Design and Implementation of RISC Processor

Submitted as Third Year Mini-Project 2B

by

[Vansh Dhoka (5), Atharva Godkar(31), Anish Godse (32) , Sukanya Pingle(47)]

Supervisor : Dr. Mrs. Jaymala Adsul

Department of Electronics Engineering

V.E.S. Institute of Technology

An Autonomous Institute affiliated to University of Mumbai

2023-24

CERTIFICATE

This is to certify that the project entitled **"Design and Implementation of RISC Processor"** is a bonafide work of "**Vansh Dhoka(5) , Atharva Godkar(31) , Anish Godse(32) , Sukanya Pingle(47)** submitted to the V.E.S. Institute of Technology as a Third Year Mini Project 2B during the academic 2023-24.

(Name and sign)
Supervisor/Guide

(Name and sign)
Head of Department

(Name and sign)
Principal

Project Report Approval

This project report entitled (***Design and Implementation of RISC Processor***) by (**Vansh Dhoka(5) , Atharva Godkar(31) , Anish Godse(32) , Sukanya Pingle(47)**) is approved as **Third Year Mini project 2B** during Academic year 2023-24.

Examiners

1.----------

2.----------

Declaration

We declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, We have adequately cited and referenced the original sources. We also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Signature)

(Names of students and Roll No.)

(Names of students and Roll No.)

(Names of students and Roll No.)

(Names of students and Roll No.)

Date:

INDEX

CHAPTER 1
INTRODUCTION

CHAPTER 1 INTRODUCTION

- A processor receives instructions from the computer's memory, interprets them, and performs the necessary operations to execute those instructions. They are commonly found in personal computers, servers, laptops, smartphones, tablets, and various embedded systems. The aim of our project is to make a RISC processor.

- Our RISC processor will be able to execute 23 different types of instructions. ALU uniquely identifies these instructions using opcode. Size of opcode is 6 Bits. 2^6 = 64 this is more than sufficient for executing 16 instructions.

- Any instruction cannot be given to the processor in raw format. It is always converted into binary form and stored in program memory. This process is called encoding. The processor then fetches these instructions one by one and executes them. This process is called decoding.

- The size of all our instructions will be 32 Bits. It will contain opcode, source registers, destination register & constant value.

- Each instruction will require 5 clock cycles to complete. “STORE” is an exception which requires only 4 clock cycles. The execution of instruction is broken down into 5 stages i.e FETCH, DECODE, EXECUTE, MEMORY, WRITE-BACK. One clock cycle corresponding to each stage.

Architecture of RISC Processor:

- The ALU performs mathematical and logical operations, such as addition, subtraction, multiplication, and comparison.
- Registers are small, high-speed storage units within the processor that hold temporary data and instructions during processing.
- ROM (program memory) stores all the instructions which are to be executed. The processed data is finally stored in RAM (data memory). Harvard based architecture, separate ROM and RAM.
- The Program Counter keeps track of the memory address of the next instruction to be fetched. After each instruction is fetched, the PC is incremented to point to the next instruction in memory.

CHAPTER 2
LITERATURE REVIEW

CHAPTER 2 LITERATURE REVIEW

1. Design A 1-Bit Low Power Full Adder Using Cadence Tool

Kavita Khare , Krishna Dayal Shukla, MANIT/ Electronics & Communication, Bhopal, India

This paper presents a novel low-power majority function-based 1-bit full adder that uses MOS capacitors in its structure. In the project,the circuits simulated using CADENCE tool 0.18µm CMOS process technology.. The 1-bit full adder cell has 28 transistors. Different logic styles can be investigated from different points of view.. Evidently, they tend to favour one performance aspect at the expense of others.For example, static approach presents robustness against noise effects, so automatically provides a reliable operation.

2.DESIGN OF LOW POWER MULTIPLIER USING CADENCE TOOL J.Jayakumar

International Journal For Technological Research In Engineering Volume 3, Issue 7, March-2016

This paper shows a low power and High Speed bypassing multiplier. The multiplier embraces ripple-carry adder with less extra equipment components. In addition, the bypassing architecture can improve operating speed by the extra parallel architecture to abbreviate the delay time of the multiplier.Simulation results demonstrate that the proposed configuration can achieve more power efficiency with less additional equipment and power delay product among various counterparts

3.ASIC Design Flow And Methodology, Ashish A Shetty RV College Of Engineering-Bengaluru-59.

SSRG International Journal of Electrical and Electronics Engineering (SSRG - IJEEE) - Volume 6 Issue 7 - July 2019

In this paper, the ASIC specification and RTL Coding,Linting, Top level Integration and Verification,Synthesis , Logic Equivalence Check , these steps are performed.ASICs are complex.In general, the ASIC design happens at the following stages. 1. RTL Design. 2. Logic Synthesis. 3. Physical Design.Synthesis converts HDL description or RTL to gate-level implementation using standard gates, cell library, and memory blocks. It contains translation, optimization, and mapping. The translation is the conversion of the HDL format code to gate level Boolean equation

4.Design Procedure for Digital and Analog ICs using Cadence Tools Ganesh.R

CVR Journal of Science and Technology, Volume 9, December 2015

In the paper, the VLSI, Cadence, EDA Tools, IC Design, System on Chip methods can be understood. The selection of hardware or software methodology is based on non-functional constraints like maximum operating frequency, area, power etc .The VLSI Design is divided into two major groups as programmable design and non-programmable designs. This paper mainly concentrates on design flows and procedures of non-programmable IC designs using Cadence EDA tools. The introduction to Cadence tool flow for semi-custom and full custom is explained.

5.Design A 1 Bit Low Power Full Adder Using Cadence Tool

Kavita Khare and Krishna Shukla

This paper presents a novel low-power majority function-based 1-bit full adder that uses MOS capacitors in its structure. The power consumption and general characteristics of an adder are then compared against low power adders, the transmission function adder (TFA) and the conventional CMOS full adder. The circuits simulated using CADENCE tool 0.18 µm CMOS process technology.

CHAPTER 3
METHODOLOGY

CHAPTER 3 METHODOLOGY

The VLSI design flow is a complex and intricate process that transforms a concept into a physical silicon chip. It involves multiple stages, including specification and architecture design, RTL design and verification, logic synthesis, physical design, DFT, timing analysis, manufacturing, testing, and packaging.

In semester 5, we focused on the front end section. The steps involved are:

1. Creating a block diagram & Creating Verilog modules
2. Writing Verilog code for the data path & a test bench to verify it
3. Writing Verilog code for the control path & a test bench to verify it. Joining the data path & control path.
4. Converting single-cycle processor to multi-cycle processor by introducing the pipelining technique.
5. Solving the problems related to pipelining i.e data hazards, control hazards & structural hazard
6. Interfacing LCD screen & keyboard to FPGA board.

In 6th semester , we focussed on the back end part.
Physical design in VLSI (Very Large Scale Integration) refers to the process of transforming a logical circuit description into a physical layout that can be fabricated onto a semiconductor substrate, typically silicon. It involves converting the logical design, which describes the functionality of the circuit, into a geometric representation that specifies the exact locations and interconnections of physical components on the chip.

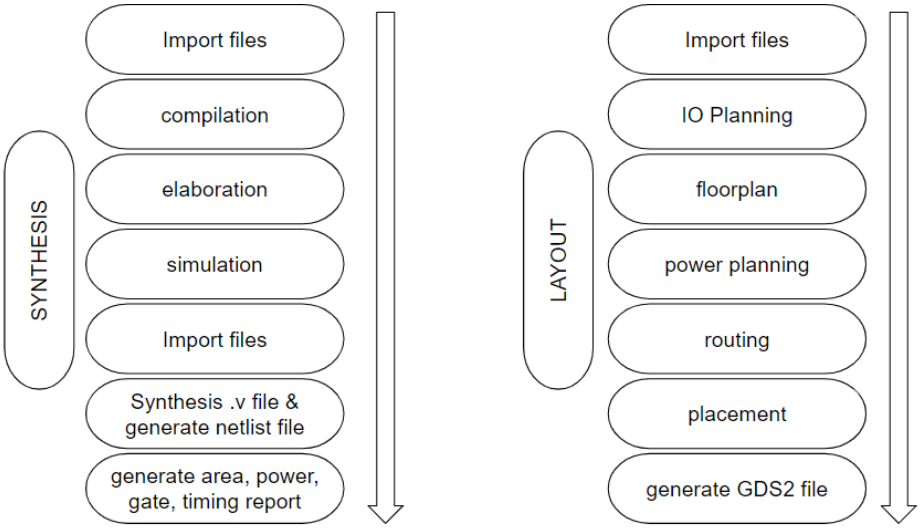

FIGURE 3.1

CHAPTER 4
BLOCK DIAGRAM & WORKING

CHAPTER 4 BLOCK DIAGRAM & WORKING

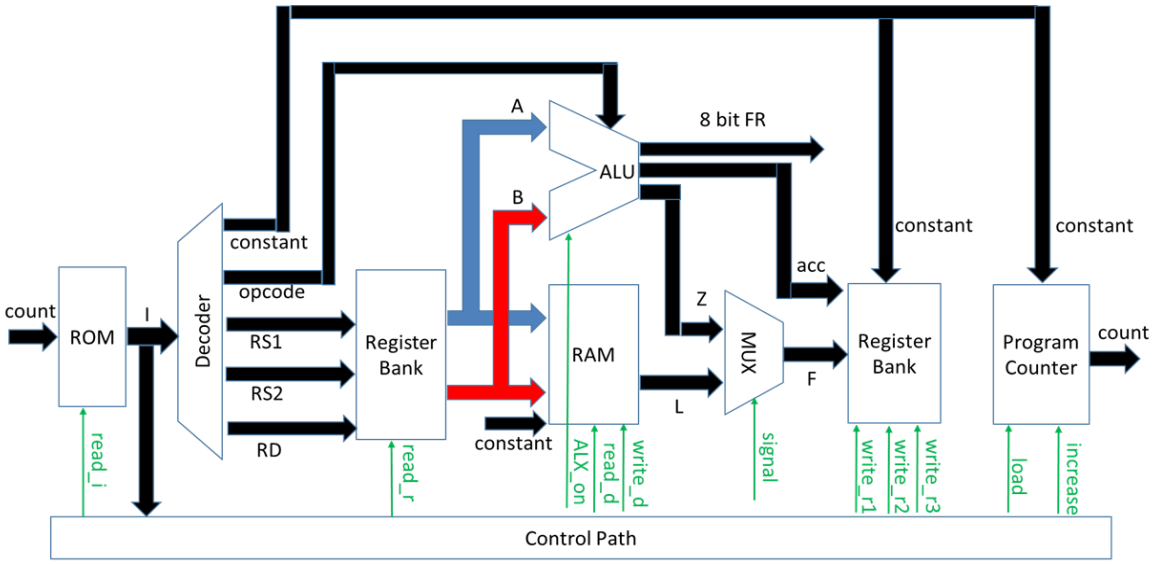

FIGURE 4.1

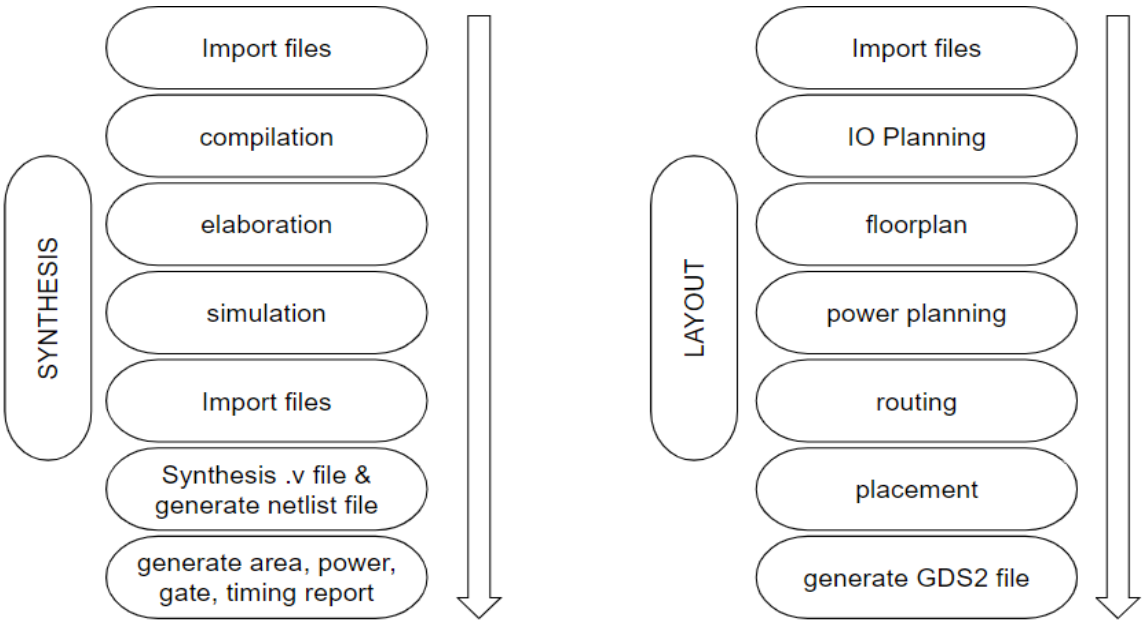

FIGURE 4.2

CHAPTER 4 BLOCK DIAGRAM & WORKING

Opcode	Format					Operation
000001	ADD	R1	R2	R3		R3 = R1 + R2
000010	SUB	R1	R2	R3		R3 = R1 – R2
000011	AND	R1	R2	R3		R3 = R1 & R2
000100	OR	R1	R2	R3		R3 = R1 | R2
000101	LT	R1	R2			For R1<R3 flag3 = 1, for R1>R3 flag3 = 0
000110	GT	R1	R2			For R1<R3 flag3 = 1, for R1>R3 flag3 = 0
000111	LOAD	K		R3	I.V.	R3 = mem [I.V. + K]
001000	STORE	K	R2		I.V.	mem [I.V. + K] = R2
001001	MOV			R3	57	R3 = 57
001010	RR	R1		R3		R3 = R1 rotated right by 4 bits
001011	RL	R1		R3		R3 = R1 rotated left by 4 bits
001100	JZ	R1			10	PC = 10 (if R1 == 0)
001101	JNZ	R1			10	PC = 10 (if R1 != 0)
001110	JE	R1	R2		10	PC = 10 (if R1 == R2)
001111	JNE	R1	R2		10	PC = 10 (if R1 != R2)
010000	JMP				10	PC = 10
010001	MUL	R1	R2	R3		R3 = R1 * R2
010010	DIV	R1	R2	R3		R3 = R1 % R2
010011	MOD	R1	R2	R3		R3 = R1 / R2
010100	NOT	R1		R3		R3 = ~R1
010101	MOVR	R1		R3		R3 = R1
010110	JFZ				10	PC = 10 (if flag4 = 0)
010111	JFNZ				10	PC = 10 (if flag4 = 1)

FIGURE 4.3

CHAPTER 4 BLOCK DIAGRAM & WORKING

Steps for simulation:

- Pwd
- Gedit source.v
- Gedit testbench.v
- Csh
- Source /home/install/cshrc
- Nclaunch -new&
- ======== **window opens** =======
- **Multiple steps**
- **Create cds.lib file**
- **Save**
- **Don't include any libraries**
- **Ok**
- ====== **new window opens** ======
- Click both verilog files on lhs
- Compile them using ncvlog
- Click both compiled files on rhs
- Elaborate them using ncelab
- Only click elaborated testbench
- Simulate it using ncsim
- ====== **new window opens** ======
- Click both files on lhs
- **Send selected object to target waveform window**
- ====== **new window opens** ======
- Select all IO pins on LHS
- Press the run simulation button

CHAPTER 4 BLOCK DIAGRAM & WORKING

Steps for Synthesis:

- pwd
- csh
- source /home/install/cshrc
- genus
- read_lib slow.lib
- read_hdl filename.v
- elaborate topmodule_name
- read_sdc filename.sdc
- syn_gen
- syn_map
- syn_opt
- report area
- report power
- report gates
- report timing -unconstrained
- write_hdl < filename_schematic.v
- write_sdc < filename_sdc.sdc
- gui_show
- Gui_hide

Steps for Layout:

- Pwd
- Csh
- Source /home/install/cshrc
- Innovus
- ======= **innovus tool opens** =======
- File > Import design
- ====== **window opens** ========
- Netlist = project.v
- Lef file = macro.lef & tech.lef
- Power net = VDD
- Ground net = VSS
- Mmmc = project.view
- Save everything with name project.global
- Press ok
- Floorplan > specify floorplan
- Power > global net connection
- Power > power planning > add rings
- Power > power planning > add strips
- Route > special route
- Place > place standard cell
- File >save > GDS/OASIS

CHAPTER 4 BLOCK DIAGRAM & WORKING

How to write .sdc file

A Constraint file is popularly known as an SDC(Synopsys Design Constraints) file by its extension of the file. It contains basically:
Units (Time, Capacitance, Resistance, Voltage, Current, Power), System interface (Driving cell, load), Design rule constraints (max fanout, max transition), Timing constraints (Clock definitions, clock latency, clock uncertainty, input/output delay), Timing exceptions (Multi-cycle and false paths)

- Rise Time/Fall Time: Refers to the time taken by a signal to transition from a low to a high state (rise time) or from a high to a low state (fall time).
- Skew: Represents the variation in arrival times of the same signal at different destinations. Minimizing skew ensures synchronous operation within the design.
- Jitter: The variation in the timing of a signal's edges caused by noise or other factors. Jitter constraints are vital for maintaining signal integrity.
- Latency: The time delay experienced by a signal as it propagates through various components of the design. Latency constraints are essential, especially in high-speed designs.
- Uncertainty: Refers to the variation or uncertainty in the timing analysis due to factors such as process variation, environmental conditions, etc. Constraining uncertainty ensures reliable timing performance across different conditions.+

Syntax of SDC File:

- **create_clock:** Defines the clock signal(s) used in the design.
- **derive_clocks:** Infers additional clocks based on specified relationships.
- **set_input_delay / set_output_delay:** Sets input/output delay constraints.
- **set_max_delay / set_min_delay:** Specifies maximum/minimum delay constraints.
- **set_false_path:** Defines paths that are not to be analyzed for timing.
- **set_multicycle_path:** Specifies multi-cycle paths.
- **set_clock_groups:** Groups clocks to constrain timing relationships.
- **set_case_analysis:** Defines constraints for case analysis paths.

Sample .sdc file

- create_clock -name **clk** -period **10** -waveform {**0 5**} [get_ports "**clk**"]
- set_clock_transition -rise **0.1** [get_clocks "**clk**"]
- set_clock_transition -fall **0.1** [get_clocks "**clk**"]
- set_clock_uncertainty -min_rise **0.12** [get_clocks "**clk**"]
- set_clock_uncertainty -min_fall **0.12** [get_clocks "**clk**"]
- set_clock_latency **2.35** [get_pins "**pin**"]
- create_generated_clock -multiply_by **3** -source **clk** [get_pins "**pin**"]
- create_generated_clock -divide_by **5** -source **clk** [get_pins "**pin**"]
- create_generated_clock –duty_cyle **25** -source **clk** [get_pins "**pin**"]
- create_generated_clock -invert -source **clk** [get_pins "**pin**"]
- set_input_delay -clock **clk** –max_rise **4** [get_ports "**port**"]
- set_input_delay -clock **clk** –min_rise **2** [get_ports "**port**"]
- set_input_delay -clock **clk** –max_fall **4** [get_ports "**port**"]
- set_input_delay -clock **clk** –min_fall **2** [get_ports "**port**"]
- set_output_delay -clock **clk** –max_rise **5** [get_ports "**port**"]
- set_output_delay -clock **clk** –min_rise **7** [get_ports "**port**"]
- set_output_delay -clock **clk** –max_fall **5** [get_ports "**port**"]
- set_output_delay -clock **clk** –min_fall **7** [get_ports "**port**"]

How to write .io file

```
(globals
version = 3
io_order = clockwise
space = 10 #1185-500-9*65 = 100/10=10
total_edge = 11
)

(iopad

(topleft (inst name="CornerCell1" cell-pfrelr offset=0 orientation=R180 place_status=fixed)
(left
(inst name="AO" cell-pc3b03ed place_status=fixed)          # pin no 1
(inst name="A1" cell-pc3b03ed place_status=fixed)          # pin no 2
(inst name="A2" cell-pc3b03ed place_status=fixed)          # pin no 3
(inst name="A3" cell-pc3b03ed place_status=fixed)          # pin no 4
(inst name="A4" cell-pc3b03ed place_status=fixed)          # pin no 5
(inst name="AS" cell-pc3b03ed place_status=fixed)          # pin no 6
(inst name="A6" cell-pc3b03ed place_status=fixed)          # pin no 7
(inst name="A7"cell-pc3b03ed place_status=fixed)           # pin no 8
(inst name="POWER_VDD1" cell-pvdi place_status=fixed)      # pin no 9
```

```
(topright (inst name="CornerCell2" cell-pfrelr offset=0 orientation=R90 place_status=fixed))
(top
(inst name="BO" cell-pc3b03ed place_status=fixed)          # pin no 10
(inst name="B1" cell-pc3b03ed place_status=fixed)          # pin no 11
(inst name="B2" cell-pc3b03ed place_status=fixed)          # pin no 12
(inst name="B3" cell-pc3b03ed place_status=fixed)          # pin no 13
(inst name="B4" cell-pc3b03ed place_status=fixed)          # pin no 14
(inst name="B5" cell-pc3b03ed place_status=fixed)          # pin no 15
(inst name="B6" cell-pc3b03ed place_status=fixed)          # pin no 16
(inst name="B7" cell-pv0i place_status=fixed)              # pin no 17
(inst name="POWER_VSS2" cell-pc3b03ed place_status=fixed)   # pin no 18
)
(bottomright (inst name="CornerCell3" cell-pfrelr offset=0 orientation=RO place_status=fixed))
(right
(inst name="out0" cell-pc3b03ed place_status=fixed)     # pin no 27
(inst name="out1" cell-pc3b03ed place_status=fixed)     # pin no 26
(inst name="out2" cell-pc3b03ed place_status=fixed)     # pin no 25
(inst name="out3" cell-pc3b03ed place_status=fixed)     # pin no 24
(inst name="out4" cell-pc3b03ed place_status=fixed)     # pin no 23
(inst name="out5" cell-pc3b03ed place_status=fixed)     # pin no 22
(inst name="out6" cell-pc3b03ed place_status=fixed)     # pin no 21
(inst name="out7" cell-pc3b03ed place_status=fixed)     # pin no 20
(inst name="POWER_VDD01" cell-pvda place_status=fixed)  # pin no 19
)

(bottomleft (inst name="CornerCell4" cell-pfrelr offset=0 orientation=R270 place_status=fixed))
(bottom
(inst name="SO" cell-pc3b03ed place_status=fixed)          # pin no 36
(inst name="S1" cell-pc3b03ed place_status=fixed)          # pin no 35
(inst name="S2" cell-pc3b03ed place_status=fixed)          # pin no 34
(inst name="S3" cell-pc3b03ed place_status=fixed)          # pin no 33
(inst name="clock_in" cell-pc3b03ed place_status=fixed)    # pin no 32
(inst name="c_outo" cell-pc3b03ed place_status=fixed)      # pin no 31
(inst name="extra_out1" cell-pc3b03ed place_status=fixed)  # pin no 30
(inst name="extra_out2" cell-pc3b03ed place_status-fixed)  # pin no 29
(inst name="POWER_VSSO2" cell-pv0a place_status=fixed)     # pin no 28
)
```

CHAPTER 4 BLOCK DIAGRAM & WORKING

Design Import:

- As soon as the innovus tool is launched, the first step is to load all the necessary files into the innovus tools. This includes the netlist file and sdc file generated by genus tool, liberty files, io placement file & library exchange format files.

- From the innovus window click file > import design. A new window opens up. Select the netlist file from your working directory. Top cell > By User. mention the name of top module who's layout is to be made.

- Next step is to upload the .lef files. First we upload tech.lef file and then we upload the macro.lef file. The order in which files are uploaded is very crucial.

- Next we name the power net > VDD and ground net > VSS. these 2 nets will be used in power planning, floor planning, routing etc.

- Next step is to add .view file. Click on " create analysis configuration". A new window will appear with name mmmc. U have to load liberty and sdc files in these window. After that "save & close". Tool automatically creates .view file for you.

- Finally save everything. Tool will create a .global file. Now press ok. A rectangle box will appear on the gui of innovus tool.

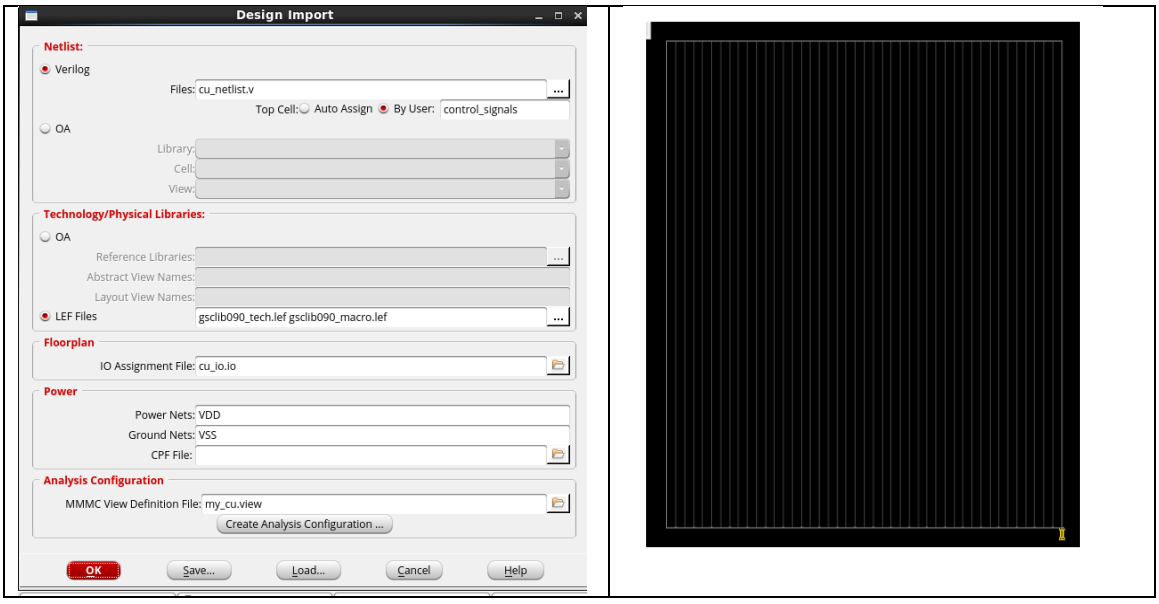

FIGURE 4.4

CHAPTER 4 BLOCK DIAGRAM & WORKING

Floor Planning:

- Floor Planning involves determining the location, shape, and size of modules in a way that one can avoid congestion. Floor Planning is a quintessential step which decides the layout of the VLSI design. A well-optimized floor planning allows an ASIC design that has higher performance.
- Floorplanning is an essential design step for hierarchical, building-module design methodology.
- Floorplanning provides early feedback that evaluates architectural decisions, estimates chip areas, and estimates delay and congestion caused by wiring.
- Floorplanning is the first major step in physical design; it is particularly important because the resulting floor plan affects all the subsequent steps in physical design, such as placement and routing.
- In addition to chip area minimization, modern VLSI floorplanning also needs to handle some important issues such as soft modules and fixed-outline constraints. Unlike a hard module that has a fixed dimension (width and height), the shape of a soft module is to be decided during floorplanning, although its area is fixed. Therefore, a floorplanner needs to find a desired aspect ratio for each soft module to optimize the floorplan cost.

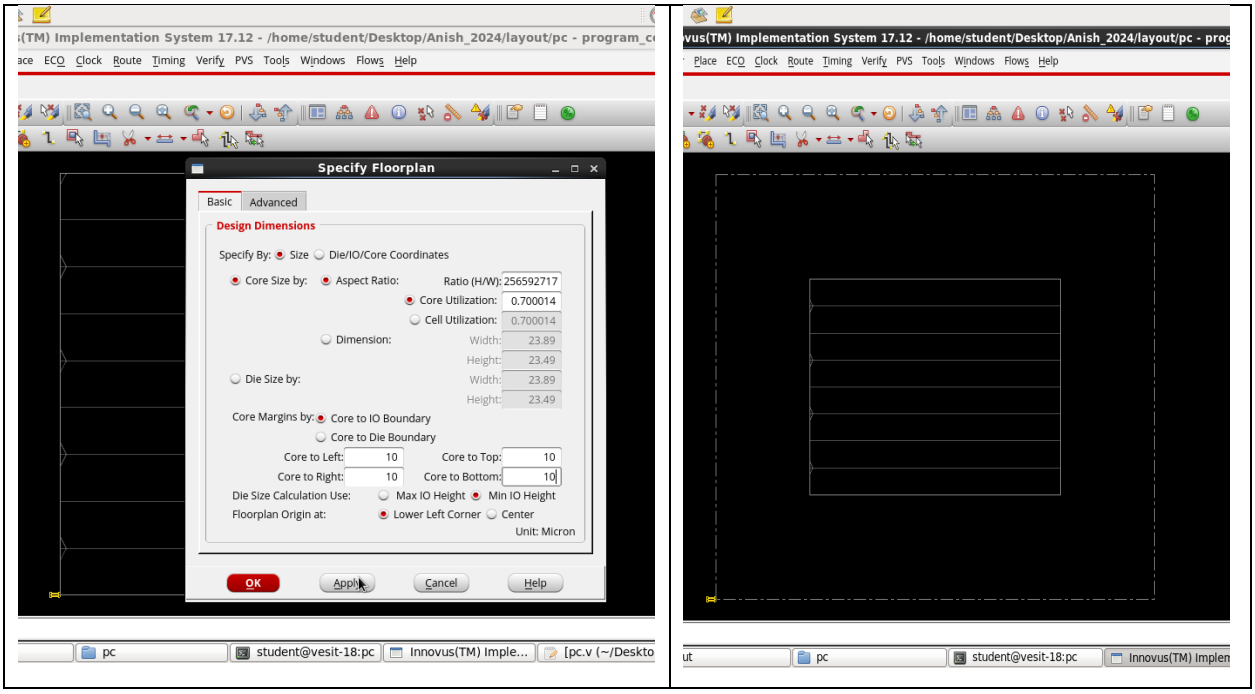

FIGURE 4.5

CHAPTER 4 BLOCK DIAGRAM & WORKING

Power Planning:

- Power planning ensures proper power distribution across the chip, minimizing voltage drop and noise.
- It involves strategically placing power supplies, power stripes, and decoupling capacitors to ensure stable power delivery.
- The main agenda of the power plan is to create a proper way to send the power from outside of the world to the standard cells and macros through the power routes.
- The levels of power distribution are:

Power Pads→Power Rings→Power Stripes→Power Rails→Standard Cells

- The power pads are placed along with the IO pads in the pads section of the full chip.These pads pass the power from outside of the chips to the power rings.
- The power rings serve as local power distribution networks, providing power to the circuits within the enclosed area. Power rings are especially useful for providing clean and stable power to sensitive or high-performance circuitry, as they help reduce noise and voltage fluctuations.
- The power stripes are the flies over the core area. The power stripes on both ends connect to the core ring.
- Power stripes are always created in the top metal layer. These stripes supply power from the core rings to the power rails.
- By strategically placing power stripes, designers ensure that power can be efficiently distributed to all areas of the chip, minimizing voltage drop and power distribution network (PDN) resistance.
- Thus , we can say that the components of power planning involves :

Power Rings: These carry both VDD and VSS around the chip.

Power Straps: They connect VDD and VSS from rings across the chip.

Power Rails: These connect VDD and VSS to the standard cells.

CHAPTER 4 BLOCK DIAGRAM & WORKING

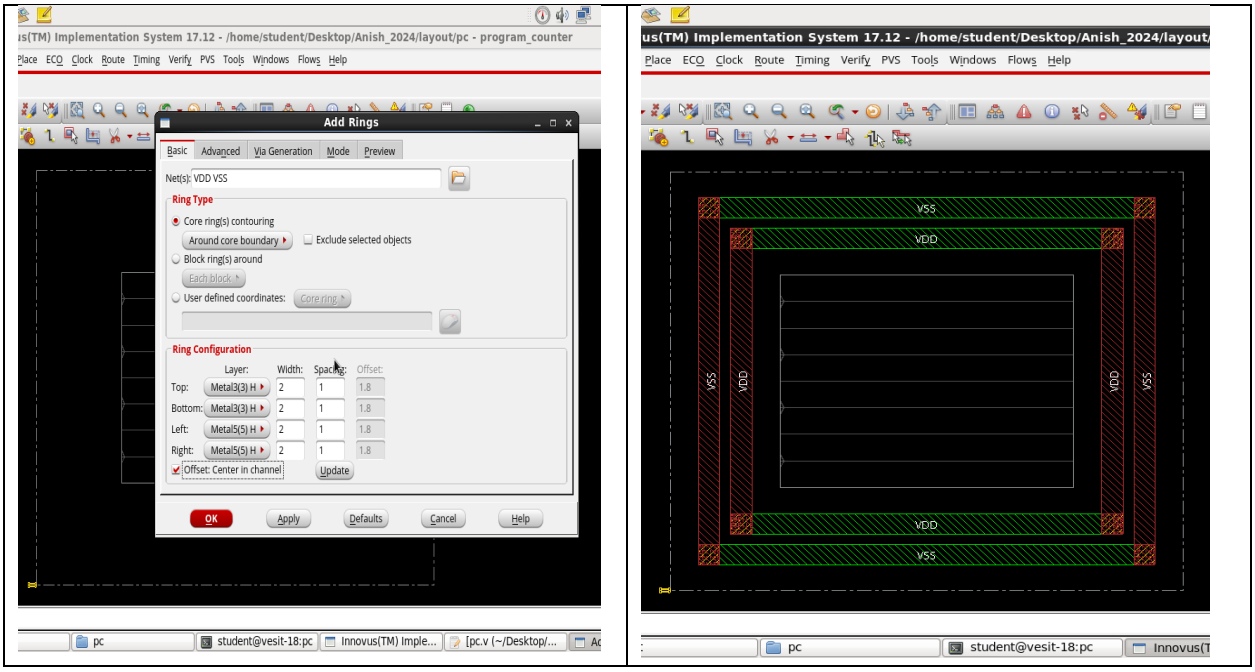

FIGURE 4.6

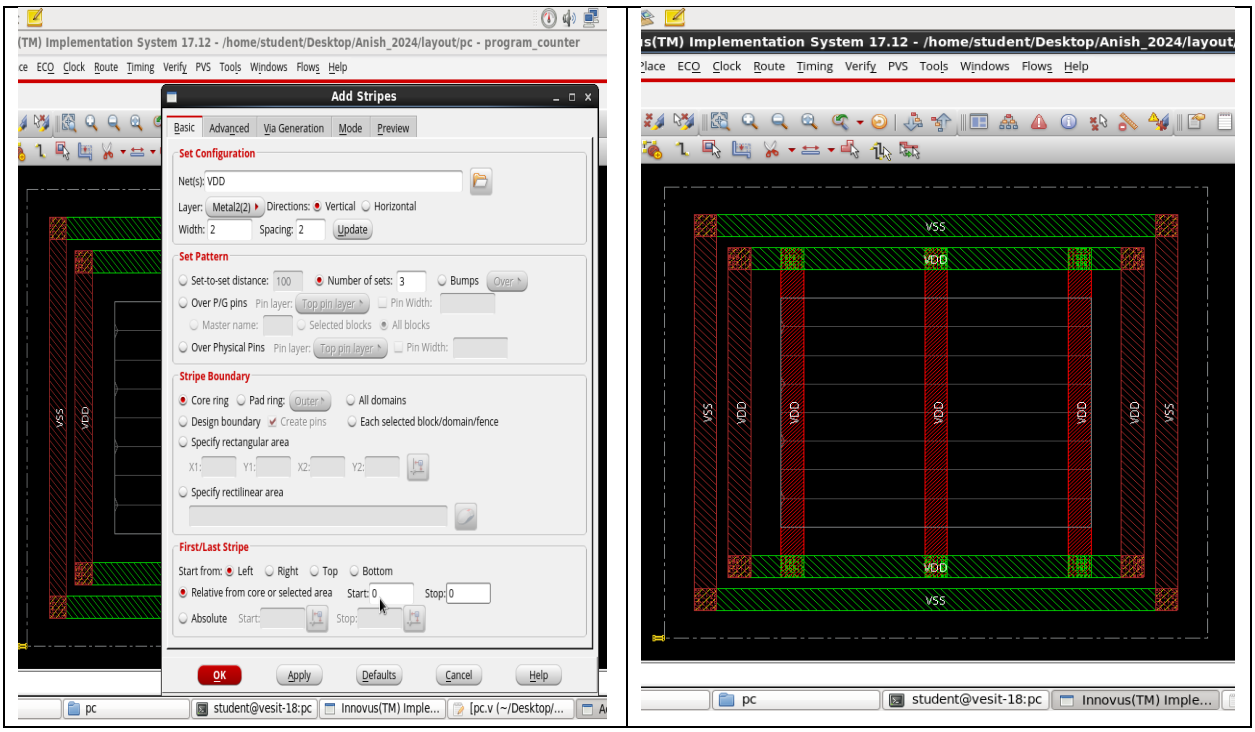

FIGURE 4.7

Routing

- Routing involves creating physical connections (metal traces) between the placed logic cells, power pads, and I/O pads. It aims to minimize wire length, congestion, and parasitic effects while meeting timing and signal integrity requirements.
- Route metal traces to connect logic cells while adhering to design rules and constraints. Minimize wire length and congestion to optimize performance and area.
- Ensure proper shielding and spacing to reduce noise and signal interference.
- It includes Defining routing constraints, such as minimum trace width, spacing, and maximum length.
- Uses automated or manual routing techniques to connect the components and IOs according to the design requirements.
- It Optimizes routing for signal integrity, power distribution, and manufacturability.
- The routing mechanism establishes the specific pathways for interconnection. This contains the regular cells and macro pins, block boundary pins and chip boundary pads.
- Each metal layer in a grid based routing system has its tracks and preferred routing directions which are described in a unified cell in the standard cell library.
- Routing activities are divided into 4 steps:
  - Global route
  - Track assignment
  - Detail routing
  - Search and repair

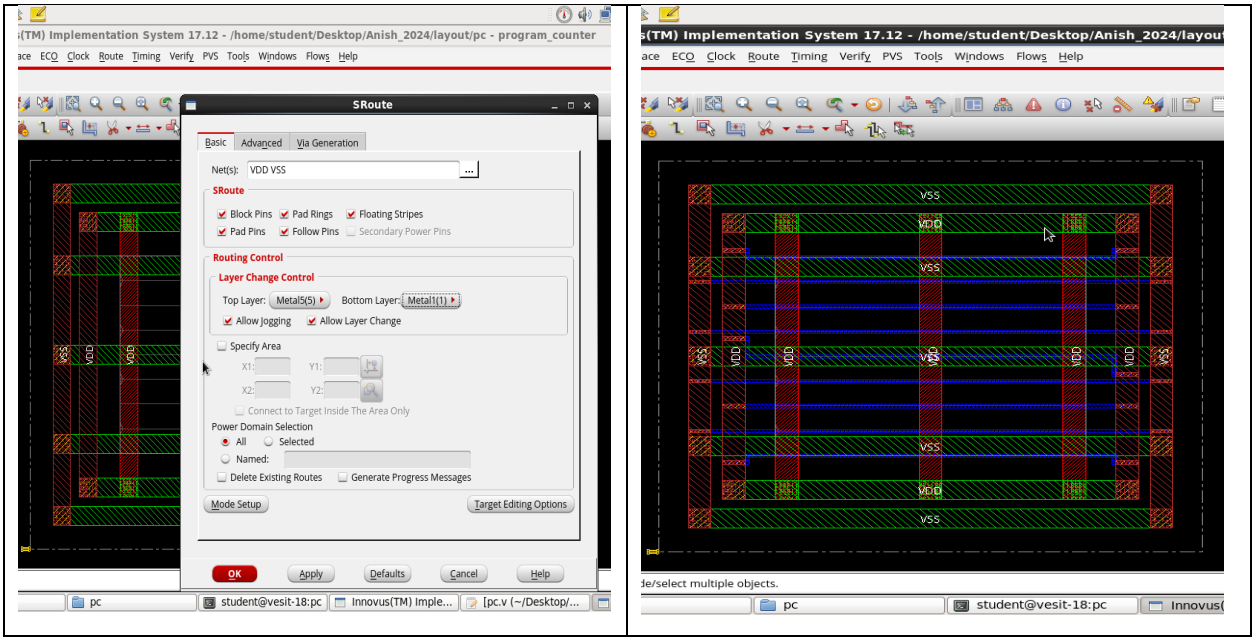

FIGURE 4.8

CHAPTER 4 BLOCK DIAGRAM & WORKING

Placement:

- Placement involves placing the synthesized logic cells onto the chip floorplan while considering timing, power, and area constraints.
- It aims to optimize performance, minimize wire length, and facilitate efficient routing.
- Place logic cells within allocated placement regions while adhering to design constraints.
- Place power supplies strategically to minimize voltage drops and ensure stable power distribution to all components.
- Placement is the process of placing the standard cells inside the core boundary in an optimal location. The tool tries to place the standard cell in such a way that the design should have minimal congestions and the best timing.

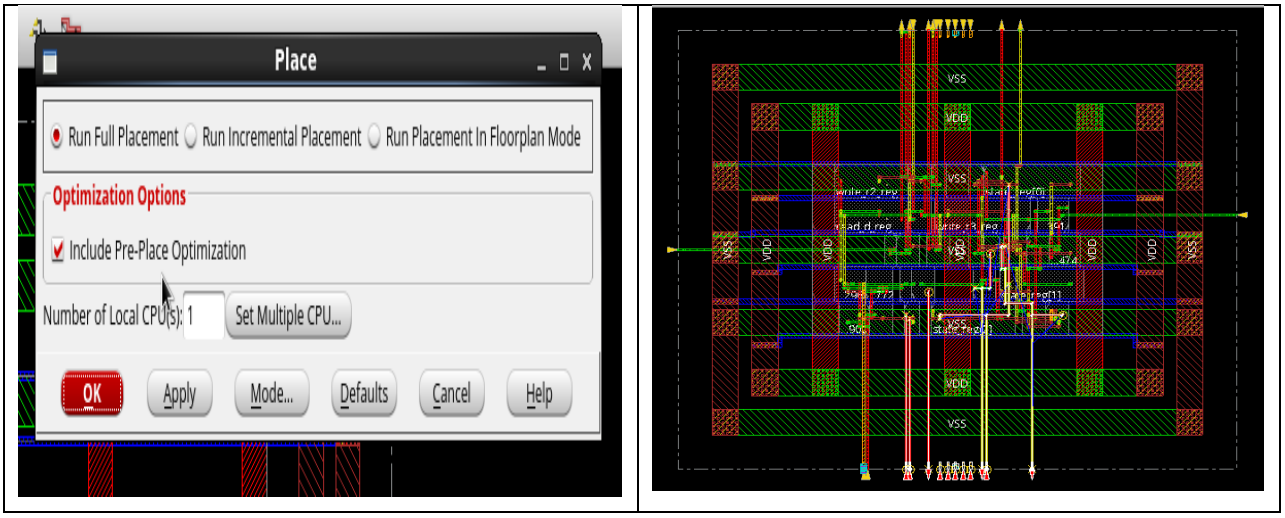

FIGURE 4.9

CHAPTER 4 BLOCK DIAGRAM & WORKING

IO Planning:

- It Identifies the input and output requirements of design, including signal types (analog, digital, high-speed), voltage levels, and signal integrity considerations.
- It forms group of I/O pins based on their functions and connectivity requirements.
- It determine the location of connectors, headers, and other external interfaces on PCB layout.
- It ensures proper signal routing and signal integrity by minimizing signal reflections, crosstalk, and impedance mismatches.

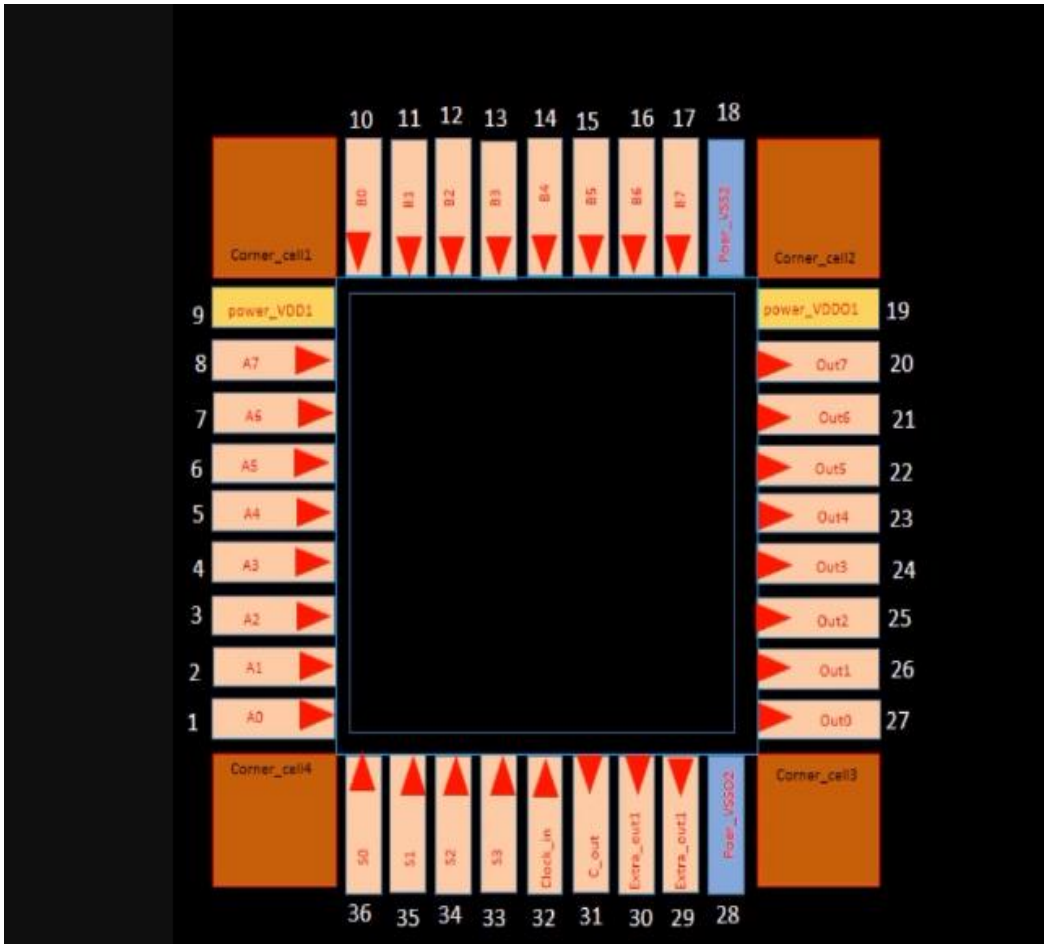

FIGURE 4.10

CHAPTER 5
HARDWARE & SOFTWARE

CHAPTER 5 HARDWARE & SOFTWARE

Cadence Design Systems, founded in 1988 by Dr. James D. Solomon, offers cutting-edge software, hardware, and intellectual property solutions for electronic design automation (EDA) in the semiconductor industry. With its headquarters in San Jose, California, Cadence has established itself as a global leader in electronic design innovation. The company provides a wide range of products and services including digital integrated circuit design, functional verification, custom IC design, PCB layout and design, and system design enablement. in this project we have used the following three tools of cadence -

ncsim tool -

NCsim is a powerful and widely-used Verilog and VHDL simulator. It is primarily employed for functional verification and simulation of digital circuits at the RTL and gate-level abstraction levels. NCsim provides engineers with a comprehensive environment for debugging, testing, and analyzing the behavior of their designs. With features such as advanced waveform viewing, assertion-based verification, and support for SystemVerilog constructs, NCsim facilitates efficient and accurate simulation of complex digital designs. Its robust capabilities make it a go-to solution for verifying the functionality and performance of digital circuits, ensuring designs meet specifications before moving on to the next stages of the design flow. During the entire simulation process total 3 tools are used i.e ncsim, ncelab, ncvlog whic are a part of Cadence Verification Suit. Order in which the above tools are used ncvlog > ncelab > ncsim. NCsim is for simulation, Ncelab is for elaboration and optimization, and Ncvlog is for compiling Verilog source files.

NCsim:

- NCsim is a simulation tool used for functional verification and simulation of digital circuits. It supports both Verilog and VHDL.
- NCsim allows engineers to simulate designs at various abstraction levels, including RTL (Register Transfer Level) and gate-level.
- It provides features for waveform viewing, debugging, and assertion-based verification.

Ncelab:

- Ncelab is a tool primarily used for elaboration and optimization of Verilog and VHDL designs.
- Elaboration is the process of expanding and analyzing design hierarchy to prepare it for simulation.
- Ncelab ensures that all design elements are properly connected and instantiated before simulation.

Ncvlog:

- Ncvlog is a Verilog compiler tool used for compiling Verilog source files into a design library.
- It analyzes Verilog code syntax and semantics, checking for errors and generating intermediate files for simulation.

genus tool -

Genus is a high-performance RTL synthesis tool developed by Cadence. It translates Register Transfer Level (RTL) code into a gate-level netlist, optimizing the design for area, power, and timing constraints while preserving functionality. It plays a crucial role in the digital design flow, helping designers achieve better quality of results (QoR) and faster time-to-market for their integrated circuit designs. Genus is known for its advanced optimization algorithms and support for the latest design methodologies, making it a key tool for semiconductor companies and design teams worldwide. genus requies 2 input files i.e verilog file and liberty file (slow.lib fast.lib). it then produces produces output

innovus tool -

The Innovus Implementation System provides new capabilities in placement, optimization, routing, and clocking. Its unique architecture accounts for upstream and downstream steps and effects in the design flow to minimize design iterations and provide a runtime boost.

The Innovus system offers mixed-macro and standard-cell placement, which enables macro locations to be automatically generated, reducing the time to create an optimal floorplan from days to hours. The latest advances in machine learning computer science are very relevant for digital implementation flows.

CHAPTER 6
RESULT & DISCUSSION

CHAPTER 6 RESULT & DISCUSSION

Decoder netlist

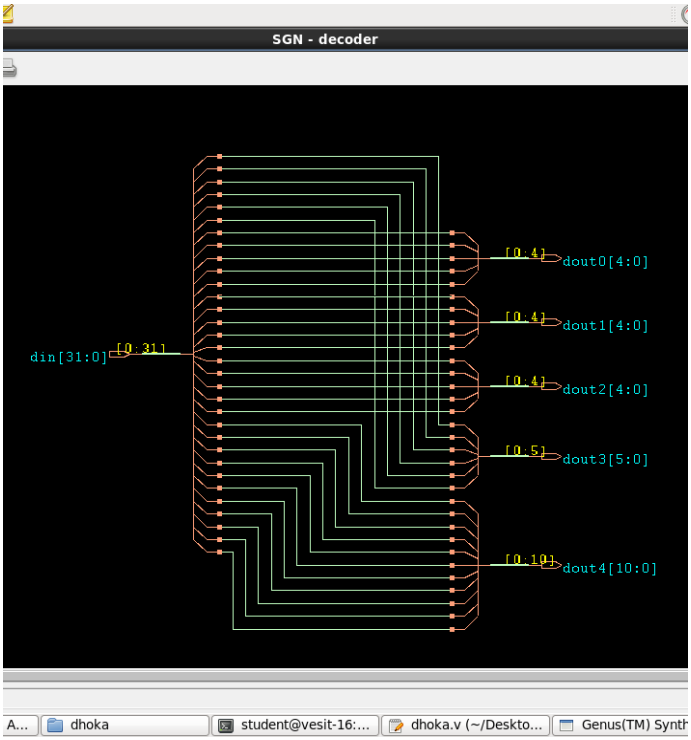

FIGURE 6.1

mux netlist

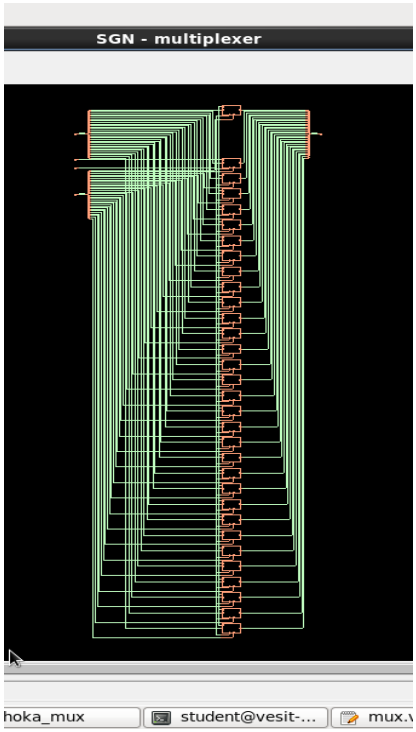

FIGURE 6.2

ALU netlist

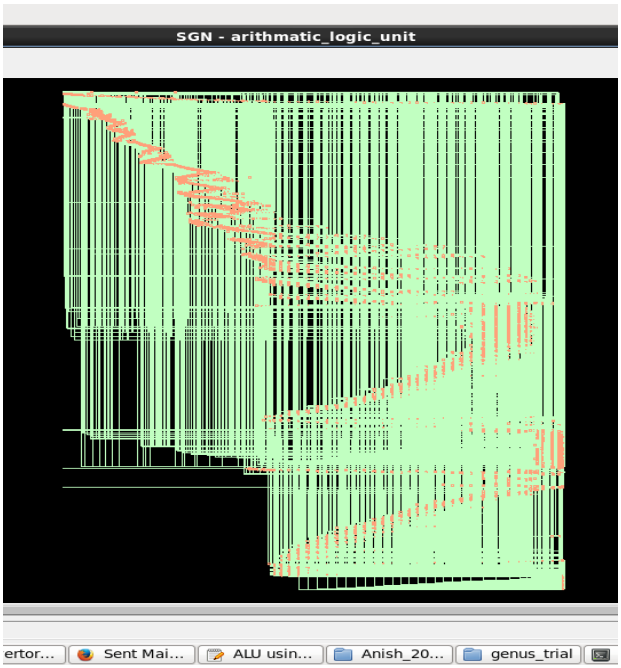

FIGURE 6.3

register bank netlist

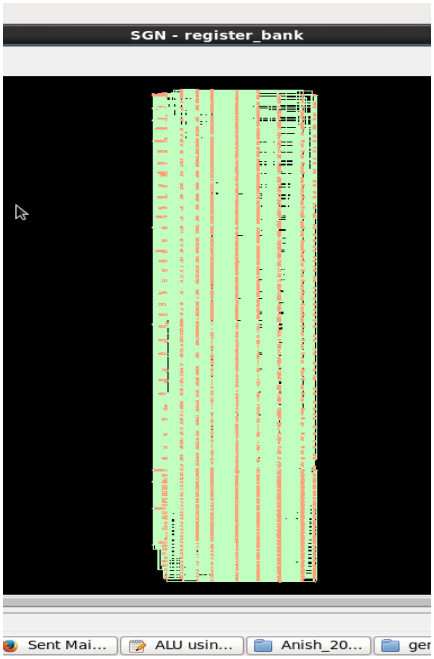

FIGURE 6.4

CHAPTER 6 RESULT & DISCUSSION

Control unit netlist

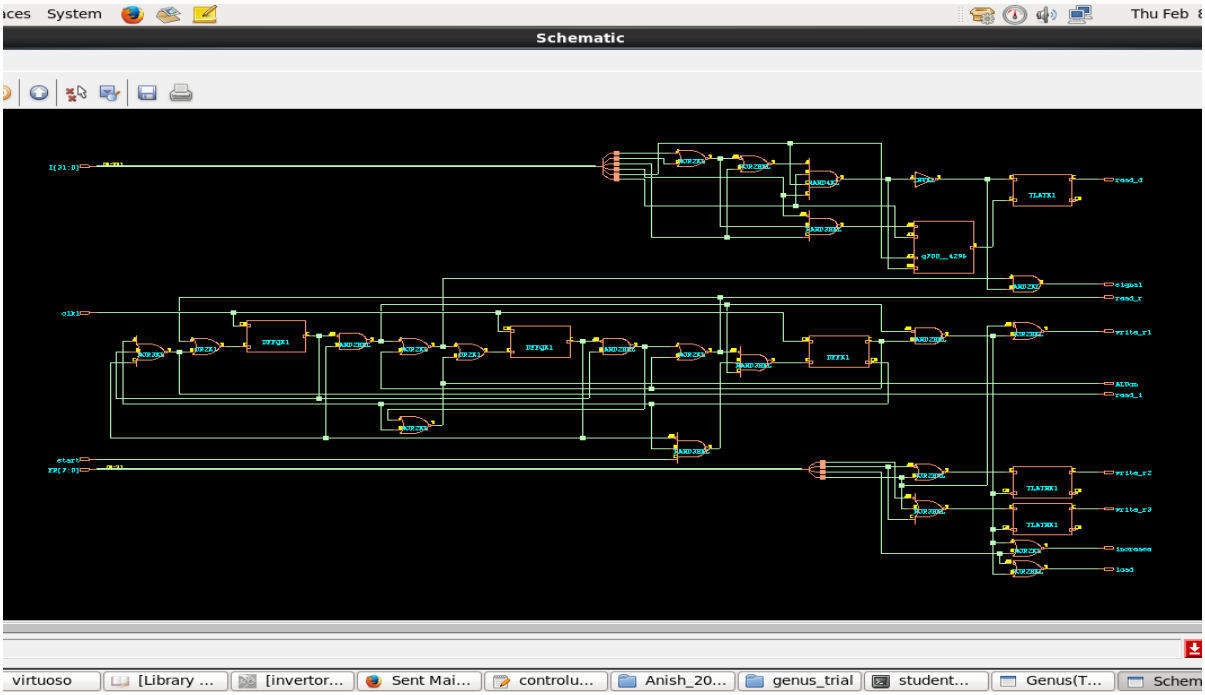

FIGURE 6.5

Program counter netlist

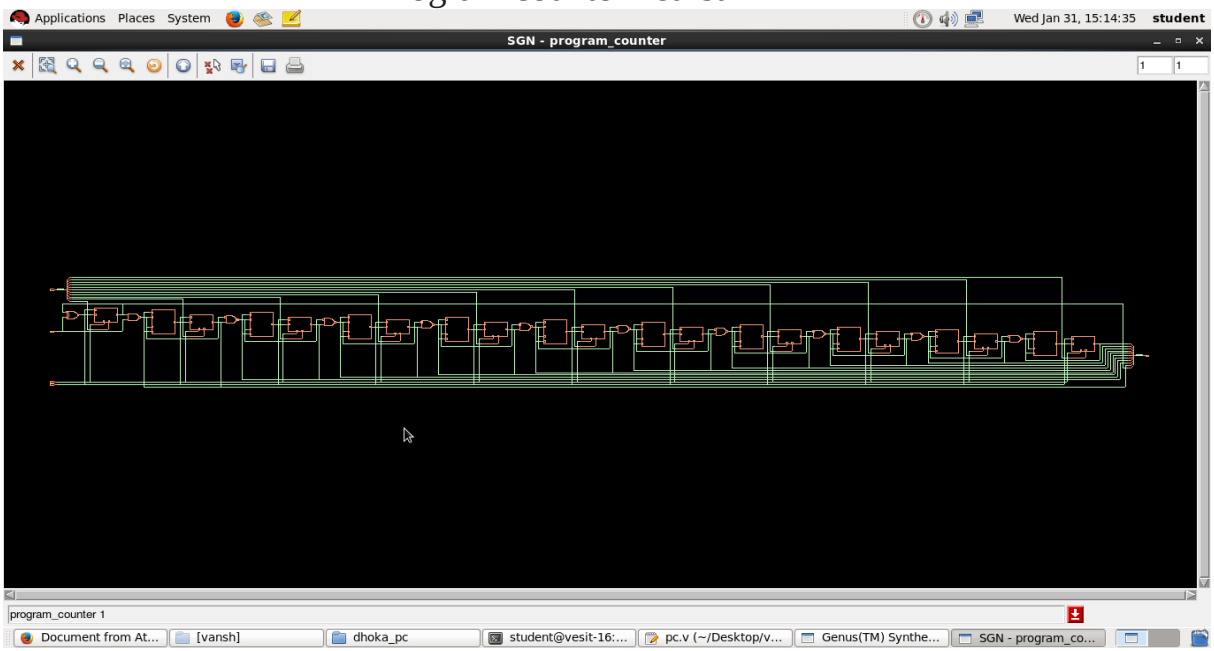

FIGURE 6.6

CHAPTER 6 RESULT & DISCUSSION

RISC processor netlist

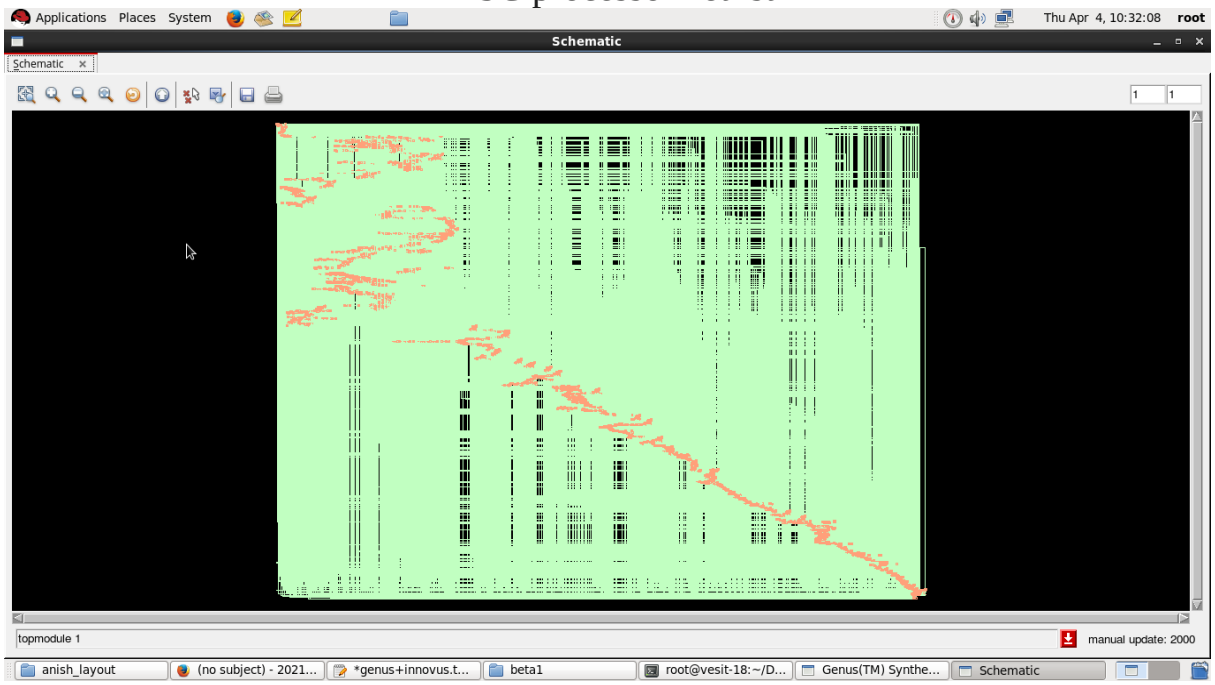

FIGURE 6.7

RISC processor layout

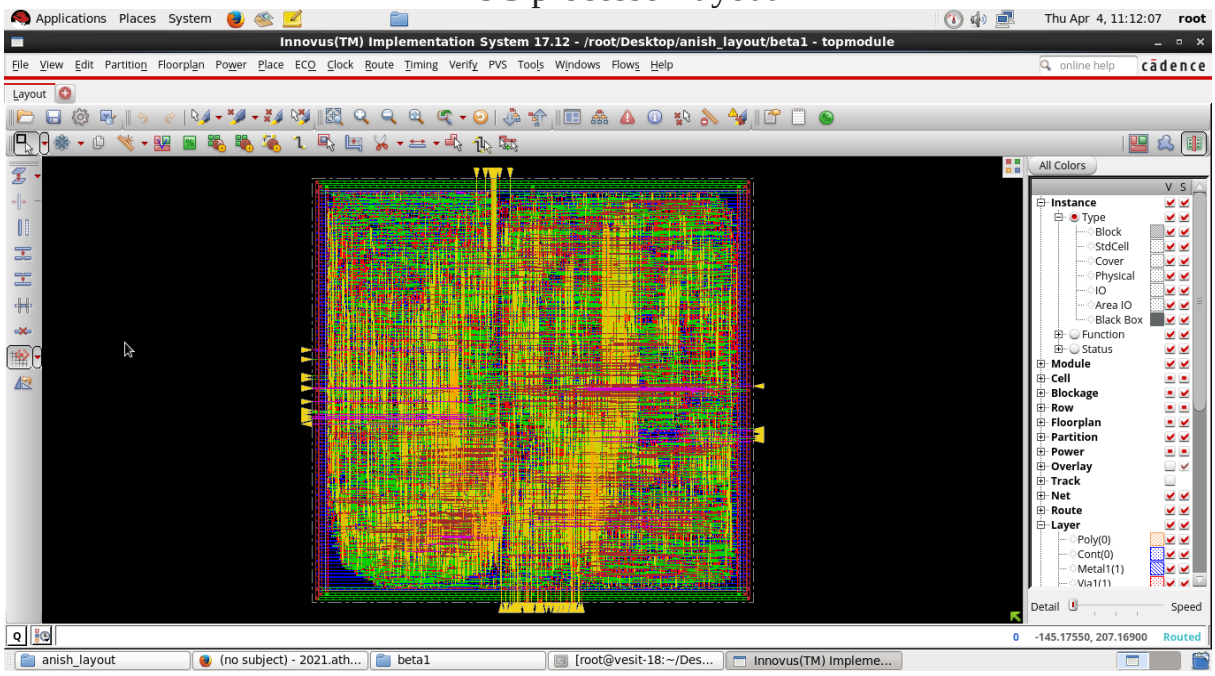

FIGURE 6.8

CHAPTER 6 RESULT & DISCUSSION

RISC area report

```
////////////////////////////////////////////////////////////////////////////

innovus 3> report_area
Depth  Name         #Inst  Area (um^2)
---------------------------------------
0      topmodule    9911   71514.9396
1

////////////////////////////////////////////////////////////////////////////

------------------------------------------------------------------------
```

ESP32-C6 area report

- For information about tape, reel, and chip marking, please refer to *Espressif Chip Packaging Information*.
- The pins of the chip are numbered in anti-clockwise order starting from Pin 1 in the top view. For pin numbers and pin names, see also Figure 2-1 ESP32-C6 Pin Layout (QFN40, Top View) and Figure 2-2 ESP32-C6 Pin Layout (QFN32, Top View).

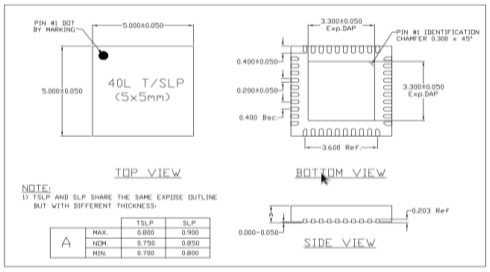

Figure 6-1. QFN40 (5×5 mm) Package

FIGURE 6.9

RISC power report

```
Total Power
--------------------------------------------------------------------------------
Total Internal Power:        9.59481050           67.8225%
Total Switching Power:       4.54484179           32.1260%
Total Leakage Power:         0.00729597            0.0516%
Total Power:  |             14.14694826
--------------------------------------------------------------------------------


Group                          Internal   Switching     Leakage       Total  Percentage
                               Power      Power         Power         Power  (%)
--------------------------------------------------------------------------------------
Sequential                         2.26      0.9057    0.002255       3.168        22.4
Macro                                 0           0           0           0           0
IO                                    0           0           0           0           0
Combinational                     7.335       3.639    0.005041       10.98        77.6
Clock (Combinational)                 0           0           0           0           0
Clock (Sequential)                    0           0           0           0           0
--------------------------------------------------------------------------------------
Total                             9.595       4.545    0.007296       14.15         100
--------------------------------------------------------------------------------
```

ESP32-C6 power report

4.1 Absolute Maximum Ratings

Stresses above those listed in Table 4-1 *Absolute Maximum Ratings* may cause permanent damage to the device. These are stress ratings only and normal operation of the device at these or any other conditions beyond those indicated in Section 4.2 Recommended Power Supply Characteristics is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

Table 4-1. Absolute Maximum Ratings

Parameter	Description	Min	Max	Unit
Input power pins[1]	Allowed input voltage	-0.3	3.6	V
T_{STORE}	Storage temperature	-40	150	°C

[1] For more information on input power pins, see Section 2.5.1 Power Pins.

[2] The product proved to be fully functional after all its IO pins were pulled high while being connected to ground for 24 consecutive hours at ambient temperature of 25 °C.

FIGURE 6.10

RISC timing report

```
------------------------------------------------------------
          timeDesign Summary
------------------------------------------------------------

Setup views included:
 worst_case

+--------------------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|          WNS (ns):|  0.000  |   N/A   |  0.000  |
|          TNS (ns):|  0.000  |   N/A   |  0.000  |
|    Violating Paths:|    0    |   N/A   |    0    |
|         All Paths:|    0    |   N/A   |    0    |
+--------------------+---------+---------+---------+

+----------------+-------------------------------+------------------+
|                |              Real             |       Total      |
|    DRVs        +------------------+------------+------------------+
|                |  Nr nets(terms)  | Worst Vio  |  Nr nets(terms)  |
+----------------+------------------+------------+------------------+
|   max_cap      |      0 (0)       |   0.000    |      0 (0)       |
|   max_tran     |      0 (0)       |   0.000    |      0 (0)       |
|   max_fanout   |      0 (0)       |     0      |      0 (0)       |
|   max_length   |      0 (0)       |     0      |      0 (0)       |
+----------------+------------------+------------+------------------+

Density: 69.646%
Routing Overflow: 0.00% H and 0.07% V
------------------------------------------------------------
Reported timing to dir timingReports
Total CPU time: 0.79 sec
Total Real time: 0.0 sec
Total Memory Usage: 1225.648438 Mbytes
```

FIGURE 6.11

area report before optimization	area report after optimization

```
innovus 1> report_area
Depth  Name         #Inst  Area (um^2)
---------------------------------------
0      topmodule    9830   71878.2516
1
```

```
innovus 3> report_area
Depth  Name         #Inst  Area (um^2)
---------------------------------------
0      topmodule    9911   71514.9396
1
```

FIGURE 6.12

power report before optimization	power report after optimization

```
Total Power
--------------------------------------------------------------------------------
Total Internal Power:        3.63035363           70.3074%
Total Switching Power:       1.52627332           29.5586%
Total Leakage Power:         0.00691556            0.1339%
Total Power:                 5.16354254
--------------------------------------------------------------------------------


Group                    Internal  Switching  Leakage  Total  Percentage
                         Power     Power      Power    Power  (%)
----------------------------------------------------------------------------
Sequential                  1.048     0.3771  0.002255  1.427       27.64
Macro                           0          0         0      0           0
IO                              0          0         0      0           0
Combinational               2.583     1.149  0.004661  3.736       72.36
Clock (Combinational)           0          0         0      0           0
Clock (Sequential)              0          0         0      0           0
----------------------------------------------------------------------------
Total                        3.63      1.526  0.006916  5.164         100
----------------------------------------------------------------------------
```

```
Total Power
--------------------------------------------------------------------------------
Total Internal Power:        9.59481050           67.8225%
Total Switching Power:       4.54484179           32.1260%
Total Leakage Power:         0.00729597            0.0516%
Total Power:                14.14694826
--------------------------------------------------------------------------------


Group                    Internal  Switching  Leakage  Total  Percentage
                         Power     Power      Power    Power  (%)
----------------------------------------------------------------------------
Sequential                   2.26    0.9057  0.002255  3.168       22.4
Macro                           0          0         0      0           0
IO                              0          0         0      0           0
Combinational                7.335     3.639  0.005041  10.98       77.6
Clock (Combinational)           0          0         0      0           0
Clock (Sequential)              0          0         0      0           0
----------------------------------------------------------------------------
Total                        9.595      4.545  0.007296  14.15         100
----------------------------------------------------------------------------
```

FIGURE 6.13

Timing report before optimization	Timing report after optimization

```
------------------------------------------------------------
          timeDesign Summary
------------------------------------------------------------

Setup views included:
 worst_case

+--------------------+--------+--------+---------+
|     Setup mode     |  all   | reg2reg | default |
+--------------------+--------+--------+---------+
|           WNS (ns):| 0.000  |  N/A   |  0.000  |
|           TNS (ns):| 0.000  |  N/A   |  0.000  |
|    Violating Paths:|   0    |  N/A   |    0    |
|         All Paths:|   0    |  N/A   |    0    |
+--------------------+--------+--------+---------+

+----------------+-------------------------------+------------------+
|                |             Real              |       Total      |
|    DRVs        +---------------+---------------+------------------|
|                | Nr nets(terms) | Worst Vio  | Nr nets(terms)  |
+----------------+---------------+---------------+------------------+
|   max_cap      |    159 (159)  |   -0.075   |    159 (159)     |
|   max_tran     |    107 (3589) |   -2.024   |    107 (3589)    |
|   max_fanout   |      0 (0)    |      0     |      0 (0)       |
|   max_length   |      0 (0)    |      0     |      0 (0)       |
+----------------+---------------+---------------+------------------+

Density: 69.999%
Routing Overflow: 0.00% H and 0.01% V
------------------------------------------------------------
Reported timing to dir timingReports
Total CPU time: 3.63 sec
Total Real time: 5.0 sec
Total Memory Usage: 1161.777344 Mbytes
```

```
------------------------------------------------------------
          timeDesign Summary
------------------------------------------------------------

Setup views included:
 worst_case

+--------------------+--------+--------+---------+
|     Setup mode     |  all   | reg2reg | default |
+--------------------+--------+--------+---------+
|           WNS (ns):| 0.000  |  N/A   |  0.000  |
|           TNS (ns):| 0.000  |  N/A   |  0.000  |
|    Violating Paths:|   0    |  N/A   |    0    |
|         All Paths:|   0    |  N/A   |    0    |
+--------------------+--------+--------+---------+

+----------------+-------------------------------+------------------+
|                |             Real              |       Total      |
|    DRVs        +---------------+---------------+------------------|
|                | Nr nets(terms) | Worst Vio  | Nr nets(terms)  |
+----------------+---------------+---------------+------------------+
|   max_cap      |      0 (0)    |   0.000    |      0 (0)       |
|   max_tran     |      0 (0)    |   0.000    |      0 (0)       |
|   max_fanout   |      0 (0)    |      0     |      0 (0)       |
|   max_length   |      0 (0)    |      0     |      0 (0)       |
+----------------+---------------+---------------+------------------+

Density: 69.646%
Routing Overflow: 0.00% H and 0.07% V
------------------------------------------------------------
Reported timing to dir timingReports
Total CPU time: 0.79 sec
Total Real time: 0.0 sec
Total Memory Usage: 1225.648438 Mbytes
```

FIGURE 6.14

CHAPTER 7
CONCLUSION & FUTURE SCOPE

CHAPTER 7 CONCLUSION & FUTURE SCOPE

In this project we completed all the steps of digital IC design flow successfully. Now the next task would be to generate GDS 2 file. A GDS II (Graphic Data System) file is a standard format used in the semiconductor industry for describing the geometry of electronic design data. It's essentially a layout file that contains information about the physical layout of the various components that make up an integrated circuit (IC). This includes details such as the positions and shapes of transistors, wires, contacts, and other elements.

Mask Generation: The GDS II file is used to generate masks, which are essentially templates used in the semiconductor manufacturing process. Each layer of the IC design corresponds to a separate mask. These masks are used to transfer the design onto silicon wafers during the fabrication process.

Photolithography: During the fabrication process, the masks are used in photolithography. This involves shining light through the mask onto a photosensitive material coating a silicon wafer. The pattern from the mask is transferred onto the wafer, creating the desired circuit layout on the wafer's surface.

Etching and Deposition: After the pattern is transferred onto the wafer, various processes such as etching and deposition are used to remove or add materials to create the actual circuit components according to the layout specified in the GDS II file.

Integration and Testing: Once all the layers have been processed, the individual chips are integrated into a larger wafer and tested for functionality.

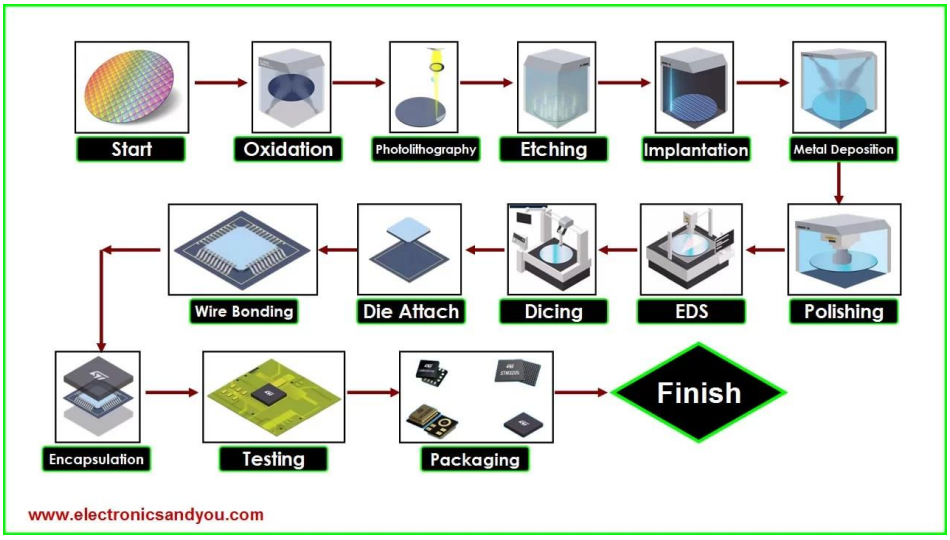

FIGURE 7.1

CHAPTER 8
REFERENCES

CHAPTER 8 REFERENCES

Research papers -

- (T. Zheng, G. Cai and Z. Huang, "A Soft RISC-V Processor IP with High-performance and Low-resource consumption for FPGA," 2022 IEEE International Symposium on Circuits and Systems (ISCAS), Austin, TX, USA, 2022, pp. 2538-2541, doi: 10.1109/ISCAS48785.2022.9937742.)
- MIPS® Architecture For Programmers Volume II-A: The MIPS32® Instruction Set
- The RISC-V Instruction Set Manual Volume I: User-Level ISA Document Version 2.2
- ARM Architecture Reference Manual Copyright © 1996-1998, 2000, 2004, 2005 ARM Limited. All rights reserved.
- ECE6133: Physical Design Automation of VLSI Systems Georgia Institute of Technology Prof. Sung Kyu Lim
- Command Reference for Encounter RTL Compiler Product Version 9.1 July 2009
- (Urquhart, Roddy (29 March 2021). "What Does RISC-V Stand For? A brief history of the open ISA". Systems & Design: Opinion. Semiconductor Engineering.)
- ("The History of the Integrated Circuit". Nobelprize.org. Archived from the original on 29 Jun 2018. Retrieved 21 Apr 2012.)
- (Pawson, Richard (30 September 2022). "The Myth of the Harvard Architecture". IEEE Annals of the History of Computing. 44 (3): 59–69. doi:10.1109/MAHC.2022.3175612. S2CID 252018052)
- "Design Of 32 Bit Asynchronous RISC-V Processor Using Verilog", International Journal of Emerging Technologies and Innovative Research (www.jetir.org | UGC and issn Approved), ISSN:2349-5162, Vol.7, Issue 3, page no. pp1717-1722, March-2020,
- International Research Journal of Engineering and Technology (IRJET) Volume: 08 Issue: 06 | June 2021 . Shruthi1, Dr. Jamuna S2 1PG Student (M.Tech in VLSI Design and Embedded Systems), Department. of ECE, DSCE Bangalore, Karnataka 2Professor, Department. of ECE, DSCE, Bangalore, Karnataka
- Mr. Rajkumar D. Komati1, Aditya Kolekar2, Kunal Kasbekar3, Ms. Avanti Godbole4 1Assistant Professor, Dept. of ENTC, MIT College of Engineering, Kothrud, Pune 2, 3, 4Undergraduate Student Dept. of ENTC, MIT College of Engineering, Kothrud, Pune.