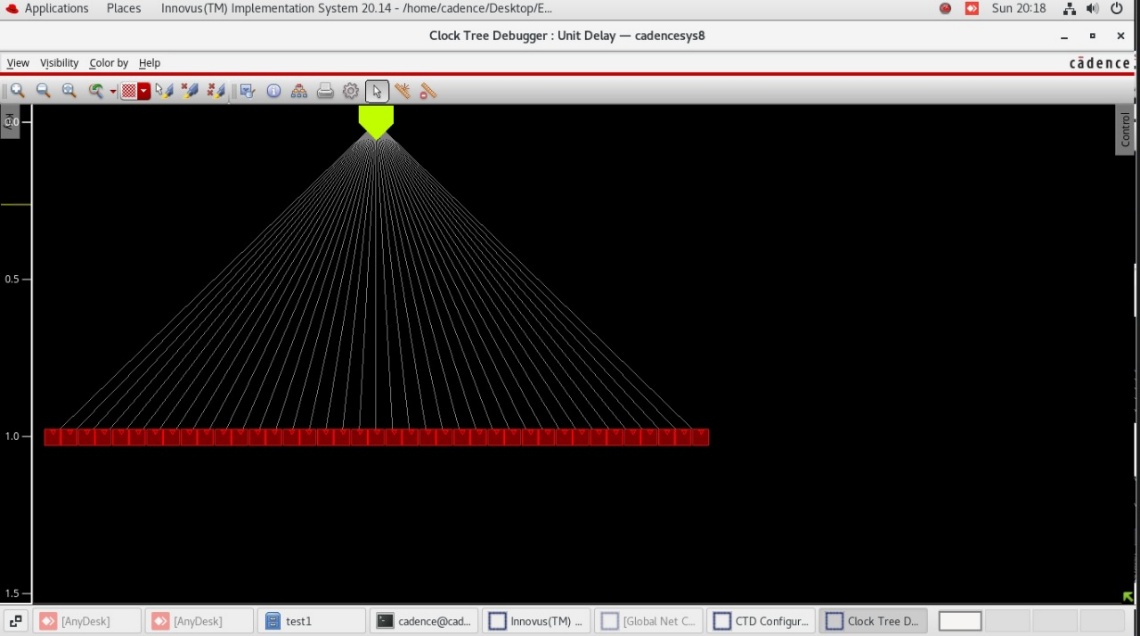
For the ASIC implementation of the Health Monitoring System, Clock Tree Synthesis (CTS) was performed using **Cadence Innovus**. CTS is a vital step in the physical design phase, responsible for distributing the clock signal efficiently to all sequential elements in the design, such as flip-flops, with minimal skew and balanced latency.

**Objective of CTS**

The primary objective of CTS in this project was to ensure consistent and reliable clock delivery to all synchronous blocks, such as the arithmetic logic responsible for computing BMI, BFP, RFM, BMD, and BMR. Proper clock distribution is critical to avoid timing violations and to guarantee that data is captured correctly across all stages of the pipeline.

**Key Tasks in CTS**

* Clock Net Identification:  
  Clock nets driving sequential logic blocks, including the main computational units and storage elements, were identified and isolated from combinational data paths. This allowed for targeted optimization during clock tree synthesis.
* Insertion of Clock Buffers:  
  Automated buffer insertion was carried out to strengthen the clock signal across the chip. These buffers helped reduce clock signal degradation, especially in areas with high sequential logic density.
* Balancing Clock Paths:  
  The CTS tool in Innovus balanced clock paths by equalizing delays between the clock source and each sequential element. This reduced clock skew, which is crucial for maintaining synchronous operation and avoiding setup/hold violations.
* Skew and Latency Report Generation:  
  Post-CTS analysis generated detailed reports on clock skew and insertion delay. These reports ensured that the clock network met the stringent timing requirements necessary for real-time health metric calculations.



**GDSII Generation**

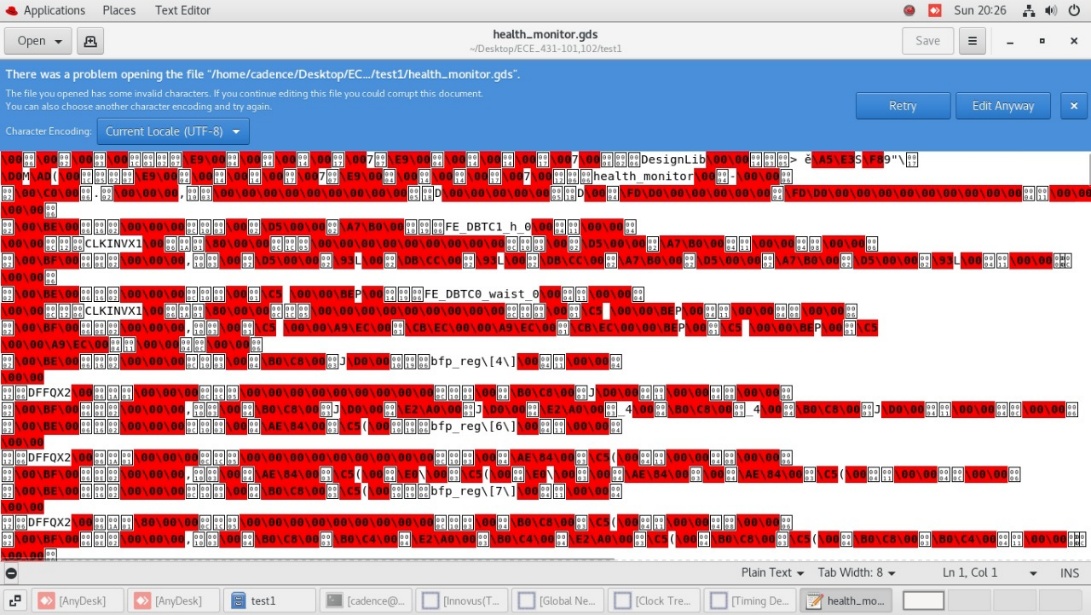
The final layout of the Health Monitoring System ASIC was exported in GDSII (Graphic Data System II) format using Cadence Innovus, which is the industry-standard format for delivering physical design data to semiconductor foundries for fabrication.

**Purpose of GDSII**

The GDSII file serves as the definitive representation of the physical layout of the chip. It includes all geometric patterns, layer information, and mask definitions required to manufacture the Health Monitoring System on silicon. This file marks the final *tape-out* version of the design, ready for fabrication.

**Steps in GDSII Generation**

* DRC and LVS Clean Layout:  
  Prior to GDSII export, the physical layout underwent rigorous checks to ensure there were no Design Rule Check (DRC) or Layout vs Schematic (LVS) violations. This ensures that the layout is both manufacturable and consistent with the functional schematic.
* Metal Fill Insertion:  
  To meet foundry-specified metal density rules, dummy metal fills were inserted into unused layout regions. These fills help prevent issues such as dishing and erosion during the chemical-mechanical polishing process, without affecting circuit functionality.
* GDSII Stream Out:  
  Using Cadence Innovus, the final verified layout of the Health Monitoring System was streamed out into GDSII format, capturing all the required layers (e.g., diffusion, polysilicon, metal, via) for photolithography and mask generation.
* Sign-Off Checks:  
  Before finalizing the GDSII, additional sign-off checks were performed, including analysis of antenna effects, signal integrity, and overall manufacturability compliance. These checks ensured that the layout met the reliability and yield standards expected in a real-world healthcare-oriented ASIC application.

****

**Static Timing Analysis (STA)**

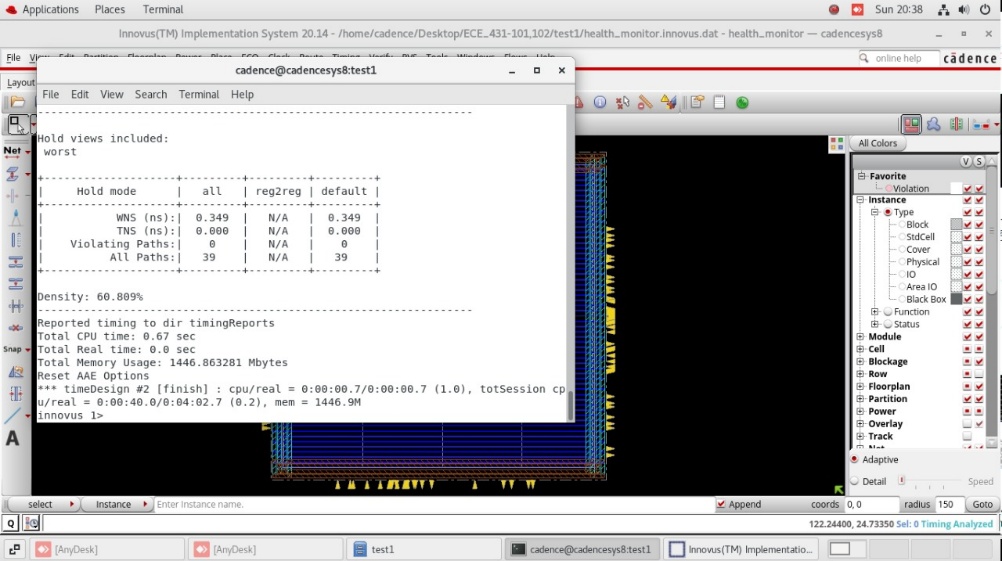
Static Timing Analysis (STA) was carried out using Cadence Tempus to ensure that the Health Monitoring System ASIC meets all timing requirements across various operating conditions. STA is a crucial verification step in digital design, helping to guarantee that the design functions reliably at the target clock frequency without any setup or hold time violations.

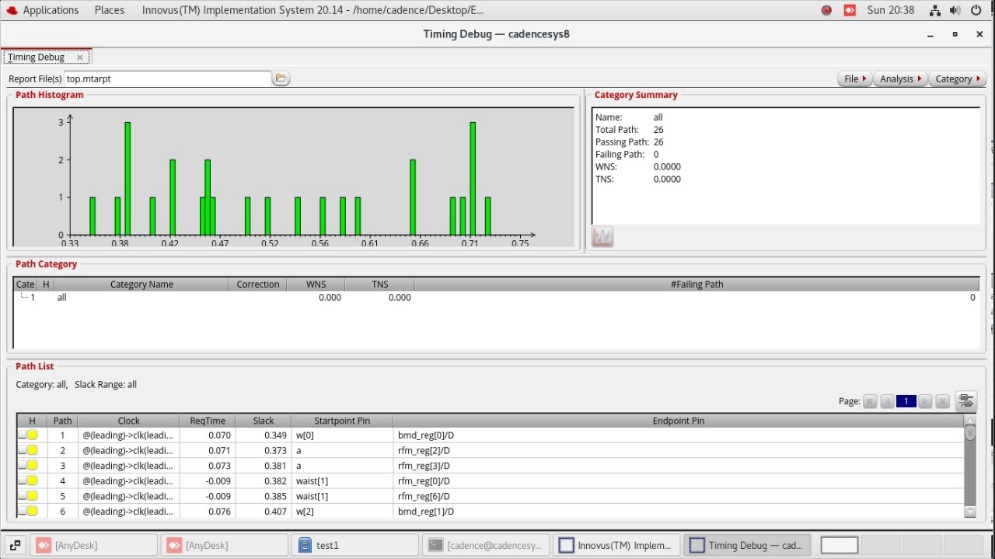
**Purpose of STA**

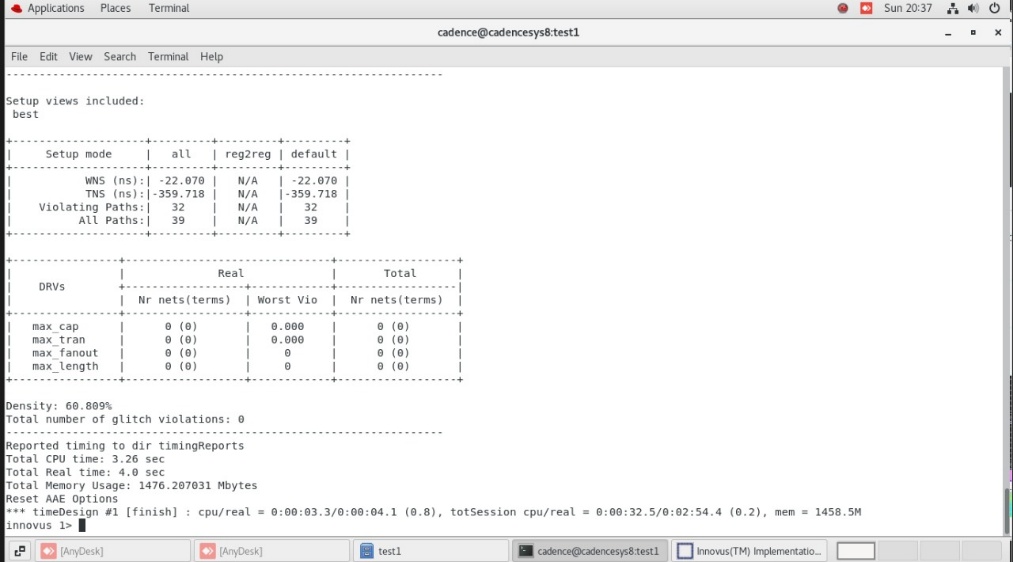
* STA validates that all data signals arrive at their destinations in sync with the clock, ensuring correct operation of all sequential logic components. For a system like the Health Monitoring Unit—where accurate, real-time computation of health parameters is essential—timing integrity is critical for consistent and error-free performance.

**STA Analysis Includes**

* Setup and Hold Checks:  
  Timing constraints were applied and analyzed to ensure that signals feeding flip-flops met the required setup and hold times. This ensures stable data capture and prevents incorrect computations of BMI, BFP, RFM, BMD, and BMR.
* Timing Paths Analysis:  
  Critical timing paths between the arithmetic logic blocks, input registers, and output registers were analyzed under worst-case (slow corner) and best-case (fast corner) conditions. This ensured that all paths operate correctly across process, voltage, and temperature (PVT) variations.
* Clock Domain Crossing (CDC) Checks:  
  Although the current design operates under a single clock domain, the system was reviewed for any potential asynchronous behavior. This is essential for maintaining data stability and preventing metastability in future expanded versions of the design.
* Slack Report Generation:  
  STA reports were generated and analyzed, with a focus on slack values for all paths. The absence of negative slack across all constraints confirmed that the design met its timing requirements with margin, ensuring safe and reliable operation under real-time conditions.







**CONCLUSION:-**

The digital design of the Health Monitoring System was successfully developed and verified using the complete RTL design and simulation flow. The project began with RTL coding in Verilog, focusing on the accurate implementation of medical formulas to compute Body Mass Index (BMI), Body Fat Percentage (BFP), Relative Fat Mass (RFM), Body Mineral Density (BMD), and Basal Metabolic Rate (BMR). Functional simulation ensured the correctness of the computation logic and the system’s ability to handle corner cases, such as division by zero and value overflow.

The design was synthesized to ensure it met area and performance constraints suitable for FPGA or ASIC implementation. Design robustness was further enhanced by limiting output values to 8-bit ranges and maintaining clock-driven sequential operation, supporting real-time responsiveness.

Although physical design steps like layout and fabrication were not included in the scope of this project, the RTL module is well-structured and ready for integration into larger systems such as fitness wearables, embedded health devices, or FPGA-based health trackers.

Overall, this project demonstrates a practical application of digital system design in the biomedical domain, transforming complex health-related computations into efficient hardware logic. The result is a reliable, scalable module capable of supporting personal health monitoring applications with minimal power and area overhead.