IMPLEMENTATION OF VLSI DESIGN FLOW FOR MIPS-BASED SOC

LEE ZHAO MIN

A project report submitted in partial fulfilment of the requirements for the award of the degree of Bachelor of Engineering (Hons) Electronic Engineering

Faculty of Engineering and Green Technology
Universiti Tunku Abdul Rahman

May 2022

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature	:	[illegible]
Name	:	Lee Zhao Min
ID No.	:	17AGB04345
Date	:	25.4.2022

APPROVAL FOR SUBMISSION

I certify that this project report entitled **“IMPLEMENTATION OF VLSI DESIGN FLOW FOR MIPS-BASED SOC”** was prepared by **LEE ZHAO MIN** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Hons) Electronic Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature	:	[illegible]
Supervisor	:	Ir Dr. Loh Siu Hong
Date	:	29/4/2022

ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my deep sense of gratitude to my supervisor, Ir Dr. Loh Siu Hong for his invaluable advice, guidance, and his enormous patience throughout the development of the research.

In addition, I would also like to express my gratitude to my family, course mates and friends who had helped and given me encouragement.

IMPLEMENTATION OF VLSI DESIGN FLOW FOR MIPS-BASED SOC

ABSTRACT

MIPS is a VLSI microprocessor based on RISC architecture which focuses on increasing the performance with the trade-off of its hardware and instruction complexity. VLSI design flow is the common design methodology used for integrated circuit design. The two phases in the VLSI design flow are front-end design and back-end design. The complete VLSI design flow is implemented to produce a MIPS-based SoC. EDA tools from Synopsys Inc are used in this project to carry out the processes including logic synthesis, floorplanning, placement, routing, physical verification, and others. The use of EDA tools could shorten the long VLSI design cycle with design automation. The MIPS design is optimised to reduce the design cost and improve the performance of the design. Synopsys Design Compiler is used for the front-end design while Synopsys IC compiler is used to complete the back-end design. The final layout produced from the IC compiler is able to pass the timing and physicals verifications.

TABLE OF CONTENTS

DECLARATION **ii**
APPROVAL FOR SUBMISSION **iii**
ACKNOWLEDGEMENTS **v**
ABSTRACT **vi**
TABLE OF CONTENTS **vii**
LIST OF TABLES **x**
LIST OF FIGURES **xi**
LIST OF SYMBOLS / ABBREVIATIONS **xvi**
CHAPTER **1**

1 INTRODUCTION **1**
1.1 Background 1
1.1.1 IC design 1
1.1.2 CPU architecture 2
1.2 Problem Statements 4
1.3 Aims and objectives 5

2 LITERATURE REVIEW **6**
2.1 CISC architecture 6
2.2 RISC Architecture 7
2.2.1 MIPS architecture 7
2.3 RICS and CISC comparison 9
2.4 MIPS instruction 9
2.4.1 R-type 10
2.4.2 I-type 10
2.4.3 J-type 10
2.5 Type of MIPS Microarchitectures 11
2.5.1 Single-cycle microarchitecture 11

2.5.2 Multicycle microarchitecture 12
2.5.3 Pipelined microarchitecture 13
2.6 VLSI design flow 15
2.6.1 Digital design using HDL 18
2.6.2 Logic synthesis 18
2.6.3 Physical design 19
2.6.3.1 Partitioning 19
2.6.3.2 Floorplanning 21
2.6.3.3 Placement 22
2.6.3.4 Routing 22
2.6.3.5 Physical Verification 23

3 METHODOLOGY 24
3.1 VLSI Design Methodology 24
3.2 Front-end design 26
3.2.1 Register Transfer Level (RTL) Design 26
3.2.2 Logic Synthesis 26
3.2.3 Static Timing Analysis 27
3.3 Back-end design 28
3.3.1 Floorplanning 28
3.3.2 Power Network Synthesis 28
3.3.3 Placement 29
3.3.4 Clock Tree Synthesis 29
3.3.5 Routing 29
3.3.6 Chip Finishing 30
3.3.7 Physical Verification 30
3.4 Design Tools 31
3.4.1 Synopsys Design Compiler 31
3.4.2 Synopsys IC Compiler 32

4 RESULTS AND DISCUSSIONS 33
4.1 Design Verification 33
4.1.1 Design Compilation and Simulation 33
4.2 Logic Synthesis 41

4.2.1 DC Setup and Design Translation 41
4.2.2 Design Checking before mapping and optimization 43
4.2.3 Analysis of Timing, Area and Power before Mapping 46
4.2.4 Mapping and Optimization of Design 48
4.2.5 Analysis of Timing, Area and Power after Mapping and Optimization 50
4.2.6 Design Checking after Mapping and Optimization 53
4.2.7 Output Gate-level Netlist 58
4.3 Physical design 59
4.3.1 ICC Setup and Gate-netlist import 59
4.3.2 Floorplanning 61
4.3.3 Placement, CTS, and Routing 70
4.3.4 Chip Finishing and Tape Out 75

5 CONCLUSION AND RECOMMENDATIONS 84
5.1 Conclusion 84
5.2 Recommendation 85
REFERENCE 87
APPENDICES 89

LIST OF TABLES

TABLE	TITLE	PAGE
2.1	MIPS register set (Harris and Harris, 2013)	8
3.1	Instructions supported by the design	26
4.1	Expected data stored in RAM after executing the program	34

LIST OF FIGURES

FIGURE	TITLE	PAGE
2.1	MIPS instruction format (Harris and Harris, 2013)	10
2.2	Example of a single-cycle MIPS processor (Harris and Harris, 2013)	11
2.3	Timing diagram of a single-cycle processor and a pipelined processor (Harris and Harris, 2013)	14
2.4	VLSI design flow (Kishore and Prabhakar, 2009)	17
2.5	Flow of logic synthesis (Das, 2010)	19
2.6	Partitioning example using 2 different cuts (Kahng *et* al, 2011)	20
3.1	Top-down approach for VLSI design methodology	25
3.2	General flow for front-end design using Design Compiler	31
3.3	General flow in IC Compiler for back-end design	32
4.1	The source codes are compiled and simulated successfully with correct results	34
4.2	Output waveform for cycle 1 - 9 of the program	35
4.3	Output waveform for cycle 10 - 18 of the program	36
4.4	Output waveform for cycle 19 - 27 of the program	37
4.5	Output waveform for cycle 28 - 36 of the program	38
4.6	Output waveform for cycle 37 - 45 of the program	39

4.7	Output waveform for cycle 46 - 51 of the program	40
4.8	Libraries setup result in DC	41
4.9	Result of loading and compilation of source codes in DC	42
4.10	Result of building and translating source codes in DC	42
4.11	Result of listing libraries used in DC	43
4.12	Summary of check design before mapping.	44
4.13	Detail of unconnected ports before mapping	44
4.14	Detail of constant and shorted outputs	45
4.15	Detail of unloaded nets warning	45
4.16	Timing report before mapping	46
4.17	Area report before mapping	47
4.18	Power analysis before mapping	48
4.19	Execution and result of mapping and optimization process	49
4.20	Result of timing report after mapping	50
4.21	Area report after mapping	51
4.22	Power report after mapping	52
4.23	Unconnected ports in optimised design	54
4.24	Information of part of the unconnected ports	55
4.25	Connection of adder from pcreg module with connected ports	55
4.26	Final result of check design after removing unconnected ports	55
4.27	Detail of the warnings in check design after removing unconnected ports	56
4.28	Unloaded nets in pcreg module	56

4.29 Unloaded nets in alu1 module 56

4.30 Hold time violation 57

4.31 Result showing min capacitance and max area violation after mapping 57

4.32 Result for output gate-level netlist 58

4.33 Load tech file during design library creation 59

4.34 Setup of tluplus files 60

4.35 Import of design and loading of libraries 60

4.36 Initial view of the top module after loading the design 60

4.37 Checking on libraries loading 60

4.38 Setting to create floorplan and summary of the floorplan 62

4.39 Floorplan generated by the tool 62

4.40 Summary of virtual flat placement 63

4.41 Result of virtual flat placement 63

4.42 Global route congestion map 64

4.43 Enlarged congestion map with congestion calculation 65

4.44 Congestion report based on GRC 65

4.45 Setting applied to PNS 67

4.46 Maximum IR drop reported in PNS summary 67

4.47 Heat map for PNS 68

4.48 Chip layout after committing the power network 68

4.49 Result of placement legalization 69

4.50 Congestion report based of GRC after global routing 69

4.51	Result for timing optimization and fixing of design rule violation	69
4.52	Summary of placement optimization	70
4.53	CTS result	71
4.54	Part of QoR report after CTS	71
4.55	Clock tree summary	72
4.56	Status of clock tree routing	72
4.57	Visualisation of the clock tree	72
4.58	Result of initial routing.	74
4.59	Result of postroute optimization	74
4.60	LVS after initial route	74
4.61	Detail of the floating ports	75
4.62	Short critical area of the design	77
4.63	Open critical area	78
4.64	Metal 3 to be pushed off from the track	78
4.65	Metal 2 to become widen and pushed off from track	79
4.66	Redundant conversion report before chip finishing	79
4.67	Redundant conversion report after chip finishing	80
4.68	Final DRC verification result	80
4.69	Final LVS verification result	80
4.70	Final timing report	81
4.71	Final area report	81
4.72	Final power report	82
4.73	QoR report showing hold time violation information	82
4.74	Result of outputting GDSII file	82

4.75 Final layout 83

4.76 Enlarged view of the layout 83

LIST OF SYMBOLS / ABBREVIATIONS

CAD	Computer-aided Design
CISC	Complex Instruction Set Computer
CPU	Computer Processor Unit
CTS	Clock Tree Synthesis
DRC	Design Rule Checker
EDA	Electronic Design Automation
GDSII	Graphic Database System II
GUI	Graphical User Interface
HDL	Hardware Description Language
IC	Integrated Circuit
ISA	Instruction Set Architecture
LVS	Layout vs Schematic
MIPS	Microprocessor without Interlocked Pipe Stages
PNS	Power Network Synthesis
RISC	Reduced Instruction Set Computer
RTL	Register Transfer Level
SoC	System-on-Chip
STA	Static Timing Analysis
VLSI	Very large-scale Integration

CHAPTER 1

INTRODUCTION

1.1. Background

1.1.1. IC design

According to the Semiconductor Industry Association (SIA) (2021), the semiconductor market is experiencing an increasing trend in sales globally and the sales growth rate of the year 2021 is forecast at 20 percent. The increasing sales are projecting the high demand for semiconductors worldwide. The advent of Industrial Revolution 4.0 (IR 4.0) comes with the rapid development of smart devices and gadgets, involving more use of chips. The chip can be foreseen to have an increasing demand with the ongoing IR 4.0 era. IC is playing an important role in our daily life. The application of IC chips ranges from consumer electronics, computing, industrial, military, aerospace, automotive, wired or wireless communication (King, Wu, and Pogkas, 2021).

In the past decades, the prediction from Gordon Moore, the founder of Intel has been guiding the technology development in the semiconductor industry. Moore’s Law foreseen the IC chip will double the quantity of transistors every 24 months. By doubling the complexity, a chip is expected to grow with improved performance, decreasing cost, and higher reliability (Takahashi, 2005). The leading semiconductor manufacturer, TSMC (Taiwan Semiconductor Manufacturing Company) has put the 5 nm technology into production using Field-Effect Transistor (FinFET) in 2020 and the company is planning to further downscale the transistor to 4nm for production by 2022

(TSMC, 2020). The continuous development of shrinking the size of the transistor is pushing the chip to its physical limit (Takahashi, 2005).

Very large-scale integration (VLSI) technology refers to a single IC chip to have metal oxide semiconductor field effect transistors (MOSFET) with a quantity of hundreds of thousands while system-on-chip (SoC) contains billion or more transistor per chip (Xiu, 2008). The semiconductor material is the main material used in VLSI chip manufacturing as it allows the control of conductivity for the tiny well-defined area (Kishore and Prabhakar, 2009). In this era, design automation is used to assist in the stages of the VLSI design flow.

Electronic Design Automation (EDA) tools can support the engineers in the chip manufacturing process such as planning, design, verification, and other stages (Gianfagna, 2021). In the late 1960s, the first EDA tool to optimise the placement of devices on a circuit board was created (Kahng *et* al, 2011). The EDA equipped with programmed circuit synthesis function and able to route the design automatically had become dominant in the market by the 1990s (Kahng *et* al, 2011). Nowadays, the EDA tool is further completed with various functions towards design automation and the whole VLSI design flow is linked to the EDA tool (Kahng *et* al, 2011). Some of the major EDA software companies in the market include Mentor Graphics, Cadence Design Systems, and Synopsys (Kahng *et* al, 2011).

1.1.2. CPU architecture

The central processor unit (CPU) or processor consists of 2 major components which are datapath and control to act as the active part of the computer to perform an arithmetic operation, send signals to activate input/output devices (Patterson and Hennessy, 2014). Datapath is to carry out the arithmetic operand for the CPU while control is responsible to instruct the datapath, input/output devices, and memory to respond according to the instruction from the program (Patterson and Hennessy, 2014).

The abstract interface between the lowest-level software and hardware which consists of a machine language program to include all the essential information to ensure the correct running of the program is known as instruction set architecture (ISA) or architecture (Patterson and Hennessy, 2014). The improvement of the performance of the architecture could be in several forms. One of the methods is to increase the number of things to be done by a instruction and the other method is to cut the number of instructions used to execute a particular function (Abd-El-Barr and El-Rewini, 2005). This is to reduce the number of operations to read or write memory which could result in faster performance (Abd-El-Barr and El-Rewini, 2005). Complex instruction set computer (CISC) philosophy suggested having complex instruction followed by an increase in the number of addressing modes to overcome the schematic gap during the conversion from high-level language to machine language (Abd-El-Barr and El-Rewini, 2005). For instance, Pentium from intel, PowerPC by IBM & Macintosh, and MC68000 from Motorola are some of the machines using the CISC approach (Abd-El-Barr and El-Rewini, 2005). Aside from that, the other popular approach is reduced instruction set computer (RISC) which allows the frequently used operation to be faster through a simpler instruction set and a smaller number of addressing modes (Abd-El-Barr and El-Rewini, 2005). The scalable processor architecture (SPARC) from Sun Microsystem and Microprocessor without Interlocked Pipe Stages (MIPS) architecture are the examples of RISC approach.

1.2. Problem Statements

The problem statements of the project are as shown below:

- Long turnaround time to design an MIPS-based SoC IC
- High cost to design an IC manually
- Low efficiency to execute instruction for Complex Instruction Set Computer (CISC) architectures
- Higher cost to implement CISC architectures

The VLSI design of an IC chip is a complex process that requires a long period to complete if the work is done from scratch without the help of EDA tools. The time to design a VLSI chip from scratch manually could cost years of effort (Kowalski *et al*, 1985). Design of VLSI chip with only human force involves the process of drawing logic gates manually on paper. This led to the high design cost of VLSI in terms of time and resources. Besides, the cost of making an error is high. The process of design and manufacture has to be repeated when there is an error on the end product.

The CISC architecture emphasis on hardware and it requires complex instruction to execute a simple operand. This would reduce the efficiency of the chips in executing the operand. The complex instruction and additional hardware of CISC will increase the design cost (Harris and Harris, 2013). Therefore, the cost required to implement a CISC microprocessor is high.

1.3. Aims and objectives

The objectives of the project are stated as follows:

1. To accomplish the complete VLSI design flow from front-end to back-end using MIPS architecture.
2. To shorten the design turnaround time through EDA tools.
3. To synthesis the model of MIPS with complete timing constraints.
4. To optimize the chip area and reduce the design cost.

In this project, a MIPS based SoC will be built by going through the full VLSI design flow from the front-end to the back end. A single cycle MIPS processor will be built to increase the chip efficiency. The design steps to be gone through includes logic synthesis, floorplanning, routing, partitioning, and others. EDA tools will be used to complete the IC design flow to achieve design automation. This is to shorten the time consumed for VLSI design and verify the design to meet the requirement for manufacturing.

The design is to be synthesis with complete timing constraints to ensure the functionality of the chip after manufacturing. At the same time, the chip area will be optimised to reduce the cost of design. Besides, the performance of the design can be boost through the optimisation process.

CHAPTER 2

LITERATURE REVIEW

2.1. CISC architecture

Complex Instruction Set Computer (CISC) refers to the architectures which have instructions with high complexity (Harris and Harris, 2013). Overhead is added to every instruction in CISC approach without considering the complexity even if the instruction is used in a low frequency (Harris and Harris, 2013). The common features of a CISC approach includes different kind of addressing mode, ample instruction set, and instruction set in multiple format and sizes (Jamil, 1995). A CISC processor has a microprogrammed control, and it is able to execute several instructions in different cycle (Jamil, 1995). Since the format, addressing modes, and opcode in large quantity have to be differentiated by the control unit, the control unit is said to be complex (Jamil, 1995).

2.2. RISC Architecture

Reduced Instruction Set Computer (RISC) is an architecture with a simple instruction set as well as hardware implementation (Harris and Harris, 2013). The common characteristics of a RISC processor include a reduced instruction set, regular format for instruction, redundant number of general-purpose register load and store operation for memory, only 1 instruction is executed every machine cycle, instruction set or execute units which are pipelined and control unit design with hardwired (Abd-El-Barr and El-Rewini, 2005).

2.2.1. MIPS architecture

MIPS or Microprocessor without Interlocked Pipe Stages, a VLSI microprocessor based on RISC (Reduced instruction set computer) developed by a team under the lead of John Hennessy from Standard University in the 1980s. In 1985, The team introduced R2000, the first MIPS processor to the market followed by R3000 after 2 years. The first 64-bit MIPS microprocessor was brought to the market in 1991 along with the rapid development of MIPS architecture (Lamie, 2009). The MIPS architecture is widely used in our daily life including home networking, communication, game console, and even electric vehicle. According to Voica (2016) and Silbert (2012), MIPS microprocessor is used for PlayStation from Sony, Nintendo 64, Smart Tab 1 of Karbonn Mobiles, and Tesla Model S. A MIPS architecture focuses on decreasing the complexity of individual instruction and hardware to boost the performance (Hennessy *et al*, 1982).

There are only 32 registers for MIPS architecture and the small number of registers allows the MIPS to have faster performance as it reduces the time consuming to read data from the register as compared to the CPU with a large set of the register (Harris and Harris, 2013). For MIPS, the variables are stored in two types of register naming saved register, *$s0* to *$s7* and temporary register, *$t0* to *$t9*, with the temporary register specifically to save intermediate or temporary variables (Harris and Harris, 2013). The register set for MIPS is shown in Table 2.1.

Table 2.1: MIPS register set (Harris and Harris, 2013)

Register name	Number	Function
$0	0	Contain value 0
$at	1	Assembler temporary
$v0 - $v1	2 - 3	Function return value
$a0 - $a3	4 – 7	Function arguments
$t0 - $t7	8 -15	Store temporary variables
$s0 - $s7	16 – 23	Saved variables
$t8 – St9	24 – 25	Store temporary variables
$k0 - $k1	26 – 27	Operating system (OS) temporaries
$gp	28	Global pointer
$sp	29	Stack pointer
$fp	30	Frame pointer
$ra	31	Function return address

Besides register, MIPS also stores the data in the memory which has more data locations than register with the trade-off of longer accessing time (Harris and Harris, 2013). The memory address and data words for MIPS are both 32-bit (Harris and Harris, 2013). As a byte-addressable memory, MIPS has a unique address for every byte of the memory (Harris and Harris, 2013).

2.3. RICS and CISC comparison

The main difference between the two architecture is RICS emphasis on software while CISC emphasis more on hardware (Jamil, 1995). The design objective of an RICS is to reduce the instruction execution time to the minimum by scarifying the program length (Jamil, 1995). CISC is the opposite of RISC in which maximizes the instruction to have minimum program length (Jamil, 1995).

Therefore, when both architectures are to perform the same function, a RISC approach needs to execute more instructions than a CISC approach. RISC processor is hence having many CPU registers, extra instruction caches, and decoders to overcome the downside of long instruction (Abd-El-Barr and El-Rewini, 2005). This enables the RICS to have reduced traffic between memory and processor (Abd-El-Barr and El-Rewini, 2005). While for CISC, there is a logic delay as its complex instruction comes with a complex decoding scheme (Abd-El-Barr and El-Rewini, 2005). In terms of instruction, RICS has fixed-length instruction and the instruction length of a CISC approach is variable (Jamil, 1995).

According to Jamil (1995), the chip area of RISC architecture is smaller than CISC architecture as its control unit is simpler. The shrink in the VLSI chip area allows the regularization factor of a RISC approach to be higher, resulting in a decrease in design cost and a more profitable chip (Jamil, 1995).

2.4. MIPS instruction

The instruction used for MIPS architecture is 32-bit. The 3 formats of instruction for MIPS include Register type (R-type), Intermediate type (I-type), and Jump type (J-type). The hardware of MIPS is beneficial to form its small number of instruction formats since the limitation in regularity reduces the complexity of hardware (Harris and Harris, 2013).

2.4.1. R-type

R-type instructions have 2 registers to be used as source and 1 register for the destination (Harris and Harris, 2013). The six fields for R-type instruction include operation code (*op*), source register (*rs* and *rt*), destination register (*rd*), shift amount (*shamt)* for shift operation, and function (*func*) (Harris and Harris, 2013). The *op* and *func* occupy 6 bits each and the rest of the fields occupy 5 bits per field (Harris and Harris, 2013). In R-type, the *func* will determine the operation to be executed.

2.4.2. I-type

I-type uses both register operand and immediate operand to hold 4 fields namely *op, rs, rt* and immediate (*imm*) (Harris and Harris, 2013). The bits occupied by *op, rs, rt are* similar to R-type while *imm* is occupying 16 bits (Harris and Harris, 2013).

2.4.3. J-type

J-type is a format specifically used for jump instruction begin with an *op* holding 6 bits data followed by the address operand (*add*) occupying 26 bits to state the address.

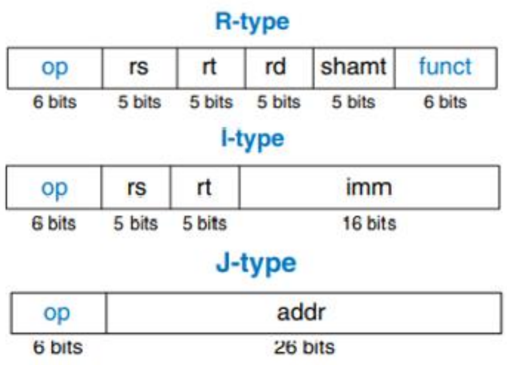


Figure 2.1: MIPS instruction format (Harris and Harris, 2013)

2.5. Type of MIPS Microarchitectures

2.5.1. Single-cycle microarchitecture

Single-cycle microarchitecture means a complete instruction is carried out for a cycle, without the presence of a non-architectural state and its control unit is uncomplicated (Harris and Harris, 2013). For single-cycle process, it requires every instruction to have an equal clock cycle length (Patterson and Hennessy, 2014). There are some limitations in the single-cycle process. The clock cycle must have sufficient length to support the slowest instruction and 3 adders are needed in the system which increases the design cost (Harris and Harris, 2013). Besides, the data memory and instruction are separated in a single-cycle process (Harris and Harris, 2013).

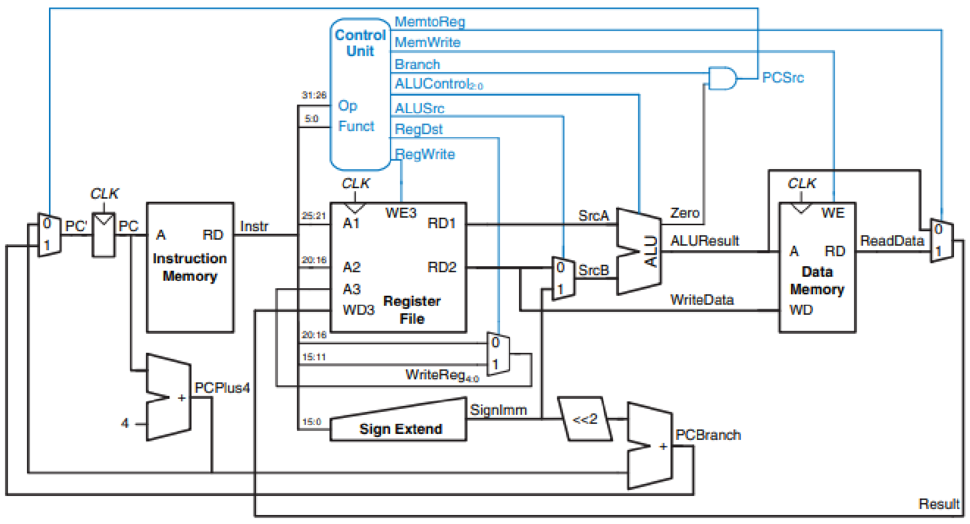


Figure 2.2: Example of a single-cycle MIPS processor (Harris and Harris, 2013)

2.5.2. Multicycle microarchitecture

The multicycle processor implements instruction in a shorter clock cycle and reuses hardware blocks like memories and adders to achieve a cheaper cost for hardware (Harris and Harris, 2013). In the multicycle process, the microprocessor utilizes a few nanoarchitectural registers for the purpose of saving intermedia values to use the hardware block for different aims in the different cycles when executing a particular instruction (Harris and Harris, 2013). Therefore, the instruction could use several clock cycles even if there is one instruction to be executed in a row (Harris and Harris, 2013).

The multicycle approach could overcome the 3 primary weaknesses in the single-cycle processor as it separates one instruction into different steps which are shorter (Harris and Harris, 2013). This allows the processor to have less complex instruction and shorten the execution time (Harris and Harris, 2013). On the other hand, the data and instructions for a multicycle processor are saved in combined memory and only an adder is needed for this approach (Harris and Harris, 2013).

2.5.3. Pipelined microarchitecture

As the name implies, it pipelines the single-cycle microarchitecture to enhance the performance by carrying out a few instructions at the same time (Harris and Harris, 2013). The pipeline approach overlaps several instructions in the execution and hence increases the efficiency (Patterson and Hennessy, 2014). In order to realise this type of microarchitecture, the dependency among the instructions to be executed simultaneously must be managed through the use of logic (Harris and Harris, 2013). Nanoarchitectural pipeline register is another essential component to realise the pipelined microarchitecture.

There are some principles to be applied to create a pipelined processor. The single-cycle processor is separated into 5 phases as shown below (Patterson and Hennessy, 2014):

1. Fetch
    - The instruction is read from the memory
2. Decode
    - The register is read for source code in this stage.
    - Control signal is decoded from the instruction.
    - The process of decoding and reading happens at the same time is allowed for MIPS instruction in regular format.
3. Execute
    - The operation is executed or perform calculation for address.
4. Memory
    - Read or write to the data memory.
5. Write back
    - The result is written back into the register.

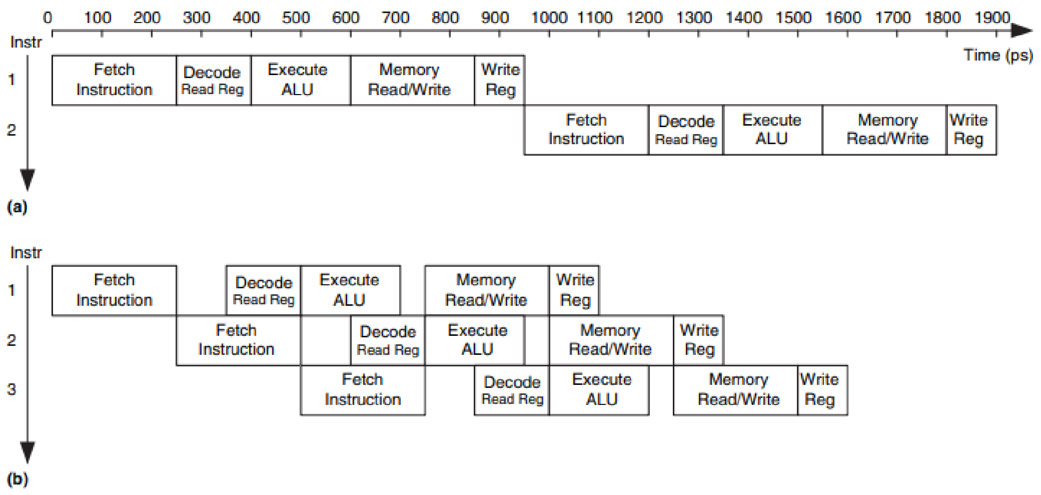


Figure 2.3: Timing diagram of a single-cycle processor and a pipelined processor (Harris and Harris, 2013)

2.6. VLSI design flow

MIPS is a VLSI microprocessor that can be designed using VLSI design flow. The VLSI design process involves design abstraction for a minimum of 5 levels: architectural, register transfer level (RTL), logical design, circuit design, and physical design (Kishore and Prabhakar, 2009). A series of steps to translate a chip idea expressed in RTL format into GDSII data is known as VLSI design flow (Kishore and Prabhakar, 2009). EDA tools or computer-aided design tool (CAD) is commonly used to turn VLAS design a partially or fully automated process (Das, 2010). These automated tools are used to help in processes such as synthesis, design, testing, simulation, and verification (Das, 2010). An example of a VLSI design flow is shown in Figure 2.4.

Concept and market research is the first step of a VLSI design flow to identify the competitiveness of the product in the market. After that, architectural specifications that reflect the design constraints in power consumption, speed, and area are specified (Kishore and Prabhakar, 2009). The hardware description (HDL) capture and RTL coding is the next phase after architectural specification. The function and structure of IC are described using HDL, Verilog and VHDL are the two standardised HDL languages to be applied in the IC design (Kishore and Prabhakar, 2009). RTL description defines the behaviour of a circuit through the description of signal flow among the hardware register and their logical operations (Kishore and Prabhakar, 2009). Next, the RTL simulation should be carried out to verify the logic correctness or functionality of the RTL description (Kishore and Prabhakar, 2009). A testbench program or test vectors can be used to simulate the output of the RTL description program and the result is checked to match with the expected output.

After ensuring the functionality of the RTL netlist, the next phase in VLSI design is logic synthesis. The translation of the behavioural description of the circuit which is commonly in RTL into the logic gate is carried out in logic synthesis to produce the schematic or netlist (Kishore and Prabhakar, 2009). The netlist produced should have the same functionality as its initial RTL code (Kishore and Prabhakar, 2009). During the logic synthesis process, logic optimization is executed to optimize the logic circuit to fulfil the design constraint (Kishore and Prabhakar, 2009). In this phase, the area of the chip is minimized to meet the required delay of the design (Kishore and Prabhakar, 2009). Formal verification is performed in the next phase to make sure the design is able to function correctly, comparison between the design and reference designs is made for the logical functions (Kishore and Prabhakar, 2009). The last step for front-end VLSI design flow is pre-layout static timing analysis (STA). Computation of estimated timing of the design is done in this stage to verify the timing validity (Kishore and Prabhakar, 2009). The path facing the problem of hold or setup violation is identified as well as slow paths, glitch and clock skew (Kishore and Prabhakar, 2009).

Next, physical design processes such as floorplanning, placement, and routing are carried out after verifying the timing. After placement, clock tree synthesis (CTS) is done so that the needed elements receive the clock signal (Kishore and Prabhakar, 2009). These processes will be repeated when the design does not fulfil the timing constraints (Kishore and Prabhakar, 2009). The last phase for a design passing post-layout STA before tape out is verifications including layout versus schematic (LVS) and design rule checking (DVS) (Kishore and Prabhakar, 2009).

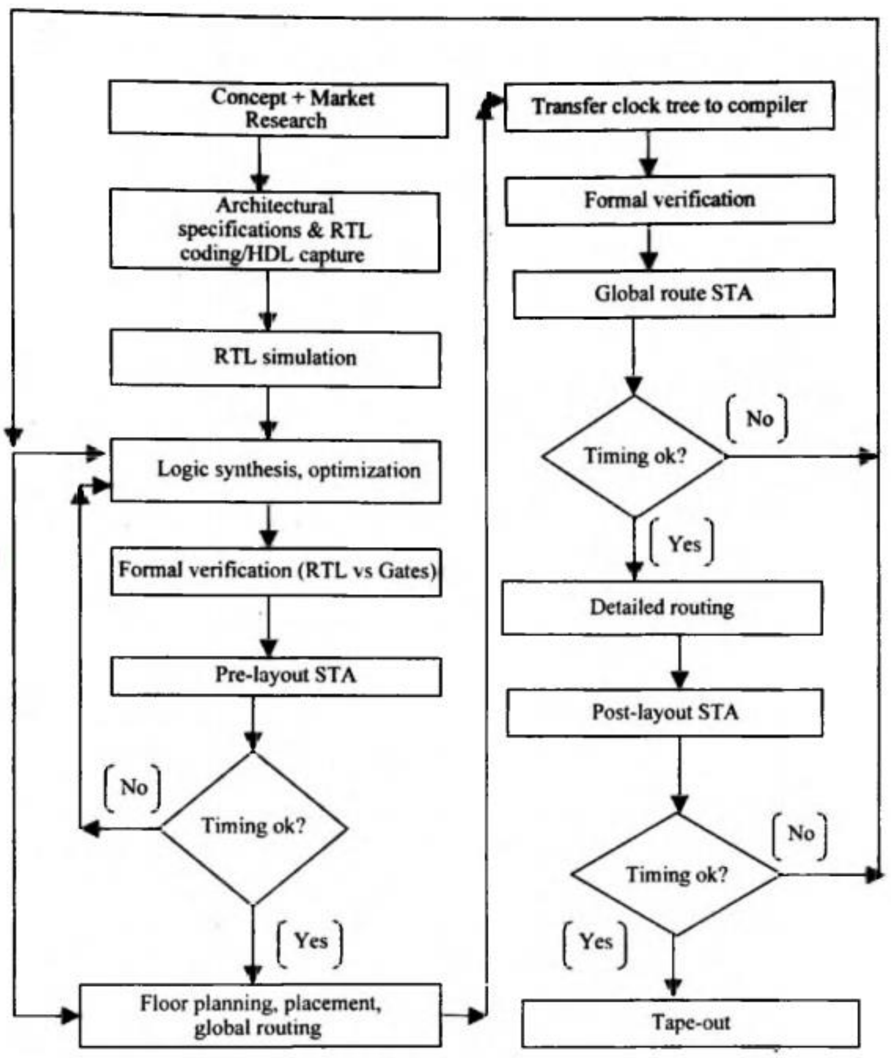

Concept + Market Research
Architectural specifications & RTL coding/HDL capture
RTL simulation
Logic synthesis, optimization
Formal verification (RTL vs Gates)
Pre-layout STA
Timing ok?
No
Yes
Floor planning, placement, global routing
Transfer clock tree to compiler
Formal verification
Global route STA
Timing ok?
No
Yes
Detailed routing
Post-layout STA
Timing ok?
No
Yes
Tape-out

Figure 2.4: VLSI design flow (Kishore and Prabhakar, 2009)

2.6.1. Digital design using HDL

The introduction of the use of HDL language reduces the complexity of the digital design of the VLSI circuit consists of more than ten thousand logic gates (Ramachandran, 2007). With the use of HDL, the design cycle is shortened since the building of circuits using gates in the schematic approach can be avoided as HDL can represent circuits precisely through behavioural, data flow, and RTL description (Ramachandran, 2007).

2.6.2. Logic synthesis

In logic synthesis, the structural design is transformed from the behavioural design (Das, 2010). The levels that could be involved in logic synthesis include the transistor level, block level, and top-level synthesis (Das, 2010). Due to a large number of transistors in a VLSI design, logic synthesis is commonly be executed with the assist of automated tools to optimise the chip in terms of area, power, and speed according to the requirements in the constraint file (Das, 2010).

The cell library used in the logic synthesis process and the algorithm of the synthesis tool will greatly define the quality of the output (Kishore and Prabhakar, 2009). A cell library is a package containing standard cells in a group (Kishore and Prabhakar, 2009). The netlist output from logic synthesis describes the interconnections and instances within the VLSI chip (Kishore and Prabhakar, 2009). It is important for the cell library to include sequential cells of various types so that they can match any storage requirement (Kishore and Prabhakar, 2009). Besides, in order to ensure the cell library is capable of performing any requirement for logic operation, there must be an adequate amount of combinational logic cells in the library (Kishore and Prabhakar, 2009). The main objective of logic synthesis is to fulfil the requirements in area, power, and speed points of view. The flow of logic synthesis is shown in Figure 2.5.

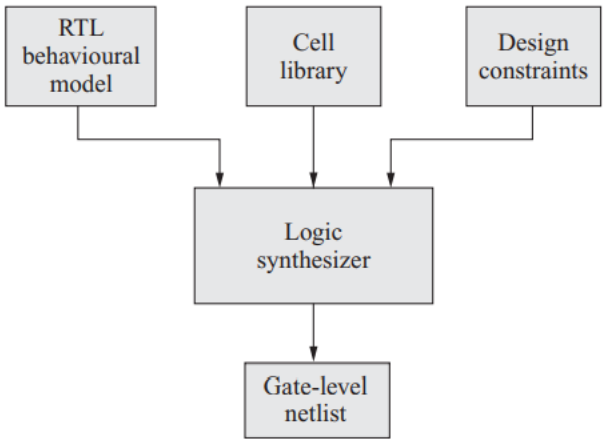


Figure 2.5: Flow of logic synthesis (Das, 2010)

2.6.3. Physical design

Physical design refers to the procedure to generate a physical layout based on the gate-level netlist of the VLSI design (Das, 2010). The components in the design circuit are transformed into a geometric presentation which contains a series of geometric patterns to perform the logical operation of the respective components (Chen, 2009). The four major steps in the physical design of VLSI design flow are partitioning, floor-planning, placement, and routing (Das, 2010). The last step in physical design is to verify the functionality of the design (Das, 2010).

2.6.3.1. Partitioning

There are more than ten thousand transistors in a VLSI chip, making the processing of the whole layout a challenging work due to limited computation power and memory space (Chen, 2009). Therefore, the complete VLSI circuit is divided into subcircuits in this phase (Chen, 2009). This is to ensure the subcircuits have the minimum interconnections among each other (Das, 2010). It is vital for all the subcircuits to fulfil all the prerequired design constraints (Kahng *et* al, 2011).

The factors affecting the partitioning output include the quantity of block and its size as well as the interconnections to link the blocks (Chen, 2009). The circuit will have more delays or become less reliable if the partition is done without taking into account the connections between the blocks (Kahng *et* al, 2011). Aside from that, inter-clock dependencies may be introduced to the design when the connections between the blocks are many and this will affect the productivity of the design (Kahng *et* al, 2011). An example of partitioning using 2 different partition methods is shown in Figure 2.6.

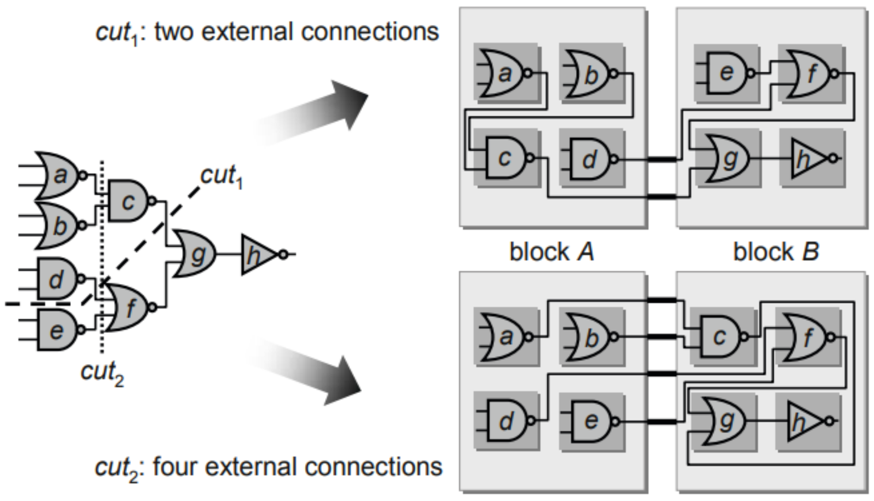

cut₁: two external connections
cut₁
cut₂
block A
block B
cut₂: four external connections
a
b
c
d
e
f
g
h

Figure 2.6: Partitioning example using 2 different cuts (Kahng *et* al, 2011)

2.6.3.2. Floorplanning

The first major step in physical design is floorplanning. The important works in floorplanning are as below (Kishore and Prabhakar, 2009):

- Die size analysation
- Package selection
- Placing of input/output (I/O)
- Placement of macro cells
- Plan for power and clock distribution
- Hierarchy partitioning

Every component and the interconnection between them are planned to be placed with the minimum occupied area (Das, 2010). The design can vary according to different characteristics including core limited, I/O limited, package limited, or block limited (Kishore and Prabhakar, 2009). These characteristics will define the dominating factor in affecting the chip size (Kishore and Prabhakar, 2009). Package selection is done according to factors such as die size, the quantity of I/O, power consumption, and cost (Kishore and Prabhakar, 2009). After that, the prime input and output cells arrangement is carried out and this step is affecting the routing of the chip (Kishore and Prabhakar, 2009). The power distribution network delivers the power to every transistor in the design with suitable voltage level so that they function correctly (Kishore and Prabhakar, 2009).

2.6.3.3. Placement

In the placement process, the placement of the cells in the appropriate location inside the floor plan is done (Kishore and Prabhakar, 2009). This step is to identify the physical layout of the VLSI design and act as the groundwork for the routing process (Das, 2010).

This is an important step in physical design as it impacts the chip area and speed. Good placement can save more area and keep the chip area as small as possible (Kishore and Prabhakar, 2009). The routing will be difficult, or the design could be unroutable if the placement is poor (Kishore and Prabhakar, 2009). With a good placement, the overall delay of the design can be reduced to the minimum as the critical paths are having the shortest available length for wire and hence boost the chip performance speed (Kishore and Prabhakar, 2009). Thus, the goals of placements are to keep the wire length as short as possible for each of the nets and reduce the possibility of interconnection congestion to the minimum (Das, 2010).

2.6.3.4. Routing

Routing is a step to finalize the defined interconnection in the netlist physically after the cells and pins with their exact location defined in placement (Kishore and Prabhakar, 2009). The wire to connect the signal, ground, power, and clock nets are drawn in this step (Das, 2010). It is crucial to make sure there is no short circuit in the nets (Kishore and Prabhakar, 2009).

There are 2 phases in routing which are global routing and detailed routing (Chen, 2009). Global routing refers to the planning of routing where there is no routing work done in this phase (Das, 2010). Global routing is carried out at the top level to define the routing regions, determine the channel terminals, and assign the nets to specific routing regions (Das, 2010). For detail routing, it is doing the real routing and it can be categorised into 2 groups which are area routing and channel routing (Das, 2010).

2.6.3.5. Physical Verification

Physical verification is the last step in physical design. Physical verification is essential to make sure the proper functioning of the layout in terms of electrical and logical (Kahng *et* al, 2011). The categories of physical verification are as shown below (Kahng *et* al, 2011):

- Layout vs Schematic (LVS)
- Design Rule Checking (DRC)
- Electrical Rule Checking (ERC)
- Parasitic extraction
- Antenna Rule Checking

LVS checks the matching of layout with the netlist (Kishore and Prabhakar, 2009). A netlist is extracted from the graphic design system (GDS) file which includes the information of physical layout of a circuit and the differences between the extracted netlist and the original netlist from the design are identified (Luo *et al,* 2010).

DRC is responsible to check the layout with a set of guidelines or rules applicable to an IC layout to ensure the design is manufacturable (Das, 2010). The semiconductor makers will provide the parameters for design rules for verification of the layout suitability (Kishore and Prabhakar, 2009). This verification process emphasis on physical aspects only without considering design timing or logical operation (Kishore and Prabhakar, 2009).

Besides, ERC is the verification to prove the connection of power and ground are correct (Luo *et al*, 2010). At the same time, it also examines the slew, fanouts, and capacitive loads to guarantee they are bounded correctly (Luo *et al,* 2010). In short, the network connectivity of the design is verified in ERC.

Parasitic extraction is a process to validate circuit electrical characteristics through deriving the parasitic elements from the geometric information (Luo *et al,* 2010). Antenna rule checking is to avoid harm to logic gates in the manufacturing process due to antenna effects (Luo *et al,* 2010).

CHAPTER 3

METHODOLOGY

3.1. VLSI Design Methodology

In VLSI design methodology, there are two design styles available which are the top-down approach and the bottom-up approach. The top-down approach as shown in Figure 3.1 was adopted in this project. The focus of this project is to implement a VLSI design flow using a MIPS architecture as the design platform. The whole VLSI design flow was gone through in this project, including both the front-end design and back-end design.

System specification, design of RTL, behavioural design, verification of the design, logic synthesis, and static timing analysis are considered as processes in front-end design. The RTL source codes of MIPS design in this project are retained from the book "Digital Design and Computer Architecture" (Harris and Harris, 2013). The EDA tool used for front-end design methodology is Synopsys Design Compiler (DC).

The back-end design involves the steps to transform the logical design produced in front-end design into the layout or physical design. The processes in back-end design includes floorplanning, power network synthesis (PNS), placement and routing, CTS, STA, chip finishing, post-layout verification, physical verification, and tape out. For back-end design, the design tool to be used is the Synopsys IC compiler.

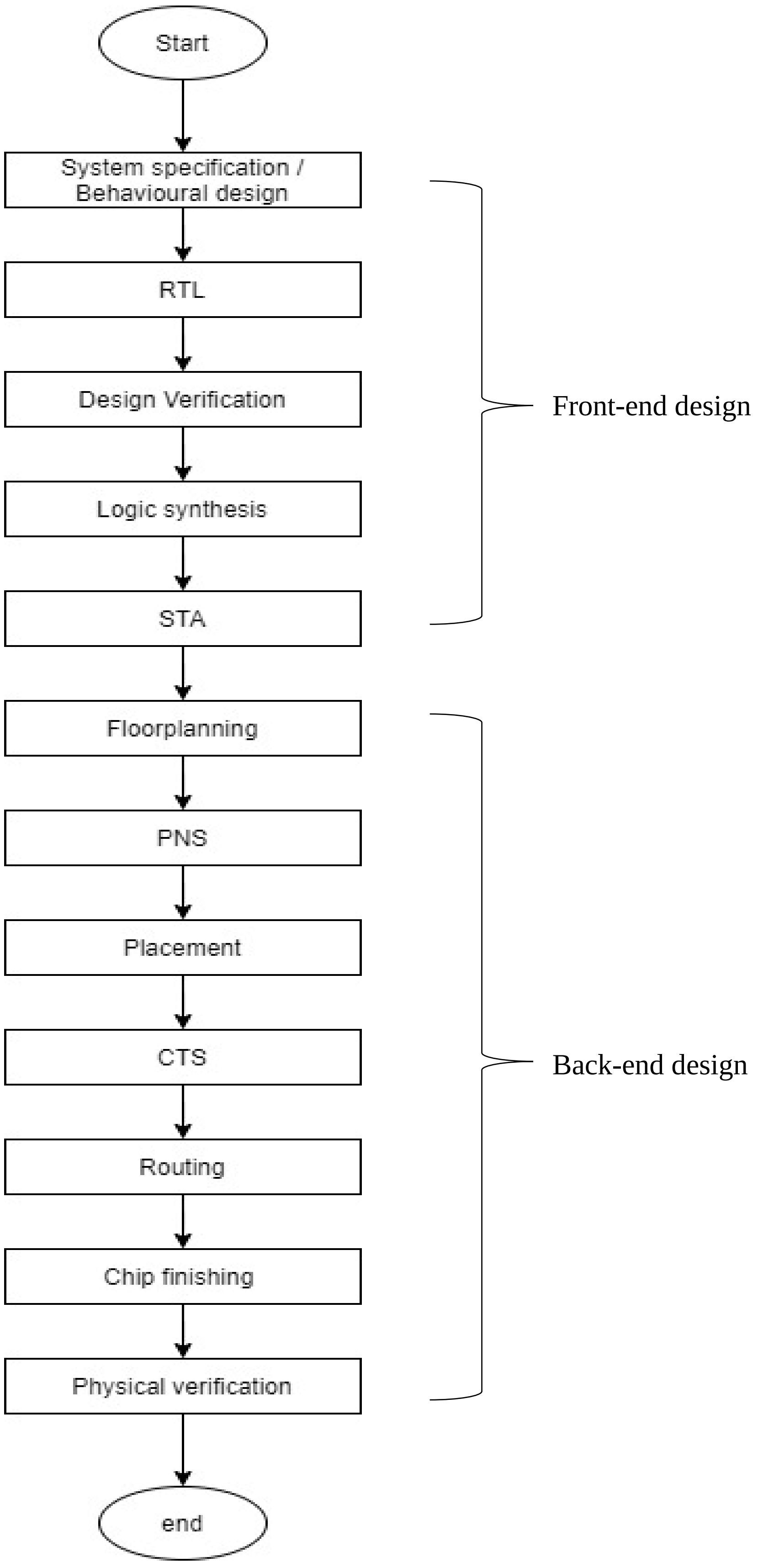


Figure 3.1: Top-down approach for VLSI design methodology

3.2. Front-end design

3.2.1. Register Transfer Level (RTL) Design

The 32-bits MIPS processor is written in System Verilog as the HDL language with ModelSim-Altera 10.1b as the platform. The MIPS processor is capable to support 29 instructions as listed in Table 3.1.

Table 3.1: Instructions supported by the design

Instruction list		
R-type	**I type**	**J type**
ADD	ADDI	J
SUB	LUI	JAL
AND	LW	
OR	SW	
SLT	BEQ	
SLL	ANDI	
XOR	ORI	
NOR	XORI	
SRL	SLTI	
SRA	BNE	
SLLV	BLEZ	
SRLV	BGTZ	
SRAV		
JR		
JALR		

3.2.2. Logic Synthesis

Logic synthesis is started by importing the verified RTL codes into Synopsys Design Compiler, which is the EDA tool to be used in the front-end design. The design constraints such as are constraints and timing constraints are sourced to the design. In this stage, the behavioural design will be transformed into a gate-level netlist. It is a must to ensure the specified technology library is set up in the DC compiler to provide information of cells and logic gates for synthesis. The gate-level netlist is mapped and optimised in terms of performance, speed, power, and area depending on the constraints sourced to the design.

3.2.3. Static Timing Analysis

In this project, the STA is done at block-level using the built-in static timing analysis engine in Synopsis Design Compiler. STA is performed to check for timing violations in the design to determine whether the design meets the timing constraints. The timing constraints must be applied to the design in DC compiler as STA is checking all the path delays by comparing them to the timing constraints.

STA is used during compilation to act as a guide for the compiler to make optimization decisions. Besides, the STA is used to generate timing reports and timing-related reports after compilation. The timing violations are fixed until there is no violated path in the design. After the timing of the design is verified, the front-end design is completed.

3.3. Back-end design

3.3.1. Floorplanning

The floorplanning includes the processes of defining core size, location of I/O, power, corner and filler pad cells, standard cell placement constraints, and power gird. The core utilization ratio in this section is a factor to decide the ratio of the area of the entire cell to be used for cell placement. The ratio is set to a suitable value to make sure there is enough space for clock tree routing and power network. Besides, the chip size and routability of the chip should be considered as well for core utilization ratio selection. For example, if the core utilisation ratio is set to 0.7, the cell placement will use 70% of the core area while the remaining 30% will be saved for routing.

The floorplan is modified until congestion is acceptable or no congestion issue. The modifications that can be made to the floorplan include altering the port or pad locations, use different metal layers, modify the size of the core, and restructure the power grid.

3.3.2. Power Network Synthesis

Power rings and power straps are created in this phase to form a power network and provide sufficient power across the design. Hence, the voltage drop from power pads to the center of the design could be reduced.

Power network constraints for macro rings, core rings, and straps are applied for the tools to perform placement of these elements. The number and width of the power straps can be calculated using the tool according to the required IR drop. IR drop analysis is done to analyse the drop of supply voltage after the power distribution network is created. When the supply voltage drop is below the design range, the power network is adjusted until the IR drop is within the design requirement.

3.3.3. Placement

The placement process can be done manually by placing the standard cells into the desired location in the core area through the drag and drop function of the tool. Alternately, the tool is equipped with an automated placement function which is more efficient and practical. Sufficient space is saved for routing during auto placement according to the requirement stated in the routing constraints to ensure the routability of the design. The physical optimization and power optimization are performed together with the auto placement using the command in the tool. After placement, the congestion and timing violations are verified. The placement is modified until the congestion and timing violations are acceptable.

3.3.4. Clock Tree Synthesis

Clock tree synthesis (CTS) is carried out after the placement phase so that the EDA tool is able to recognise the registers at their exact placement location in the floorplan. Hence, the buffer can be placed at a suitable location to have minimum clock skews and latency. The tool tried to produce a clock tree with a balanced structure using the least levels if possible.

3.3.5. Routing

The IC compiler provides the auto routing function to ease the routing process for a large number of nets in the design. Both global routing and detail routing are performed in this stage. For detail routing, the antenna fixing option is selected to fix antenna violation using the layer jumping approach in this stage. The tool will try to route the design in the way with the least timing, DRC, and LVS violations. The timing, DRC, and LVS violations are verified after detailed routing. The design is rerouted if there are any violations.

3.3.6. Chip Finishing

Chip finishing is the last step of back-end design methodology before tape out. In this phase, the EDA tool is used to reduce critical areas. The critical area is defined as a region with catastrophic spot deflect on an IC which will result in circuit failure (Walker, 1992). If the center of conductive defects falls on the critical area, a short circuit will happen during fabrication while the center of non-conductive defects falling on the critical area will cause an open circuit. Therefore, the tool is reducing the critical area to avoid short and open in the fabrication process.

The next step in chip finishing is to insert redundant vias to act as a support for a single via to reduce yield loss caused by vias failure. This step is optional as the vias are doubled in the detailed routing phase. The timing analysis, DRC, and LVS of the design are checked after each of the insertion steps to ensure the design has no violations.

3.3.7. Physical Verification

The physical verification performed in this project includes DRC, LVS, and the extraction of parasitic. All of the above physical verifications are done using the EDA tool. If the DRC violations occur, the design is rerouted to fix the violation.

While for LVS, it is to check the differences between the physical layout and the optimised gate-level netlist. This is to guarantee there is no change in the logical operation of design during the backed-end design methodology. The design must be verified to be clear from DRC and LVS violations before exporting the GSDII file using the tools for tape out.

3.4. Design Tools

3.4.1. Synopsys Design Compiler

The Design Compiler (DC) can be invoked using two interfaces which are the interactive shell, DC shell, and the interactive graphical user interface (GUI), Design Vision. The DC shell is the command line interface of the Design Compiler while Design Vision provides a graphical visualisation such as schematic generation. The general flow of using DC in this project is illustrated in Figure 3.3. A file named as *.synopsys_dc.setup* is used to setup file for DC to load the Synopsys installation directory, project working directory, and user home directory to the tool. The search path and logical libraries are included in the setup file.

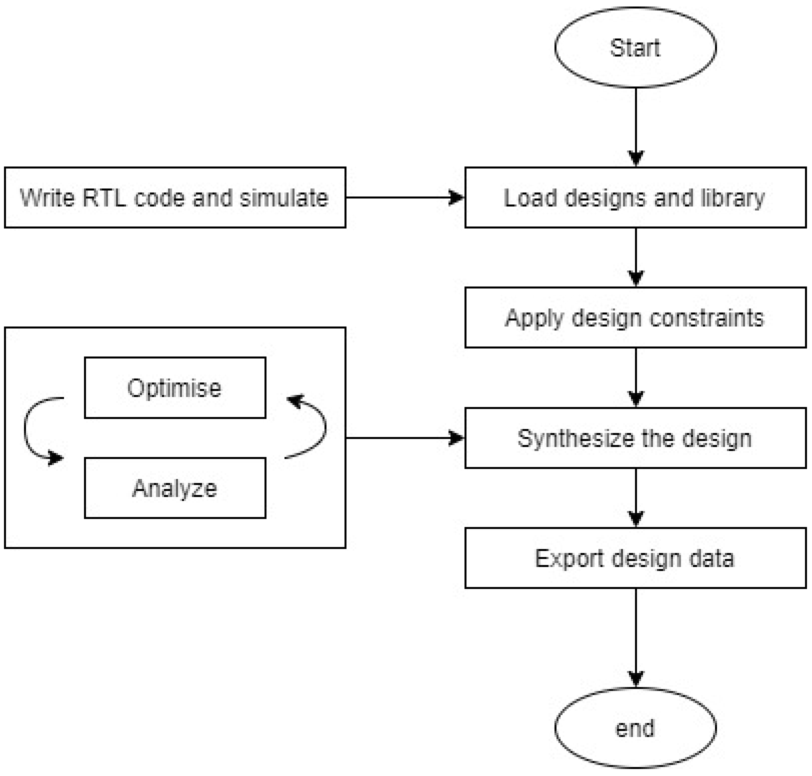


Figure 3.2: General flow for front-end design using Design Compiler

3.4.2. Synopsys IC Compiler

The IC compiler provides two types of interfaces to the user, a command line interface known as ICC shell and GUI. Both interfaces are used in this project. For GUI, it visualizes the design for physical design. For the ICC shell, commands are used in this interface to execute the respective task. The general flow of using the IC compiler in this project is illustrated in Figure 3.4. A file named as *.synopsys_dc.setup* is used to setup file for the IC compiler to load the Synopsys installation directory, project working directory, and user home directory to the tool. The search path, logical libraries, and physical libraries are included in the setup file.

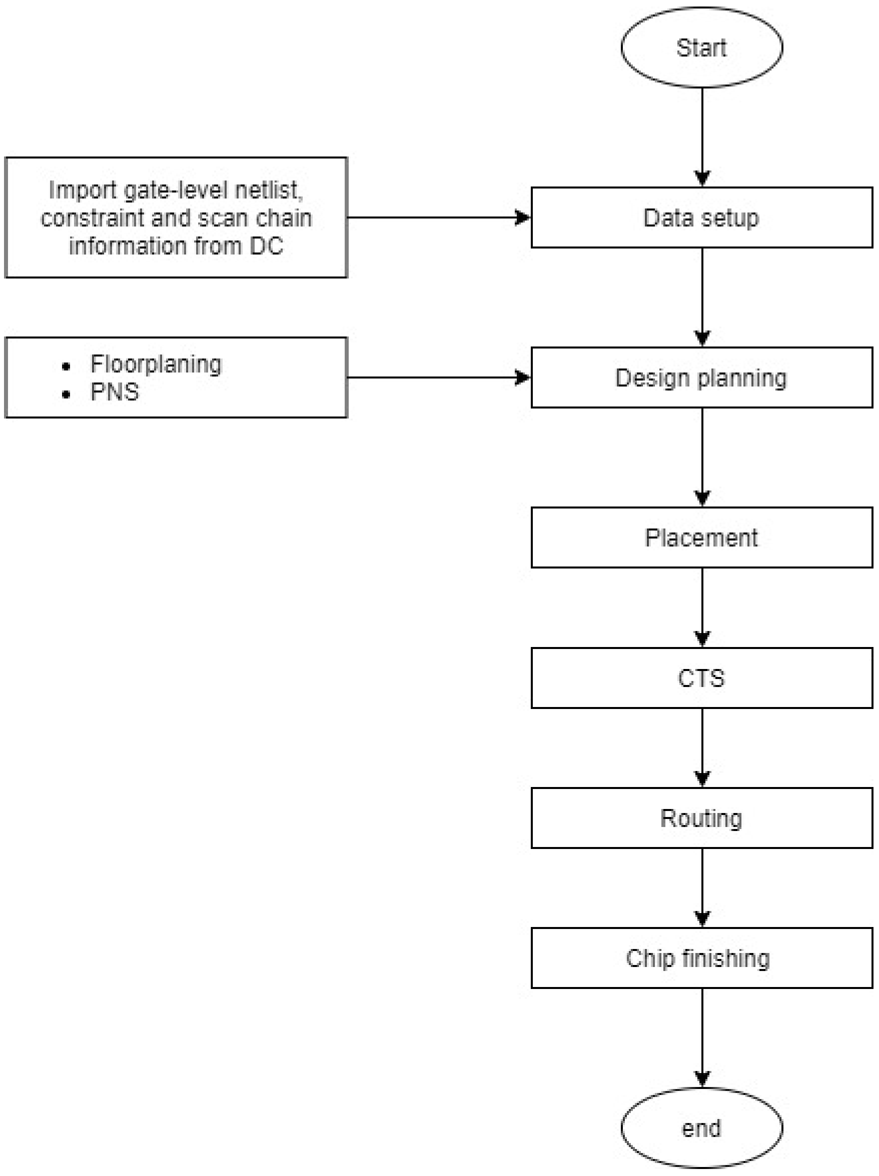


Figure 3.3: General flow in IC Compiler for back-end design

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1. Design Verification

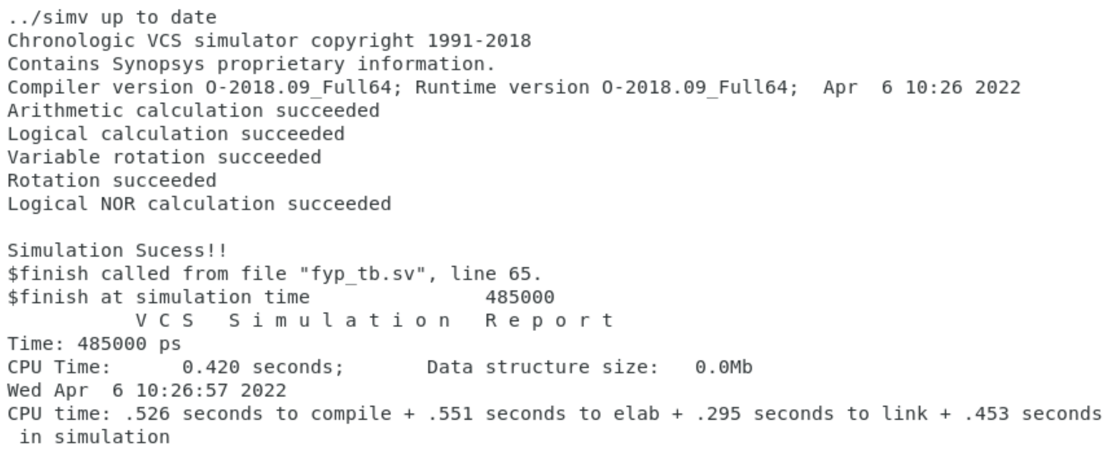
The design verification utilizes the simulation software from Synopsys DVE to check and test the functioning of the MIPS processor designed in this project. It is to ensure the design is able to respond according to the instructions loaded into the design with different input of values and generate a correct outcome.

4.1.1. Design Compilation and Simulation

After the development of the RTL coding for the MIPS processor, the source codes are compiled using Synopsys VCS. A program that performs a computation using all of the 29 instructions supported by the processor is loaded into the instruction memory. The program is designed to relate the result of several instructions and save the computation results into a specific RAM address as listed in Table 4.1. A simple testbench is used to simulate and verify the functionality for all the instructions of the MIPS processor by comparing the data stored in RAM with the expected output. The testbench will generate an error message when the simulated result is not compatible with the expected output. The content of the program and the testbench are shown in the appendices. The simulation result is shown in Figure 4.1. All the instructions are executed correctly and able to generate the desired output. The simulated waveforms of the design generated using DVE are shown below.

Table 4.1: Expected data stored in RAM after executing the program

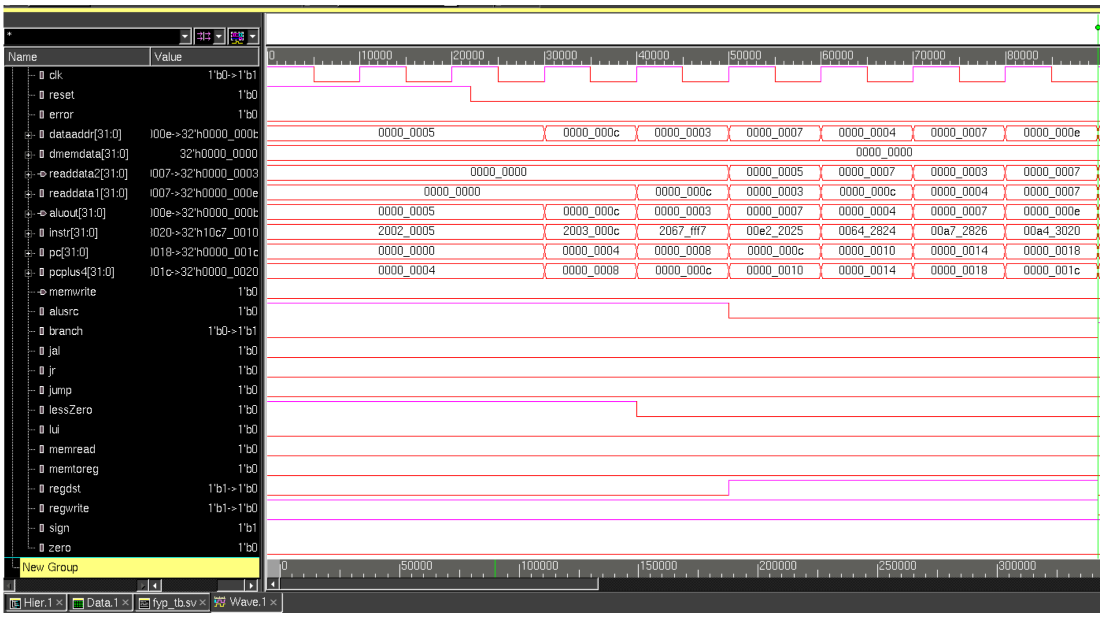
RAM Address	Expected value stored in decimal
RAM[20]	3
RAM[21]	261792
RAM[22]	2
RAM[23]	0
RAM[24]	-152



```
../simv up to date
Chronologic VCS simulator copyright 1991-2018
Contains Synopsys proprietary information.
Compiler version O-2018.09_Full64; Runtime version O-2018.09_Full64;  Apr  6 10:26 2022
Arithmetic calculation succeeded
Logical calculation succeeded
Variable rotation succeeded
Rotation succeeded
Logical NOR calculation succeeded

Simulation Sucess!!
$finish called from file "fyp_tb.sv", line 65.
$finish at simulation time              485000
           V C S   S i m u l a t i o n   R e p o r t
Time: 485000 ps
CPU Time:      0.420 seconds;       Data structure size:   0.0Mb
Wed Apr  6 10:26:57 2022
CPU time: .526 seconds to compile + .551 seconds to elab + .295 seconds to link + .453 seconds
 in simulation
```

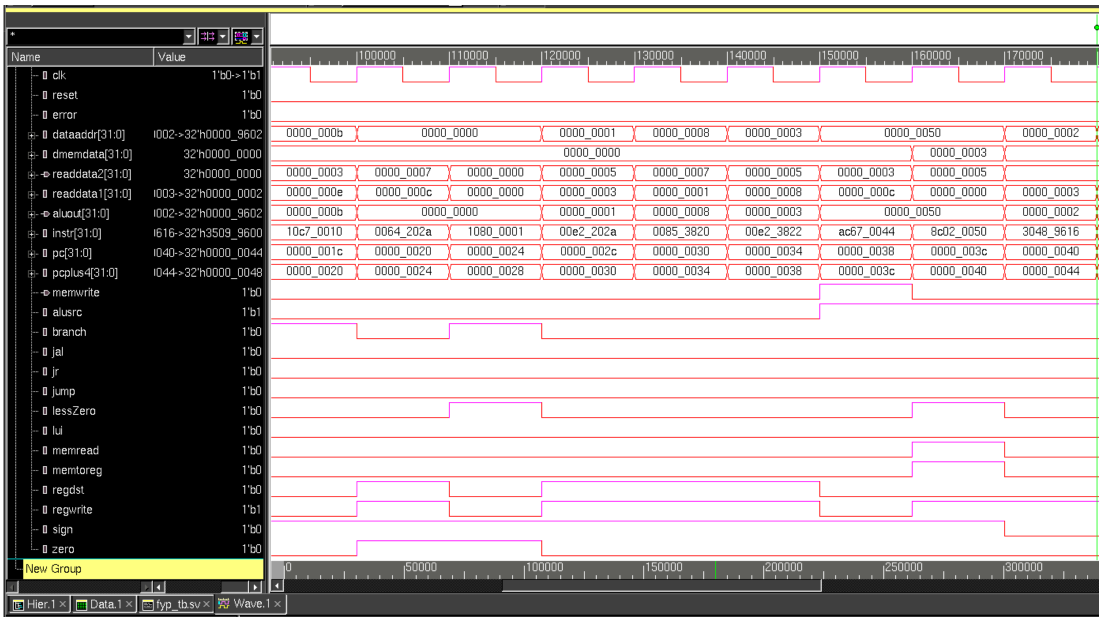
Figure 4.1: The source codes are compiled and simulated successfully with correct results

Figure 4.2 shows the program system is reset (cycle 1 – 2) follows by loading initial values into the respective register using ADDI (cycle 3 – 5) and performing OR, AND, XOR, and ADD using the initial values (cycle 6 – 9). The loading of different instructions into the instruction memory is executed correctly, the instructions are decoded successfully with desired control signals generated.



```
Name               Value
clk                1'b0->1'b1
reset              1'b0
error              1'b0
dataaddr[31:0]     )00e->32'h0000_000b   0000_0005 | 0000_000c | 0000_0003 | 0000_0007 | 0000_0004 | 0000_0007 | 0000_000e
dmemdata[31:0]     32'h0000_0000         0000_0000
readdata2[31:0]    )007->32'h0000_0003   0000_0000 | 0000_0005 | 0000_0007 | 0000_0003 | 0000_0007
readdata1[31:0]    )007->32'h0000_000e   0000_0000 | 0000_000c | 0000_0003 | 0000_000c | 0000_0004 | 0000_0007
aluout[31:0]       )00e->32'h0000_000b   0000_0005 | 0000_000c | 0000_0003 | 0000_0007 | 0000_0004 | 0000_0007 | 0000_000e
instr[31:0]        )020->32'h10c7_0010   2002_0005 | 2003_000c | 2067_fff7 | 00e2_2025 | 0064_2824 | 00a7_2826 | 00a4_3020
pc[31:0]           )018->32'h0000_001c   0000_0000 | 0000_0004 | 0000_0008 | 0000_000c | 0000_0010 | 0000_0014 | 0000_0018
pcplus4[31:0]      )01c->32'h0000_0020   0000_0004 | 0000_0008 | 0000_000c | 0000_0010 | 0000_0014 | 0000_0018 | 0000_001c
memwrite           1'b0
alusrc             1'b0
branch             1'b0->1'b1
jal                1'b0
jr                 1'b0
jump               1'b0
lessZero           1'b0
lui                1'b0
memread            1'b0
memtoreg           1'b0
regdst             1'b1->1'b0
regwrite           1'b1->1'b0
sign               1'b1
zero               1'b0
New Group
```

Figure 4.2: Output waveform for cycle 1 - 9 of the program



```
Name                Value
clk                 1'b0->1'b1
reset               1'b0
error               1'b0
dataaddr[31:0]      I002->32'h0000_9602
dmemdata[31:0]      32'h0000_0000
readdata2[31:0]     32'h0000_0000
readdata1[31:0]     I003->32'h0000_0002
aluout[31:0]        I002->32'h0000_9602
instr[31:0]         I616->32'h3509_9600
pc[31:0]            I040->32'h0000_0044
pcplus4[31:0]       I044->32'h0000_0048
memwrite            1'b0
alusrc              1'b1
branch              1'b0
jal                 1'b0
jr                  1'b0
jump                1'b0
lessZero            1'b0
lui                 1'b0
memread             1'b0
memtoreg            1'b0
regdst              1'b0
regwrite            1'b1
sign                1'b0
zero                1'b0
New Group
```

```
dataaddr   0000_000b  0000_0000                        0000_0001  0000_0008  0000_0003  0000_0050              0000_0002
dmemdata   0000_0000                                                                    0000_0003
readdata2  0000_0003  0000_0007  0000_0000  0000_0005  0000_0007  0000_0005  0000_0003  0000_0005
readdata1  0000_000e  0000_000c  0000_0000  0000_0003  0000_0001  0000_0008  0000_000c  0000_0000  0000_0003
aluout     0000_000b  0000_0000                        0000_0001  0000_0008  0000_0003  0000_0050              0000_0002
instr      10c7_0010  0064_202a  1080_0001  00e2_202a  0085_3820  00e2_3822  ac67_0044  8c02_0050  3048_9616
pc         0000_001c  0000_0020  0000_0024  0000_002c  0000_0030  0000_0034  0000_0038  0000_003c  0000_0040
pcplus4    0000_0020  0000_0024  0000_0028  0000_0030  0000_0034  0000_0038  0000_003c  0000_0040  0000_0044
```

Figure 4.3: Output waveform for cycle 10 - 18 of the program

Figure 4.3 shows that BEQ is tested for both unequal (cycle 10) and equal conditions (cycle 12). SLT is tested for the condition Rs larger than Rt (cycle 11). The result of cycle 11 is related to cycle 12. With branching happening in cycle 12, both the instructions are proved to function correctly under specific conditions. The SLT is again tested with condition Rs lesser than Rt (cycle 13), the result is then used as one of the input values for ADD. The SW (cycle 16) saves the computation result of ADD and SUB to be verified through the testbench. Based on Figure 4.1, the value loaded into data memory and the memory address are verified to be the same as the expected result, proving the proper functioning of instructions from cycles 1 to 16. The next cycle is to load the result saved into data memory in the last cycle to be used for a series of operations starting from cycle 18.

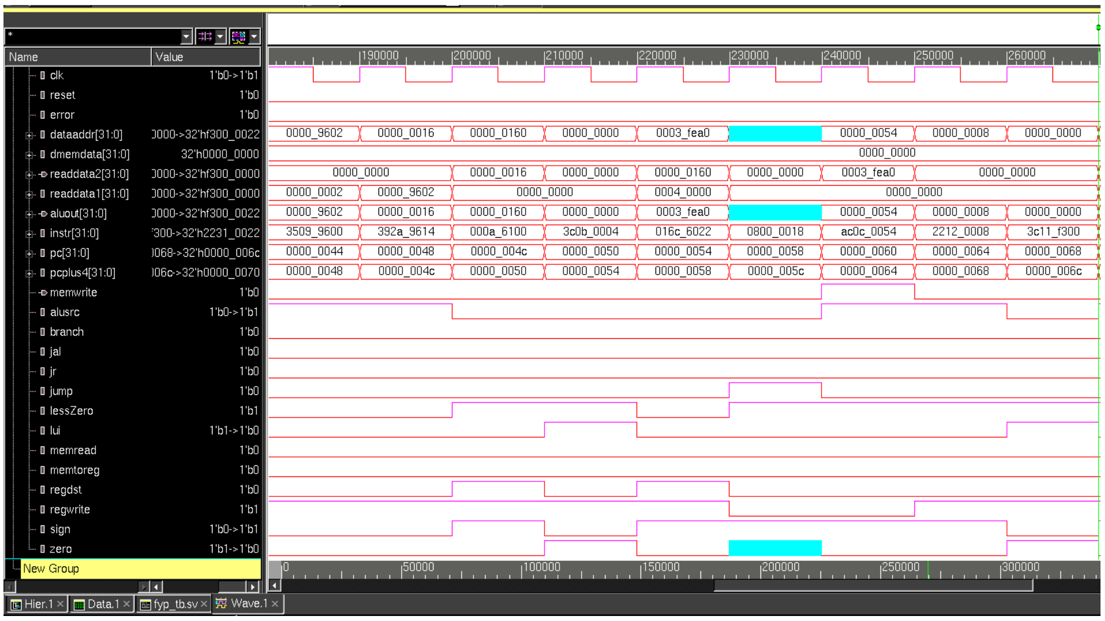


Figure 4.4: Output waveform for cycle 19 - 27 of the program

In Figure 4.4, the waveform shows ORI, XORI, SLL, LUI, SUB, and J are executed. A jump that occurs in cycle 25 shows instruction J in the last cycle is decoded correctly. In cycle 25, the final result of computation from cycles 18 to 23 is saved into data memory and verified using testbench. There is no error detected. For cycles 26 to 28, the new initial values are loaded into the register to be used for rotational operation later.

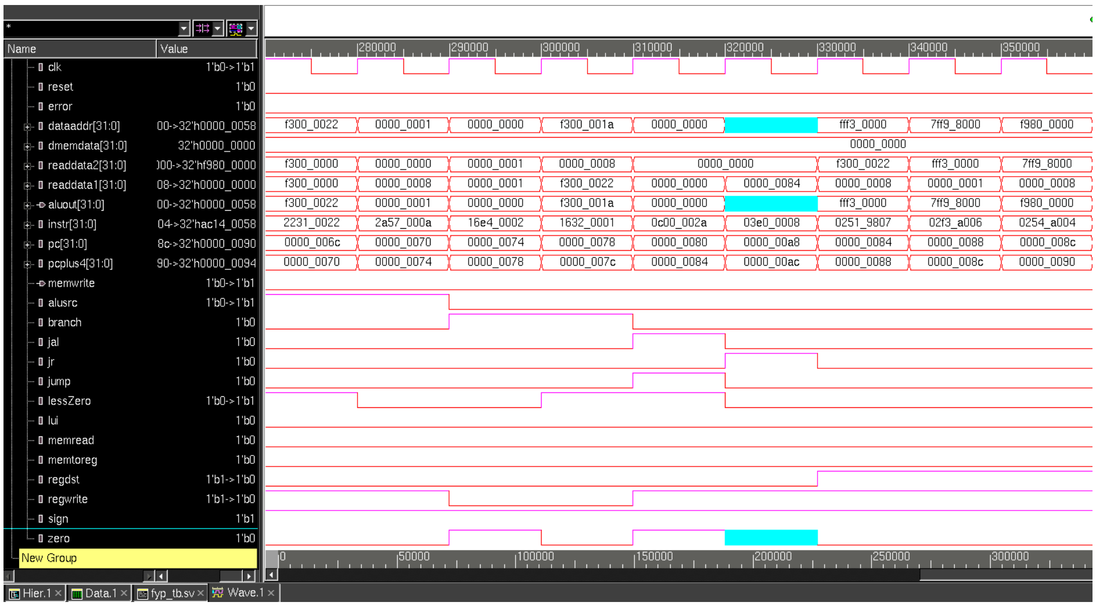
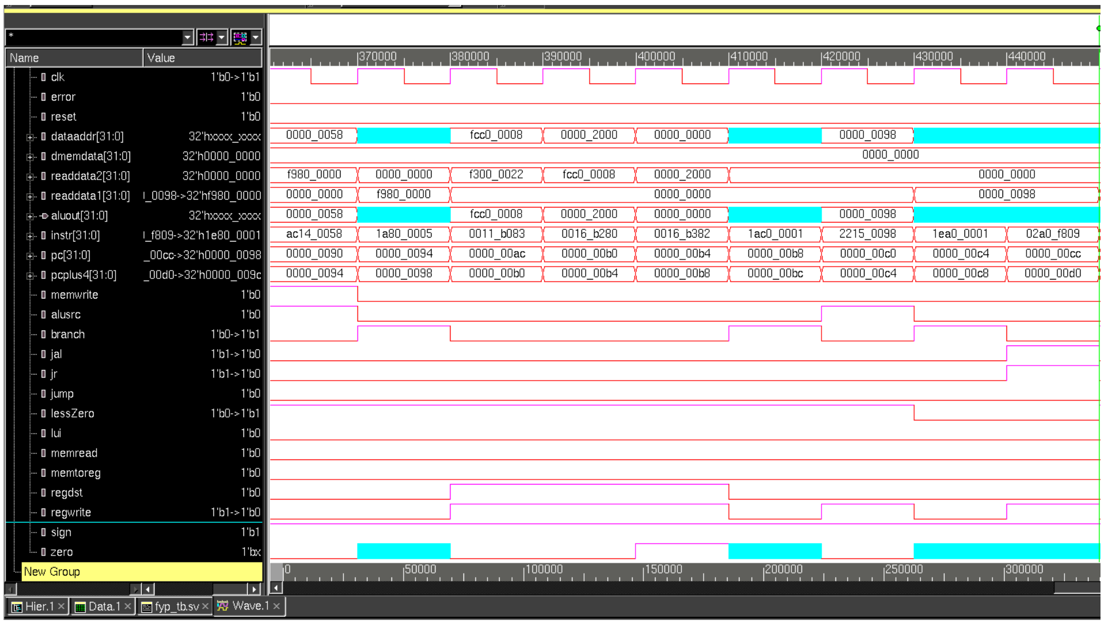


Figure 4.5: Output waveform for cycle 28 - 36 of the program

Figure 4.5 shows the simulation waveform for cycles 28 to 36. SLTI and BNE are tested in cycles 29 to 31. The result of SLTI is used in the first BNE. The correct execution of SLTI causes no branching to happen during cycle 31 since both registers contain the same value for BNE. In cycle 31, two registers with unequal values are used for BNE and branching takes place in the next cycle. Cycles 32 and 33 are to test on the JAL and JR. The JR is used to jump to the next instruction after JAL. The waveform shows the two instructions are executed correctly as the PC for cycle 34 is exactly PC+4 of cycle 32. For cycles 34 to 36, SRAV, SRLV, and SLLV are performed.



Name	Value
clk	1'b0->1'b1
error	1'b0
reset	1'b0
dataaddr[31:0]	32'hxxxx_xxxx
dmemdata[31:0]	32'h0000_0000
readdata2[31:0]	32'h0000_0000
readdata1[31:0]	ı_0098->32'hf980_0000
aluout[31:0]	32'hxxxx_xxxx
instr[31:0]	ı_f809->32'h1e80_0001
pc[31:0]	_00cc->32'h0000_0098
pcplus4[31:0]	_00d0->32'h0000_009c
memwrite	1'b0
alusrc	1'b0
branch	1'b0->1'b1
jal	1'b1->1'b0
jr	1'b1->1'b0
jump	1'b0
lessZero	1'b0->1'b1
lui	1'b0
memread	1'b0
memtoreg	1'b0
regdst	1'b0
regwrite	1'b1->1'b0
sign	1'b1
zero	1'bx

Figure 4.6: Output waveform for cycle 37 - 45 of the program

Figure 4.6 shows the storing of results from the previous cycle in the data memory in cycle 37 and the result matches the expected output. In cycle 38, the program branches to perform SRA, SLL, and SRL through the execution of BLEZ. If BLEZ does not function correctly, no branching happens, a BGTZ will be executed instead of SRA. Hence, the values save in RAM will be wrong. A value is loaded into register R5 as the address for JALR in cycle 45. This value is also used to verify the execution of BGTZ during cycle 44. If BGTZ fails to perform correctly, a LUI instruction will be used to modify the R5 values, causing JALR to jump to the wrong address. The waveform has shown the correct execution of these instructions

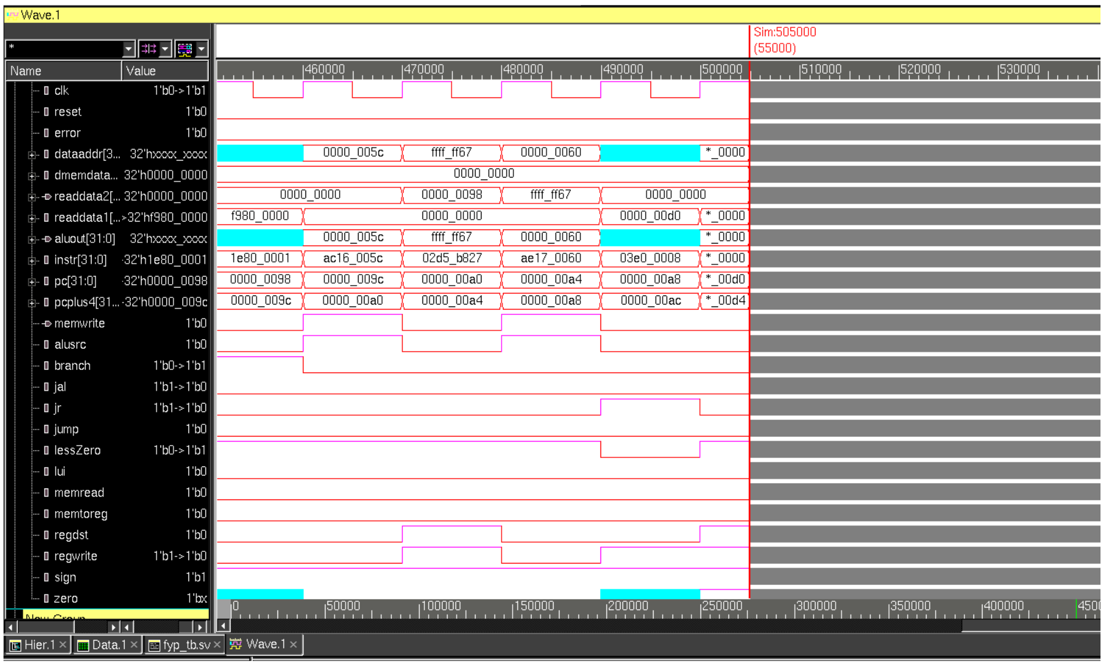


Figure 4.7: Output waveform for cycle 46 - 51 of the program

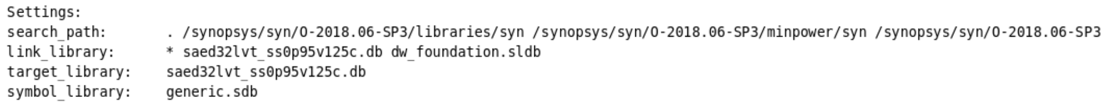
In cycle 46, BGTZ is again tested for the condition Rs are smaller than 0. AS shown in Figure 4.6, no branching is to be taken in the next cycle proving the instruction is functioning properly as it will only branch for Rs larger than Rt. The computation result of SRA, SLL, and SRL is stored in the data memory and verified to have the correct output in cycle 47. For cycles 48 and 49, NOR is executed and the result is again saved to data memory to be tested. The waveform shows there is no error detected for the NOR result. Lastly, the JR instruction is used to jump to the end of the program. Since JALR is the last instruction in the program, the JR will jump to load 0 into instruction memory as the sign for the end of the program as shown in Figure 4.6. Hence, all the 29 instructions are verified to be performed correctly by the design.

4.2. Logic Synthesis

In this stage, the verified design in RTL codes is transformed into a gate-level netlist which is the logical representation of the design. Logic synthesis is the combination of 3 main processes: translation of RTL source code into a netlist, logic optimization, and mapping of gates. The gates with 32 nm technology are used in the logic synthesis process.

4.2.1. DC Setup and Design Translation

A setup file is used to set up the library. There are three libraries: target library, link library, and synthetic library which are specified in the setup file. The target library is the library containing the cells for mapping and inferring in DC to produce a netlist that is technology specific. The link library is the cell library to be used for reference and the target library is listed in the link library list so that DC can link to the cells that are mapped in the netlist. The synthetic library refers to the standard synthetic library to be used for the implementation of the built-in HDL operators. The graphical representations of the cells from the technology library are stored in the symbol library. In this project, the saed32lvt_ss0p95v125c.db is used as the target library and listed in the link library. The synthetic library is dw_foundation.sldb while the symbol library is not specified in this project, DC is using the default library in this case which is generic.sdb to display the cells in GUI.

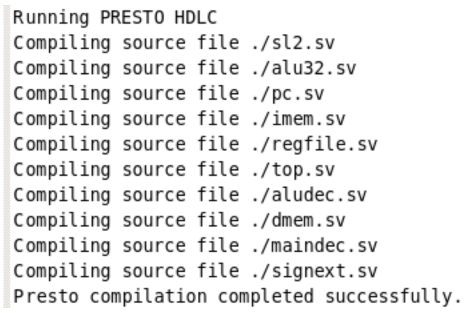


```
Settings:
search_path:      . /synopsys/syn/O-2018.06-SP3/libraries/syn /synopsys/syn/O-2018.06-SP3/minpower/syn /synopsys/syn/O-2018.06-SP3
link_library:     * saed32lvt_ss0p95v125c.db dw_foundation.sldb
target_library:   saed32lvt_ss0p95v125c.db
symbol_library:   generic.sdb
```

Figure 4.8: Libraries setup result in DC

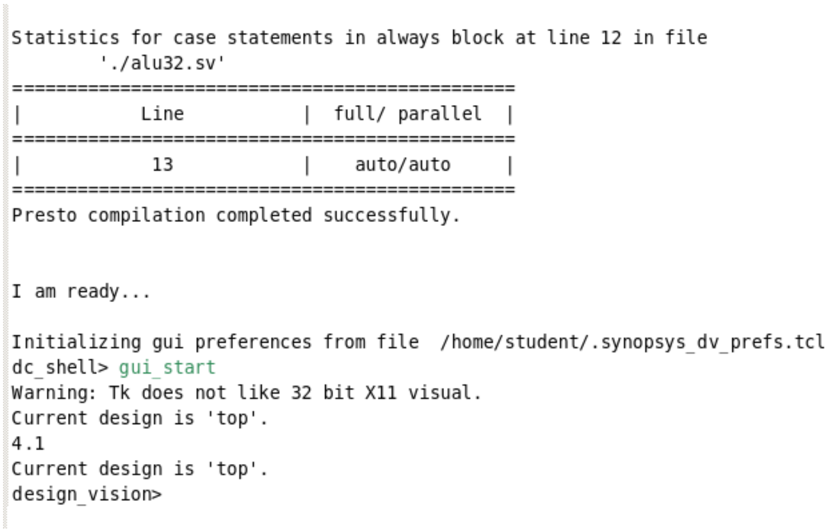
.

After setting up the libraries, the verified design in RTL codes is loaded into DC automatically as the setup file contains the command to load the design. By loading the design into DC, the RTL codes will be translated into a netlist which is unmapped and unoptimized generic technology (GTECH) netlist. The top module is set as the current design. The *analyze* command is used to read and check the design. The translation of the design is done using *elaborate* command and the command will execute *the link* command automatically for design references. Figure 4.9 and Figure 4.10 show the result of successful automated loading and translation of the design including top, sl2, alu32, pc, imem, regfile, aludec, dmem, maindec, and signext when setup DC. The top module is set as the current design after reading all the RTL codes into the tool.



```
Running PRESTO HDLC
Compiling source file ./sl2.sv
Compiling source file ./alu32.sv
Compiling source file ./pc.sv
Compiling source file ./imem.sv
Compiling source file ./regfile.sv
Compiling source file ./top.sv
Compiling source file ./aludec.sv
Compiling source file ./dmem.sv
Compiling source file ./maindec.sv
Compiling source file ./signext.sv
Presto compilation completed successfully.
```

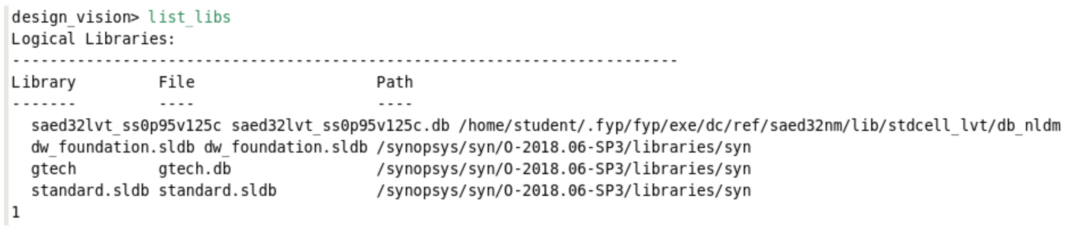
Figure 4.9: Result of loading and compilation of source codes in DC



```
Statistics for case statements in always block at line 12 in file
        './alu32.sv'
===============================================
|           Line           |  full/ parallel  |
===============================================
|            13            |    auto/auto     |
===============================================
Presto compilation completed successfully.


I am ready...

Initializing gui preferences from file  /home/student/.synopsys_dv_prefs.tcl
dc_shell> gui_start
Warning: Tk does not like 32 bit X11 visual.
Current design is 'top'.
4.1
Current design is 'top'.
design_vision>
```

Figure 4.10: Result of building and translating source codes in DC

A check on libraries setup after loading the design to ensure the desired and correct libraries are successfully loaded into the software is an essential step to avoid wrong referring to the library during the logic synthesis stage. This step is also to avoid the use of built-in libraries due to library loading errors during setup. The built-in libraries will cause failure in physical design since the libraries used in physical design do not contain data matching with built-in libraries used in DC. The *list_libs* command is used to show all the names of the libraries used in the software. Figure 4.11 shows the libraries loaded into the compiler are saed32lvt_ss095v125c.db and dw_foundation.sldb as listed in setup file.



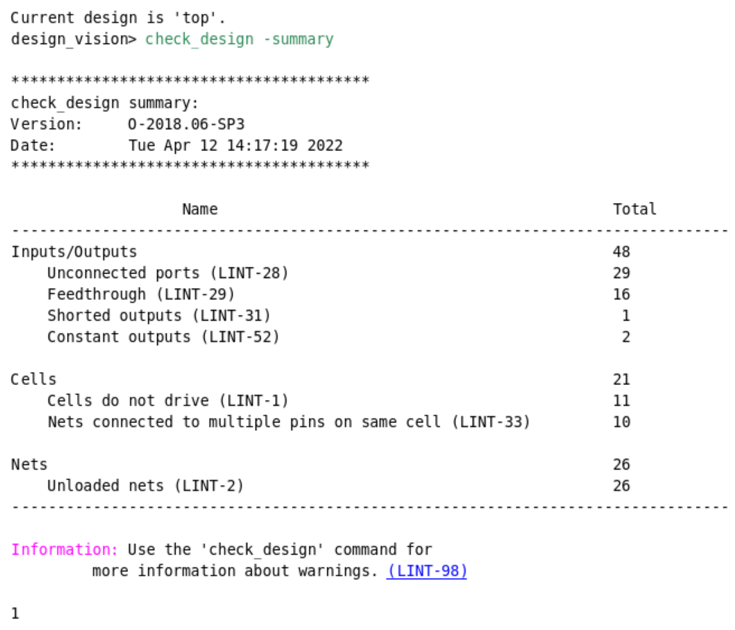
```
design_vision> list_libs
Logical Libraries:
-------------------------------------------------------------------------
Library        File                 Path
-------        ----                 ----
  saed32lvt_ss0p95v125c saed32lvt_ss0p95v125c.db /home/student/.fyp/fyp/exe/dc/ref/saed32nm/lib/stdcell_lvt/db_nldm
  dw_foundation.sldb dw_foundation.sldb /synopsys/syn/O-2018.06-SP3/libraries/syn
  gtech        gtech.db             /synopsys/syn/O-2018.06-SP3/libraries/syn
  standard.sldb standard.sldb       /synopsys/syn/O-2018.06-SP3/libraries/syn
1
```

Figure 4.11: Result of listing libraries used in DC

4.2.2. Design Checking before mapping and optimization

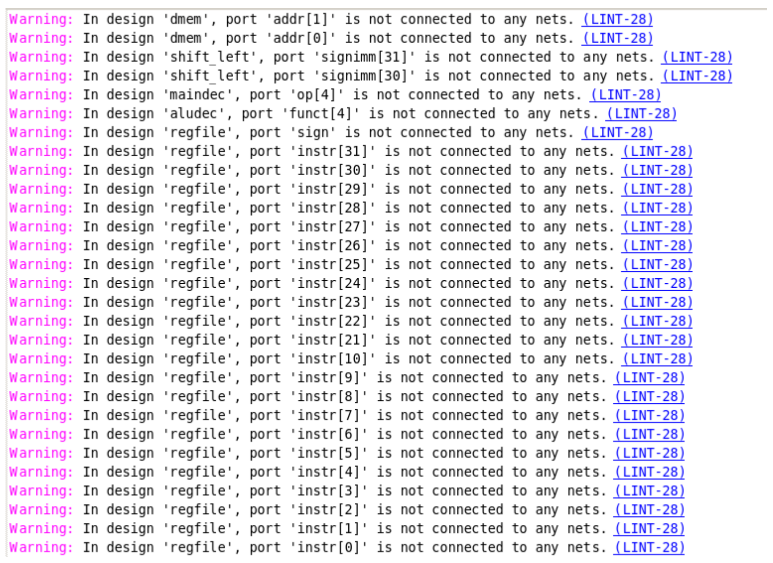
It is vital to check the design loaded into the software before proceeding to the next stage. This is to check on the hierarchy issues along with the connectivity of the design. The summary of the check design result is shown in Figure 4.12.

There are 65 unconnected ports in the design. The detail of the warning for unconnected ports is generated as shown in Figure 4.13. After checking affected ports with the RTL codes, these ports are described intentionally to be unused in the specific module. Thus, these unconnected ports are removed in DC to avoid potential design issues during optimization. There are 2 ports to have constant outputs. This warning is ignored as the *shifted [0]* and *shifted [1]* are both purposely connected to logic 0 in the source code to shift the immediate value to left by 2. Since the two ports are connected to logic 0, they are shorted together as shown in Figure 4.14. This warning will be solved together with feedthrough and net connected to multiple pins on the same cell using command during the optimization process. Feedthroughs refer to the connection between multiple ports to connect an input to an output using nets. These warnings can be solved by inserting buffer into the nets during mapping and optimization. While for the unloaded nets, they are the nets *pc [1:0]*, and *pc [31:8]* in the top module. Figure 4.15 shows all the nets with no load in the design before mapping. Since only *pc [7:2]* is connected to the imem module to obtain the address of instruction memory, the rest of the signals in *pc* net are unused. Thus, the warning is ignored as the nets are designed to be unloaded. The tools will load all the output ports when optimizing the design. The cells that drive nothing will be removed by the software during compilation.



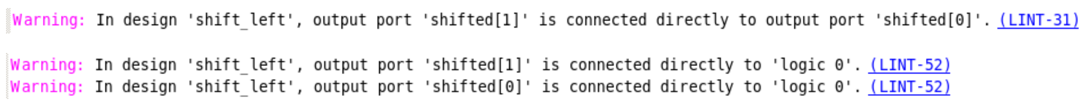
```
Current design is 'top'.
design_vision> check_design -summary

****************************************
check_design summary:
Version:     O-2018.06-SP3
Date:        Tue Apr 12 14:17:19 2022
****************************************

                   Name                                            Total
--------------------------------------------------------------------------------
Inputs/Outputs                                                     48
    Unconnected ports (LINT-28)                                    29
    Feedthrough (LINT-29)                                          16
    Shorted outputs (LINT-31)                                       1
    Constant outputs (LINT-52)                                      2

Cells                                                              21
    Cells do not drive (LINT-1)                                    11
    Nets connected to multiple pins on same cell (LINT-33)         10

Nets                                                               26
    Unloaded nets (LINT-2)                                         26
--------------------------------------------------------------------------------

Information: Use the 'check_design' command for
         more information about warnings. (LINT-98)

1
```

Figure 4.12: Summary of check design before mapping.



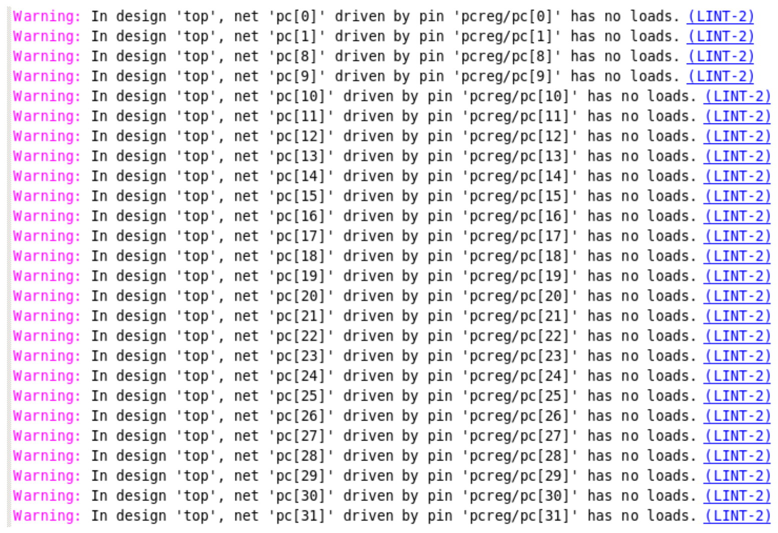
```
Warning: In design 'dmem', port 'addr[1]' is not connected to any nets. (LINT-28)
Warning: In design 'dmem', port 'addr[0]' is not connected to any nets. (LINT-28)
Warning: In design 'shift_left', port 'signimm[31]' is not connected to any nets. (LINT-28)
Warning: In design 'shift_left', port 'signimm[30]' is not connected to any nets. (LINT-28)
Warning: In design 'maindec', port 'op[4]' is not connected to any nets. (LINT-28)
Warning: In design 'aludec', port 'funct[4]' is not connected to any nets. (LINT-28)
Warning: In design 'regfile', port 'sign' is not connected to any nets. (LINT-28)
Warning: In design 'regfile', port 'instr[31]' is not connected to any nets. (LINT-28)
Warning: In design 'regfile', port 'instr[30]' is not connected to any nets. (LINT-28)
Warning: In design 'regfile', port 'instr[29]' is not connected to any nets. (LINT-28)
Warning: In design 'regfile', port 'instr[28]' is not connected to any nets. (LINT-28)
Warning: In design 'regfile', port 'instr[27]' is not connected to any nets. (LINT-28)
Warning: In design 'regfile', port 'instr[26]' is not connected to any nets. (LINT-28)
Warning: In design 'regfile', port 'instr[25]' is not connected to any nets. (LINT-28)
Warning: In design 'regfile', port 'instr[24]' is not connected to any nets. (LINT-28)
Warning: In design 'regfile', port 'instr[23]' is not connected to any nets. (LINT-28)
Warning: In design 'regfile', port 'instr[22]' is not connected to any nets. (LINT-28)
Warning: In design 'regfile', port 'instr[21]' is not connected to any nets. (LINT-28)
Warning: In design 'regfile', port 'instr[10]' is not connected to any nets. (LINT-28)
Warning: In design 'regfile', port 'instr[9]' is not connected to any nets. (LINT-28)
Warning: In design 'regfile', port 'instr[8]' is not connected to any nets. (LINT-28)
Warning: In design 'regfile', port 'instr[7]' is not connected to any nets. (LINT-28)
Warning: In design 'regfile', port 'instr[6]' is not connected to any nets. (LINT-28)
Warning: In design 'regfile', port 'instr[5]' is not connected to any nets. (LINT-28)
Warning: In design 'regfile', port 'instr[4]' is not connected to any nets. (LINT-28)
Warning: In design 'regfile', port 'instr[3]' is not connected to any nets. (LINT-28)
Warning: In design 'regfile', port 'instr[2]' is not connected to any nets. (LINT-28)
Warning: In design 'regfile', port 'instr[1]' is not connected to any nets. (LINT-28)
Warning: In design 'regfile', port 'instr[0]' is not connected to any nets. (LINT-28)
```

Figure 4.13: Detail of unconnected ports before mapping



```
Warning: In design 'shift_left', output port 'shifted[1]' is connected directly to output port 'shifted[0]'. (LINT-31)

Warning: In design 'shift_left', output port 'shifted[1]' is connected directly to 'logic 0'. (LINT-52)
Warning: In design 'shift_left', output port 'shifted[0]' is connected directly to 'logic 0'. (LINT-52)
```

Figure 4.14: Detail of constant and shorted outputs

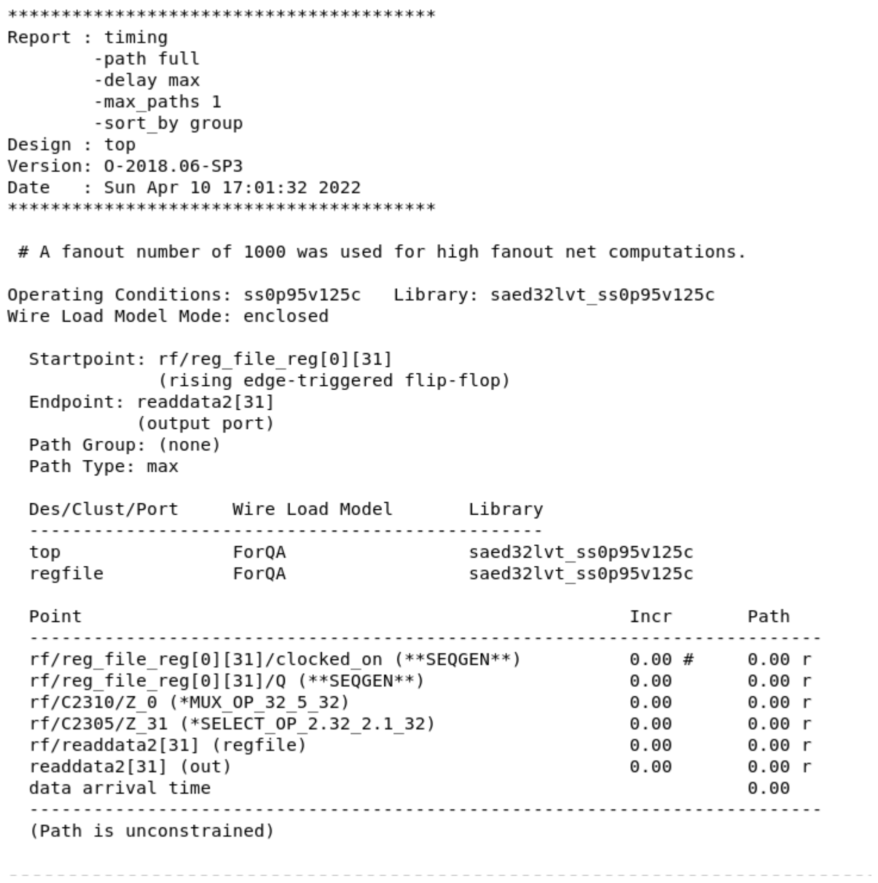


```
Warning: In design 'top', net 'pc[0]' driven by pin 'pcreg/pc[0]' has no loads. (LINT-2)
Warning: In design 'top', net 'pc[1]' driven by pin 'pcreg/pc[1]' has no loads. (LINT-2)
Warning: In design 'top', net 'pc[8]' driven by pin 'pcreg/pc[8]' has no loads. (LINT-2)
Warning: In design 'top', net 'pc[9]' driven by pin 'pcreg/pc[9]' has no loads. (LINT-2)
Warning: In design 'top', net 'pc[10]' driven by pin 'pcreg/pc[10]' has no loads. (LINT-2)
Warning: In design 'top', net 'pc[11]' driven by pin 'pcreg/pc[11]' has no loads. (LINT-2)
Warning: In design 'top', net 'pc[12]' driven by pin 'pcreg/pc[12]' has no loads. (LINT-2)
Warning: In design 'top', net 'pc[13]' driven by pin 'pcreg/pc[13]' has no loads. (LINT-2)
Warning: In design 'top', net 'pc[14]' driven by pin 'pcreg/pc[14]' has no loads. (LINT-2)
Warning: In design 'top', net 'pc[15]' driven by pin 'pcreg/pc[15]' has no loads. (LINT-2)
Warning: In design 'top', net 'pc[16]' driven by pin 'pcreg/pc[16]' has no loads. (LINT-2)
Warning: In design 'top', net 'pc[17]' driven by pin 'pcreg/pc[17]' has no loads. (LINT-2)
Warning: In design 'top', net 'pc[18]' driven by pin 'pcreg/pc[18]' has no loads. (LINT-2)
Warning: In design 'top', net 'pc[19]' driven by pin 'pcreg/pc[19]' has no loads. (LINT-2)
Warning: In design 'top', net 'pc[20]' driven by pin 'pcreg/pc[20]' has no loads. (LINT-2)
Warning: In design 'top', net 'pc[21]' driven by pin 'pcreg/pc[21]' has no loads. (LINT-2)
Warning: In design 'top', net 'pc[22]' driven by pin 'pcreg/pc[22]' has no loads. (LINT-2)
Warning: In design 'top', net 'pc[23]' driven by pin 'pcreg/pc[23]' has no loads. (LINT-2)
Warning: In design 'top', net 'pc[24]' driven by pin 'pcreg/pc[24]' has no loads. (LINT-2)
Warning: In design 'top', net 'pc[25]' driven by pin 'pcreg/pc[25]' has no loads. (LINT-2)
Warning: In design 'top', net 'pc[26]' driven by pin 'pcreg/pc[26]' has no loads. (LINT-2)
Warning: In design 'top', net 'pc[27]' driven by pin 'pcreg/pc[27]' has no loads. (LINT-2)
Warning: In design 'top', net 'pc[28]' driven by pin 'pcreg/pc[28]' has no loads. (LINT-2)
Warning: In design 'top', net 'pc[29]' driven by pin 'pcreg/pc[29]' has no loads. (LINT-2)
Warning: In design 'top', net 'pc[30]' driven by pin 'pcreg/pc[30]' has no loads. (LINT-2)
Warning: In design 'top', net 'pc[31]' driven by pin 'pcreg/pc[31]' has no loads. (LINT-2)
```

Figure 4.15: Detail of unloaded nets warning

4.2.3. Analysis of Timing, Area and Power before Mapping

By using the built-in static timing analyser from DC, a timing report is generated before optimization. Figure 4.16 shows the timing report of the design before any constraints is applied to the design. The total data arrival time is 0 ns before the optimization. It is because the design is currently a generic technology (GTECH) netlist which is technology independent. The GTECH components are built after loading the design into DC to act as the representations for the function of Boolean and they are unmapped (Bhatnagar, 2002). There is no timing information in the generic library. Thus, the longest maximum delay obtained by the tool before mapping is 0 ns. After applying constraints to the design and optimization, the tool will be able to calculate the data arrival time required by the design to meet the timing requirements.



```
****************************************
Report : timing
        -path full
        -delay max
        -max_paths 1
        -sort_by group
Design : top
Version: O-2018.06-SP3
Date   : Sun Apr 10 17:01:32 2022
****************************************

 # A fanout number of 1000 was used for high fanout net computations.

Operating Conditions: ss0p95v125c   Library: saed32lvt_ss0p95v125c
Wire Load Model Mode: enclosed

  Startpoint: rf/reg_file_reg[0][31]
              (rising edge-triggered flip-flop)
  Endpoint: readdata2[31]
            (output port)
  Path Group: (none)
  Path Type: max

  Des/Clust/Port     Wire Load Model       Library
  ------------------------------------------------
  top                ForQA                 saed32lvt_ss0p95v125c
  regfile            ForQA                 saed32lvt_ss0p95v125c

  Point                                                   Incr       Path
  --------------------------------------------------------------------------
  rf/reg_file_reg[0][31]/clocked_on (**SEQGEN**)          0.00 #     0.00 r
  rf/reg_file_reg[0][31]/Q (**SEQGEN**)                   0.00       0.00 r
  rf/C2310/Z_0 (*MUX_OP_32_5_32)                          0.00       0.00 r
  rf/C2305/Z_31 (*SELECT_OP_2.32_2.1_32)                  0.00       0.00 r
  rf/readdata2[31] (regfile)                              0.00       0.00 r
  readdata2[31] (out)                                     0.00       0.00 r
  data arrival time                                                  0.00
  --------------------------------------------------------------------------
  (Path is unconstrained)
```

Figure 4.16: Timing report before mapping

The statistics and information of the design area before the optimization are generated as shown in Figure 4.17. The report shows the non-combinational area, combinational area, total area as well as the number of nets, ports, and cells. The library used to generate area report before mapping is gtech.db. Thus, the estimation of the area is based on the generic technology. Based on the report, there are 3157 ports, 14690 nets, 5750 non-combinational cells, 3260 sequential cells, and 2427 combinational cells that exist in the design. Besides, there are 477 inverters or buffers in the design. The area report before mapping is only able to calculate the net interconnect area of 12554. 583485 µm^2. The generic technology library does not have the information on cell area. Thus, the tool calculated 0 µm^2 for the cell area before the optimization process. After the design is mapped, the tool will be able to obtain area information from the target library and calculate the cell areas.

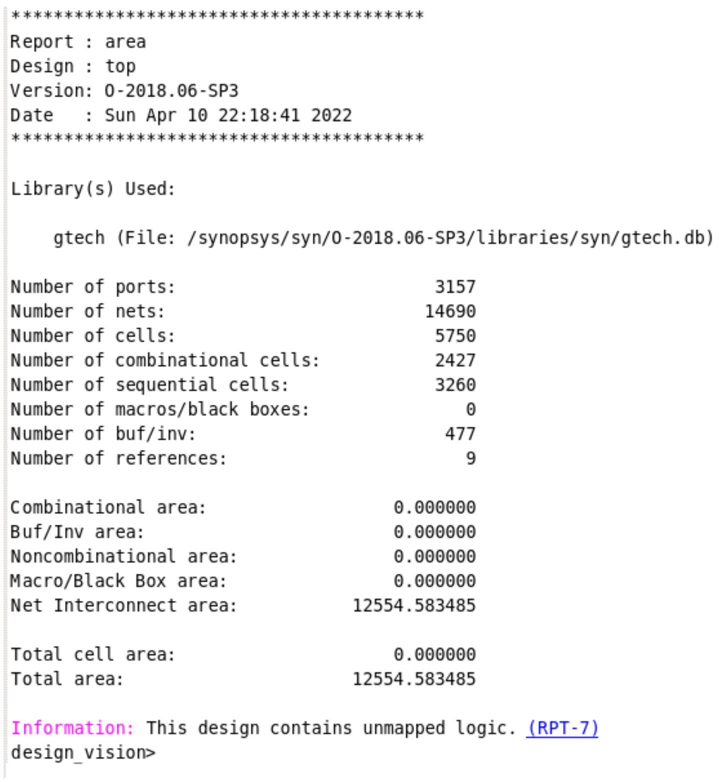
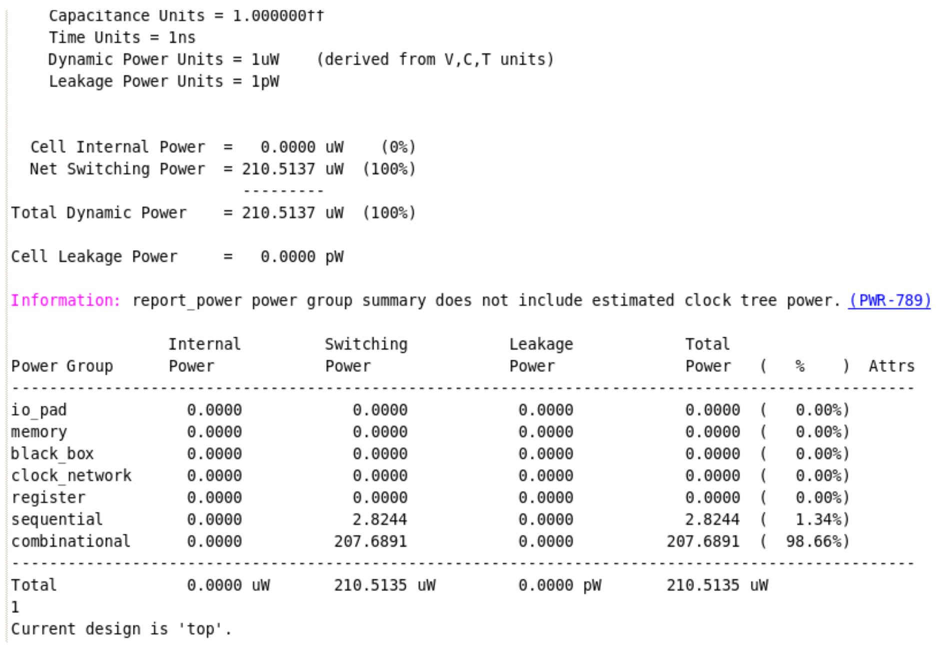

```
****************************************
Report : area
Design : top
Version: O-2018.06-SP3
Date   : Sun Apr 10 22:18:41 2022
****************************************

Library(s) Used:

    gtech (File: /synopsys/syn/O-2018.06-SP3/libraries/syn/gtech.db)

Number of ports:                         3157
Number of nets:                         14690
Number of cells:                         5750
Number of combinational cells:           2427
Number of sequential cells:              3260
Number of macros/black boxes:               0
Number of buf/inv:                        477
Number of references:                       9

Combinational area:                  0.000000
Buf/Inv area:                        0.000000
Noncombinational area:               0.000000
Macro/Black Box area:                0.000000
Net Interconnect area:           12554.583485

Total cell area:                     0.000000
Total area:                      12554.583485

Information: This design contains unmapped logic. (RPT-7)
design_vision>
```

Figure 4.17: Area report before mapping

Power analysis of the unmapped design is generated by the tool. The DC provides the power information including net switching power, cell internal power, cell leakage power, and the sum of dynamic power. The report of power analysis before the compilation is shown in Figure 4.18. From the report, the tool calculated a cell internal power of 0 µW and a net switching power of 210.5137 µW. The leakage power is 0 µW, causing the final total power consumption of the unmapped design to be 210.5137 µW. As mentioned before, the unmapped design is the GTECH components. Since they are technology independent, there is no power information for the tool to calculate the cell internal power and cell leakage power. Thus, both of the power parameters are 0 µW before the mapping process. After the mapping process, the logic cells will be mapped to specific technology which is 32 nm in this case and the target library contains the power information of the cells. Hence, a complete power report with net switching power, cell internal power, cell leakage power, and the sum of dynamic power calculated can be generated after the optimization.



```
    Capacitance Units = 1.000000ff
    Time Units = 1ns
    Dynamic Power Units = 1uW    (derived from V,C,T units)
    Leakage Power Units = 1pW


  Cell Internal Power  =   0.0000 uW    (0%)
  Net Switching Power  = 210.5137 uW  (100%)
                         ---------
Total Dynamic Power    = 210.5137 uW  (100%)

Cell Leakage Power     =   0.0000 pW

Information: report_power power group summary does not include estimated clock tree power. (PWR-789)

                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )  Attrs
--------------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000            0.0000             0.0000  (   0.00%)
memory             0.0000           0.0000            0.0000             0.0000  (   0.00%)
black_box          0.0000           0.0000            0.0000             0.0000  (   0.00%)
clock_network      0.0000           0.0000            0.0000             0.0000  (   0.00%)
register           0.0000           0.0000            0.0000             0.0000  (   0.00%)
sequential         0.0000           2.8244            0.0000             2.8244  (   1.34%)
combinational      0.0000         207.6891            0.0000           207.6891  (  98.66%)
--------------------------------------------------------------------------------------------------
Total              0.0000 uW      210.5135 uW         0.0000 pW        210.5135 uW
1
Current design is 'top'.
```

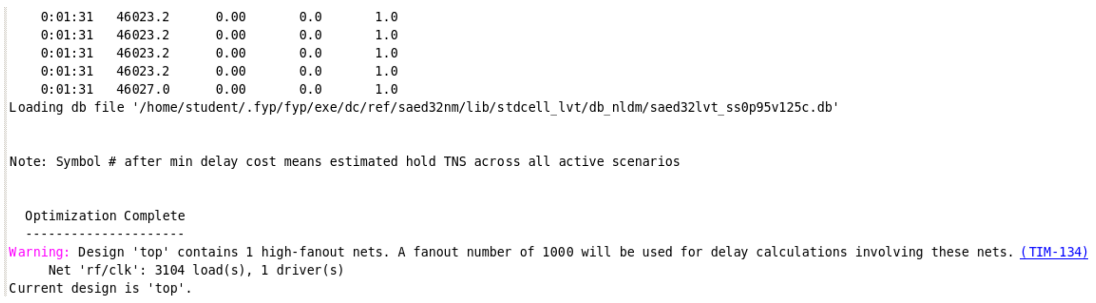
Figure 4.18: Power analysis before mapping

4.2.4. Mapping and Optimization of Design

After checking the design and ensuring all the connections are correct as desired, the next step is carried out to map and optimize the design. For Synopsys DC, the mapping and optimization are carried out based on the constraints applied to the design. The constraints are decided by the designer for the design to meet the requirements. The tool is used to automate the process of mapping and optimize the design using the algorithm. The design constraints are used to set the period for the clock signal, setup time, hold time, the load applied to the design, and other limitations.

The target library is used to provide information on the logic gates for mapping. A gate-level netlist that matches the constraints applied to the design is generated using the logic gates from the library. At the same time, the setup violation is fixed. For hold time violation, it will be fixed in the physical design stage using ICC. The tool modifies the nets, cells, and ports of the design to meet the design requirement. It is important to set some necessary net, cells, or ports as don’t touch or ideal to avoid modification during optimization which hence affects the functionality of the design. In this project, the clock and reset signals are set as an ideal net to prevent the tool from changing these ports when optimizing the design.

Since the tool can perform the mapping and optimization process, the command can be used to instruct the execution of the process. The tool provided several options for the user to select the suitable algorithm for optimization. The optimization modes including map effort, area effort, and power effort are set to high. Thus, the tool will put more CPU time during mapping, area recovery, and power optimization during compilation. Figure 4.19 shows the mapping and optimization are completed successfully and a gate-level netlist is generated.



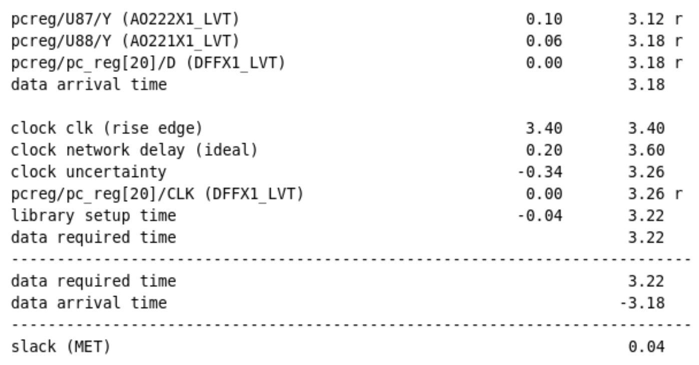
```
    0:01:31   46023.2      0.00       0.0       1.0
    0:01:31   46023.2      0.00       0.0       1.0
    0:01:31   46023.2      0.00       0.0       1.0
    0:01:31   46023.2      0.00       0.0       1.0
    0:01:31   46027.0      0.00       0.0       1.0
Loading db file '/home/student/.fyp/fyp/exe/dc/ref/saed32nm/lib/stdcell_lvt/db_nldm/saed32lvt_ss0p95v125c.db'


Note: Symbol # after min delay cost means estimated hold TNS across all active scenarios


  Optimization Complete
  ---------------------
Warning: Design 'top' contains 1 high-fanout nets. A fanout number of 1000 will be used for delay calculations involving these nets. (TIM-134)
     Net 'rf/clk': 3104 load(s), 1 driver(s)
Current design is 'top'.
```

Figure 4.19: Execution and result of mapping and optimization process

4.2.5. Analysis of Timing, Area and Power after Mapping and Optimization

The mapped and optimized design is analyzed in terms of timing, area, and power to ensure the design is meeting the design requirements. The constraints applied to the design need to be modified and reapply to the unmapped design to repeat compilation if the design fails to meet the requirements.

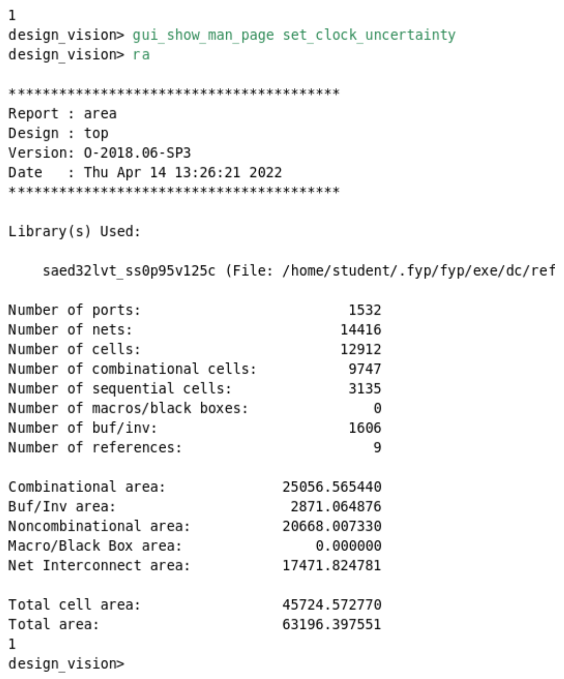
One of the most important criteria is the optimized design must be clear with the setup violations. The timing report of the optimized design is generated using the built-in static timing analyser from the tool. The tool provides automation for the calculations of data arrival time and data required time. The path with the slowest maximum delay is listed in the timing report with the detail of increment for every delay. If the arrival time exceeds the required time, a setup violation occurs. In other words, the slack must be equal to or greater than zero in order to avoid setup violation.



```
pcreg/U87/Y (AO222X1_LVT)                      0.10       3.12 r
pcreg/U88/Y (AO221X1_LVT)                      0.06       3.18 r
pcreg/pc_reg[20]/D (DFFX1_LVT)                 0.00       3.18 r
data arrival time                                         3.18

clock clk (rise edge)                          3.40       3.40
clock network delay (ideal)                    0.20       3.60
clock uncertainty                             -0.34       3.26
pcreg/pc_reg[20]/CLK (DFFX1_LVT)               0.00       3.26 r
library setup time                            -0.04       3.22
data required time                                        3.22
---------------------------------------------------------------------
data required time                                        3.22
data arrival time                                        -3.18
---------------------------------------------------------------------
slack (MET)                                               0.04
```

Figure 4.20: Result of timing report after mapping

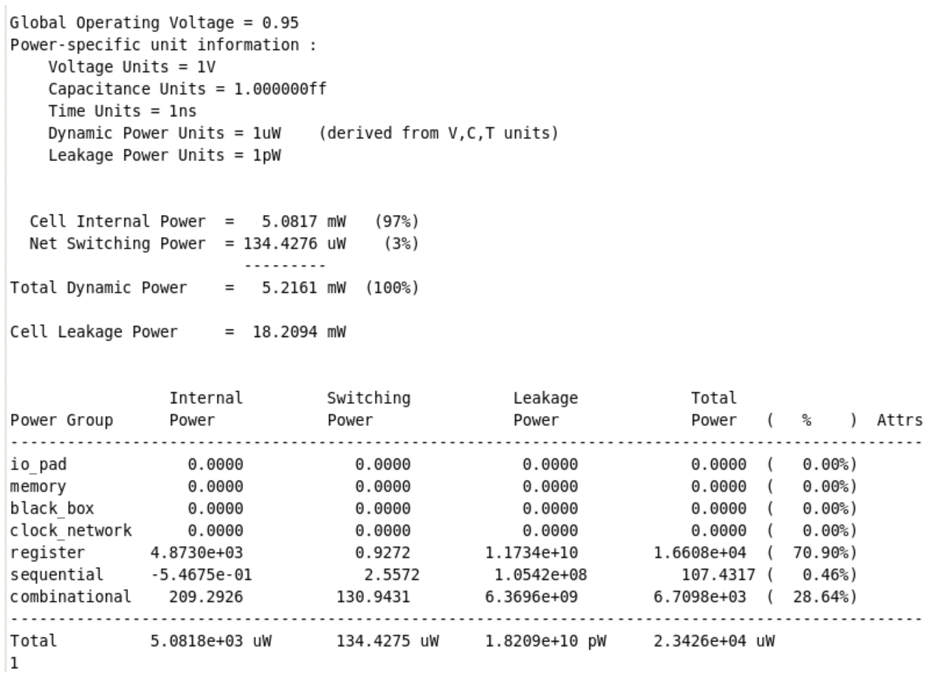
The result of the timing report is shown in Figure 4.20. The frequency of the clock applied to the design is about 294 MHz which is equivalent to a clock period of 3.4 ns. For data arrival time, it is obtained by adding the clock network delay and the delays occur at the input of the sequential element receiving the signals along the complete path. Hence, the data arrival time for the slowest maximum delay path is 3.18 ns. The required time for setup is calculated by summing the delay of the clock network, clock period, and setup time followed by the subtraction time required for library setup and the setup uncertainty. In this project, the setup uncertainty and hold uncertainty are set as 10% of the lock period or 0.34 ns. Thus, the total required time is 3.22 ns. The result shows the optimized design has a positive slack with a margin of 0.04 ns. Hence, there is no setup violation.



```
1
design_vision> gui_show_man_page set_clock_uncertainty
design_vision> ra

****************************************
Report : area
Design : top
Version: O-2018.06-SP3
Date   : Thu Apr 14 13:26:21 2022
****************************************

Library(s) Used:

    saed32lvt_ss0p95v125c (File: /home/student/.fyp/fyp/exe/dc/ref

Number of ports:                         1532
Number of nets:                         14416
Number of cells:                        12912
Number of combinational cells:           9747
Number of sequential cells:              3135
Number of macros/black boxes:               0
Number of buf/inv:                       1606
Number of references:                       9

Combinational area:             25056.565440
Buf/Inv area:                    2871.064876
Noncombinational area:          20668.007330
Macro/Black Box area:               0.000000
Net Interconnect area:          17471.824781

Total cell area:                45724.572770
Total area:                     63196.397551
1
design_vision>
```

Figure 4.21: Area report after mapping

Besides, the report to analysing on the chip area is generated using the tool. The area report of the optimized design is shown in Figure 4.21. The total area of the optimized design is 63196.397551 μm^2. Since the unmapped design only contains area information for the net interconnects area, the mapped design is found to have a larger area as the area for cells is calculated after mapping. There are no macro cells in the design. The number of ports, nets, and sequential cells reduces while the number of non-combinational cells, combinational cells, and buffer or inverter grow after optimization. The tools had modified the elements in the design to meet the constraints applied to it. Thus, the number of ports, nets non-combinational cells, combinational cells, sequential cells, and buffer or inverter changes after the design is optimized. In this project, the constraint of a maximum area equal to 0 μm^2 is applied to the design to optimize the area during the mapping process. To further ensure the smallest possible area is generated during optimization, the effort to optimize the design area is set to high so that the tool spends more CPU time at the area recovery stage. Thus, a design area of 63196.397551 µm^2 is the minimum area the tool can generate at the same time promising the design is meeting the requirements.



```
Global Operating Voltage = 0.95
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000ff
    Time Units = 1ns
    Dynamic Power Units = 1uW    (derived from V,C,T units)
    Leakage Power Units = 1pW


  Cell Internal Power  =   5.0817 mW   (97%)
  Net Switching Power  = 134.4276 uW    (3%)
                         ---------
Total Dynamic Power    =   5.2161 mW  (100%)

Cell Leakage Power     =  18.2094 mW


                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )  Attrs
--------------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000            0.0000            0.0000  (   0.00%)
memory             0.0000           0.0000            0.0000            0.0000  (   0.00%)
black_box          0.0000           0.0000            0.0000            0.0000  (   0.00%)
clock_network      0.0000           0.0000            0.0000            0.0000  (   0.00%)
register        4.8730e+03          0.9272        1.1734e+10        1.6608e+04  (  70.90%)
sequential     -5.4675e-01           2.5572        1.0542e+08          107.4317 (   0.46%)
combinational    209.2926         130.9431        6.3696e+09        6.7098e+03  (  28.64%)
--------------------------------------------------------------------------------------------------
Total           5.0818e+03 uW     134.4275 uW     1.8209e+10 pW     2.3426e+04 uW
1
```

Figure 4.22: Power report after mapping

Aside from timing and area, power consumption is another important parameter to analyze a chip's performance. DC supports automated power analysis, and a power report is generated as the result of the analysis. The result of the power report for the optimized design is shown in Figure 4.22. The design dissipated power with a total of 23.426 mW. The power due to cell leakage is 18.2094 mW and it contributes to 64.93 % of the total power dissipation.

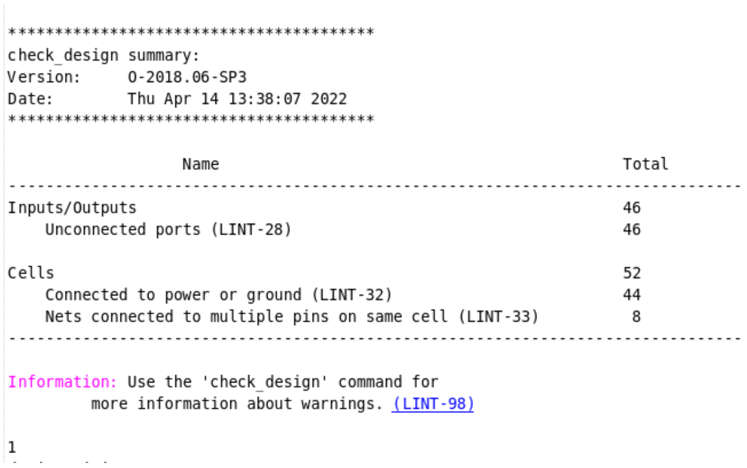
The remaining power consumption is due to the dynamic power which is equivalent to 5.2161 mW. Dynamic power is the sum of net switching power and cell internal power. The power consumption due to the discharging or changing of capacitance at the cell output ports is known as switching power while the internal power refers to the power dissipation inside a cell. The reduction in net number after optimization had caused the net switching power to drop by 47.47 % from 255.9179 µW to 134.4276 µW. The internal power consumed by the cell is 5.0817 mW. It makes up 97% of the total dynamic power. The register power group is contributing the most to the cell's internal power and the leakage power. The register power group dissipated 4.873 mW for internal power. At the same time, there is 11.734 mW of power leakage due to the registers group. Since the RAM and ROM are included in the design for logic synthesis, the use of a register in the design is plenty. Thus, the register group dissipated 70.90 % of the total power. The tool is instructed to apply high effort to optimize power during compilation. Hence, a power consumption of 23.426 mW is the minimum power the tool can achieve without violating the design requirement

4.2.6. Design Checking after Mapping and Optimization

After compilation, the design is checked again for its connectivity. A design check can avoid the change in design functionality due to the modifications made by the tool in order to meet the design requirements. Actions should be taken to clear the warning when it affects the design functionality. Figure 4.23 shows the summary of the check design result after the mapping and optimization process. There are 46 unconnected ports in the optimized design. The unconnected ports are the input ports of adder, subtractor, and comparators connected to the ground as well as some unused output ports such as the CO of an adder. Since these ports are unused, they are removed manually using a command.

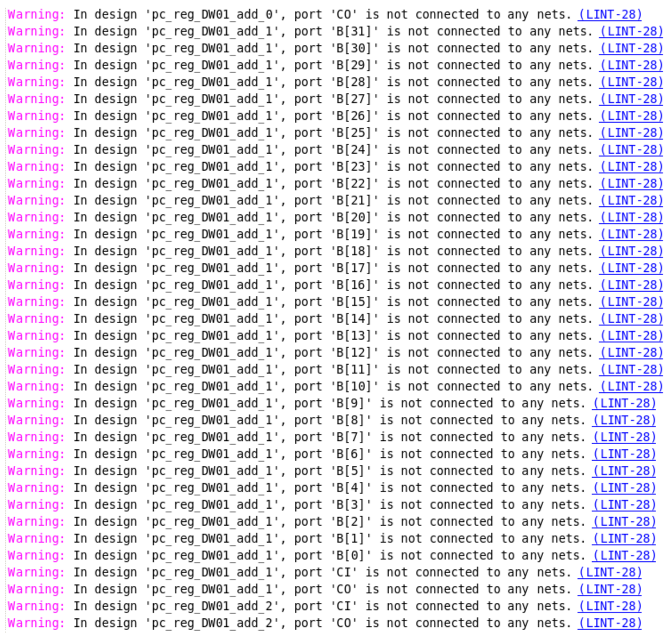
The optimized design is checked again after removing the unused ports. The new result in Figure 4.26 shows there are 9 cells that exist with warnings in which there are 4 cells are found to be connected to logic 0 or logic 1 and 5 of them are connected to several pins of one cell. Thus, the detail of the warnings is checked to verify the connectivity of the reported cells and nets. Based on the detail of cells to be connected to ground or power, the pin *shifted [1]* and *shifted [0]* in the submodule *pcreg* and *rf* are connected to the ground. Since the *shifted [1]* and *shifted [0]* are designed intentionally to always be logic 0 as the result of shifting the immediate value to left by 2, these warnings can be ignored. While for the nets connected to several pins on the same cells, the listed cells in the warning detail are designed to be the same signal in the RTL code. Thus, the warning can be ignored to ensure the proper functioning of the design after mapping and optimization. The unloaded nets show the affected nets are connected to ground or supply voltage pins as shown in Figure 4.28 and Figure 4.29. The unconnected nets are due to the removal of unused ports connected to logic 0. Since the nets are not connected to any ports, the functionality and connectivity of the design will not be affected. Hence, these warnings can be ignored with no action needed to fix them.

From the result of check design after mapping, the problems such as feedthroughs, unconnected nets, and cells do not drive are solved. The inserting of the buffer using the command can solve the feedthroughs that exist in the nets. During the optimization, the unconnected nets are removed. The tool is applied with a setting to remove the unloaded sequential cells during optimization. Thus, the cells without driving any nets are removed by the tool.



```
****************************************
check_design summary:
Version:     O-2018.06-SP3
Date:        Thu Apr 14 13:38:07 2022
****************************************

                   Name                                              Total
--------------------------------------------------------------------------------
Inputs/Outputs                                                       46
    Unconnected ports (LINT-28)                                      46

Cells                                                                52
    Connected to power or ground (LINT-32)                           44
    Nets connected to multiple pins on same cell (LINT-33)            8
--------------------------------------------------------------------------------

Information: Use the 'check_design' command for
         more information about warnings. (LINT-98)

1
```

Figure 4.23: Unconnected ports in optimised design



```
Warning: In design 'pc_reg_DW01_add_0', port 'CO' is not connected to any nets. (LINT-28)
Warning: In design 'pc_reg_DW01_add_1', port 'B[31]' is not connected to any nets. (LINT-28)
Warning: In design 'pc_reg_DW01_add_1', port 'B[30]' is not connected to any nets. (LINT-28)
Warning: In design 'pc_reg_DW01_add_1', port 'B[29]' is not connected to any nets. (LINT-28)
Warning: In design 'pc_reg_DW01_add_1', port 'B[28]' is not connected to any nets. (LINT-28)
Warning: In design 'pc_reg_DW01_add_1', port 'B[27]' is not connected to any nets. (LINT-28)
Warning: In design 'pc_reg_DW01_add_1', port 'B[26]' is not connected to any nets. (LINT-28)
Warning: In design 'pc_reg_DW01_add_1', port 'B[25]' is not connected to any nets. (LINT-28)
Warning: In design 'pc_reg_DW01_add_1', port 'B[24]' is not connected to any nets. (LINT-28)
Warning: In design 'pc_reg_DW01_add_1', port 'B[23]' is not connected to any nets. (LINT-28)
Warning: In design 'pc_reg_DW01_add_1', port 'B[22]' is not connected to any nets. (LINT-28)
Warning: In design 'pc_reg_DW01_add_1', port 'B[21]' is not connected to any nets. (LINT-28)
Warning: In design 'pc_reg_DW01_add_1', port 'B[20]' is not connected to any nets. (LINT-28)
Warning: In design 'pc_reg_DW01_add_1', port 'B[19]' is not connected to any nets. (LINT-28)
Warning: In design 'pc_reg_DW01_add_1', port 'B[18]' is not connected to any nets. (LINT-28)
Warning: In design 'pc_reg_DW01_add_1', port 'B[17]' is not connected to any nets. (LINT-28)
Warning: In design 'pc_reg_DW01_add_1', port 'B[16]' is not connected to any nets. (LINT-28)
Warning: In design 'pc_reg_DW01_add_1', port 'B[15]' is not connected to any nets. (LINT-28)
Warning: In design 'pc_reg_DW01_add_1', port 'B[14]' is not connected to any nets. (LINT-28)
Warning: In design 'pc_reg_DW01_add_1', port 'B[13]' is not connected to any nets. (LINT-28)
Warning: In design 'pc_reg_DW01_add_1', port 'B[12]' is not connected to any nets. (LINT-28)
Warning: In design 'pc_reg_DW01_add_1', port 'B[11]' is not connected to any nets. (LINT-28)
Warning: In design 'pc_reg_DW01_add_1', port 'B[10]' is not connected to any nets. (LINT-28)
Warning: In design 'pc_reg_DW01_add_1', port 'B[9]' is not connected to any nets. (LINT-28)
Warning: In design 'pc_reg_DW01_add_1', port 'B[8]' is not connected to any nets. (LINT-28)
Warning: In design 'pc_reg_DW01_add_1', port 'B[7]' is not connected to any nets. (LINT-28)
Warning: In design 'pc_reg_DW01_add_1', port 'B[6]' is not connected to any nets. (LINT-28)
Warning: In design 'pc_reg_DW01_add_1', port 'B[5]' is not connected to any nets. (LINT-28)
Warning: In design 'pc_reg_DW01_add_1', port 'B[4]' is not connected to any nets. (LINT-28)
Warning: In design 'pc_reg_DW01_add_1', port 'B[3]' is not connected to any nets. (LINT-28)
Warning: In design 'pc_reg_DW01_add_1', port 'B[2]' is not connected to any nets. (LINT-28)
Warning: In design 'pc_reg_DW01_add_1', port 'B[1]' is not connected to any nets. (LINT-28)
Warning: In design 'pc_reg_DW01_add_1', port 'B[0]' is not connected to any nets. (LINT-28)
Warning: In design 'pc_reg_DW01_add_1', port 'CI' is not connected to any nets. (LINT-28)
Warning: In design 'pc_reg_DW01_add_1', port 'CO' is not connected to any nets. (LINT-28)
Warning: In design 'pc_reg_DW01_add_2', port 'CI' is not connected to any nets. (LINT-28)
Warning: In design 'pc_reg_DW01_add_2', port 'CO' is not connected to any nets. (LINT-28)
```

Figure 4.24: Information of part of the unconnected ports

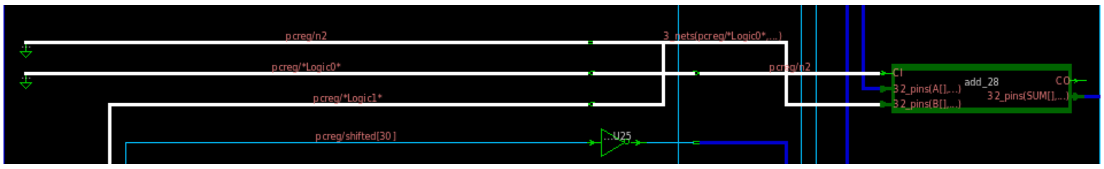
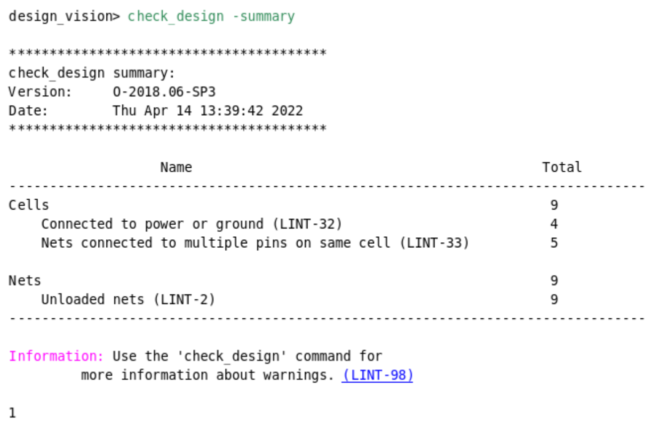
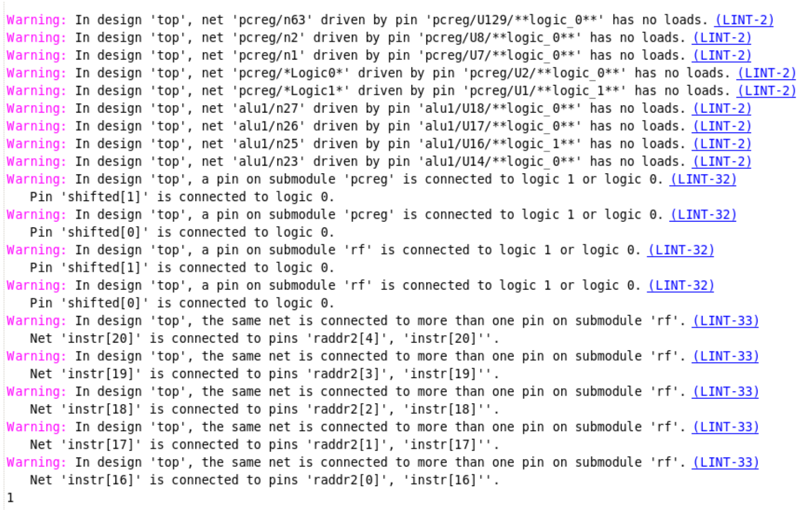


Figure 4.25: Connection of adder from pcreg module with connected ports



```
design_vision> check_design -summary

****************************************
check_design summary:
Version:     O-2018.06-SP3
Date:        Thu Apr 14 13:39:42 2022
****************************************

                   Name                                            Total
--------------------------------------------------------------------------------
Cells                                                                9
    Connected to power or ground (LINT-32)                           4
    Nets connected to multiple pins on same cell (LINT-33)           5

Nets                                                                 9
    Unloaded nets (LINT-2)                                           9
--------------------------------------------------------------------------------

Information: Use the 'check_design' command for
         more information about warnings. (LINT-98)

1
```

Figure 4.26: Final result of check design after removing unconnected ports



```
Warning: In design 'top', net 'pcreg/n63' driven by pin 'pcreg/U129/**logic_0**' has no loads. (LINT-2)
Warning: In design 'top', net 'pcreg/n2' driven by pin 'pcreg/U8/**logic_0**' has no loads. (LINT-2)
Warning: In design 'top', net 'pcreg/n1' driven by pin 'pcreg/U7/**logic_0**' has no loads. (LINT-2)
Warning: In design 'top', net 'pcreg/*Logic0*' driven by pin 'pcreg/U2/**logic_0**' has no loads. (LINT-2)
Warning: In design 'top', net 'pcreg/*Logic1*' driven by pin 'pcreg/U1/**logic_1**' has no loads. (LINT-2)
Warning: In design 'top', net 'alu1/n27' driven by pin 'alu1/U18/**logic_0**' has no loads. (LINT-2)
Warning: In design 'top', net 'alu1/n26' driven by pin 'alu1/U17/**logic_0**' has no loads. (LINT-2)
Warning: In design 'top', net 'alu1/n25' driven by pin 'alu1/U16/**logic_1**' has no loads. (LINT-2)
Warning: In design 'top', net 'alu1/n23' driven by pin 'alu1/U14/**logic_0**' has no loads. (LINT-2)
Warning: In design 'top', a pin on submodule 'pcreg' is connected to logic 1 or logic 0. (LINT-32)
   Pin 'shifted[1]' is connected to logic 0.
Warning: In design 'top', a pin on submodule 'pcreg' is connected to logic 1 or logic 0. (LINT-32)
   Pin 'shifted[0]' is connected to logic 0.
Warning: In design 'top', a pin on submodule 'rf' is connected to logic 1 or logic 0. (LINT-32)
   Pin 'shifted[1]' is connected to logic 0.
Warning: In design 'top', a pin on submodule 'rf' is connected to logic 1 or logic 0. (LINT-32)
   Pin 'shifted[0]' is connected to logic 0.
Warning: In design 'top', the same net is connected to more than one pin on submodule 'rf'. (LINT-33)
   Net 'instr[20]' is connected to pins 'raddr2[4]', 'instr[20]''.
Warning: In design 'top', the same net is connected to more than one pin on submodule 'rf'. (LINT-33)
   Net 'instr[19]' is connected to pins 'raddr2[3]', 'instr[19]''.
Warning: In design 'top', the same net is connected to more than one pin on submodule 'rf'. (LINT-33)
   Net 'instr[18]' is connected to pins 'raddr2[2]', 'instr[18]''.
Warning: In design 'top', the same net is connected to more than one pin on submodule 'rf'. (LINT-33)
   Net 'instr[17]' is connected to pins 'raddr2[1]', 'instr[17]''.
Warning: In design 'top', the same net is connected to more than one pin on submodule 'rf'. (LINT-33)
   Net 'instr[16]' is connected to pins 'raddr2[0]', 'instr[16]''.
1
```

Figure 4.27: Detail of the warnings in check design after removing unconnected ports

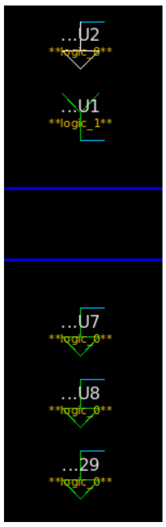


Figure 4.28: Unloaded nets in pcreg module

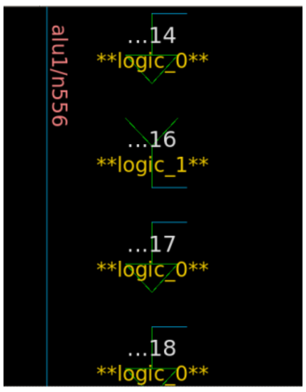
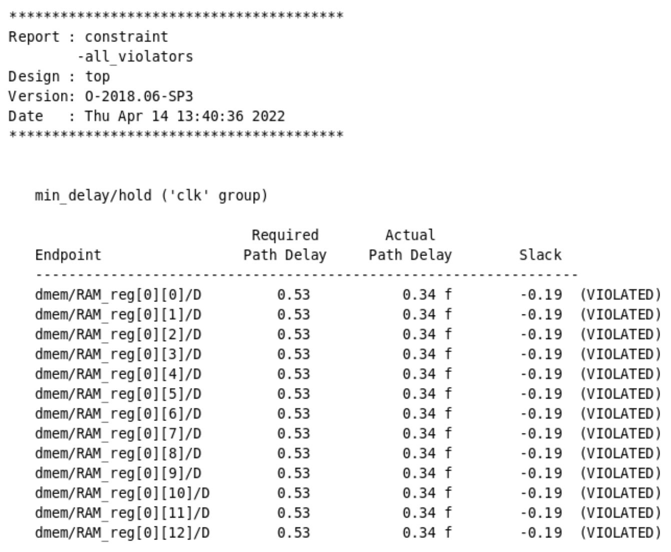


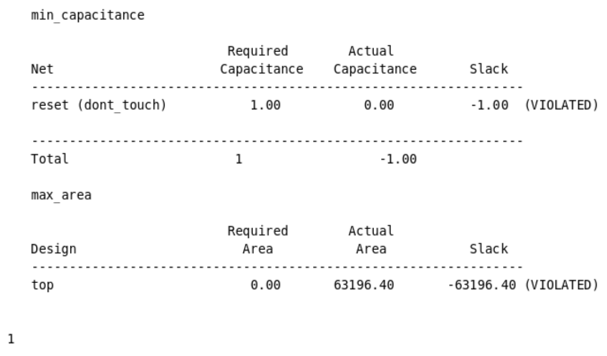
Figure 4.29: Unloaded nets in alu1 module

Before outputting the gate-level netlist, the design is checked with the constraints applied to it. The report generated by the tool to list the violated constraints is shown in Figure 4.30 and Figure 4.31. There are three violations in the optimized design. The hold time violation occurs in the optimized design. For hold time violation, it will be fixed in the physical design through CTS. Hence, the hold time violation is ignored and left to be solved in the back-end design flow. The minimum capacitance allowed in the design is 1 fF. Since the actual capacitance in *reset* net is 0 F, lower than the minimum value, there is no actual violation. The constraint of minimum capacitance is set by the library. For the tool to generate a minimum area during optimization, the constraint of the maximum area is set to be 0 µm^2. Since the constraint is applied to achieve the smallest design area instead of an actual area limitation, the actual design area of 63196.40 µm^2 is not violating the maximum area constraint. Thus, the violations can be ignored.



```
****************************************
Report : constraint
        -all_violators
Design : top
Version: O-2018.06-SP3
Date   : Thu Apr 14 13:40:36 2022
****************************************


   min_delay/hold ('clk' group)

                            Required        Actual
   Endpoint                 Path Delay     Path Delay        Slack
   -----------------------------------------------------------------
   dmem/RAM_reg[0][0]/D         0.53          0.34 f        -0.19  (VIOLATED)
   dmem/RAM_reg[0][1]/D         0.53          0.34 f        -0.19  (VIOLATED)
   dmem/RAM_reg[0][2]/D         0.53          0.34 f        -0.19  (VIOLATED)
   dmem/RAM_reg[0][3]/D         0.53          0.34 f        -0.19  (VIOLATED)
   dmem/RAM_reg[0][4]/D         0.53          0.34 f        -0.19  (VIOLATED)
   dmem/RAM_reg[0][5]/D         0.53          0.34 f        -0.19  (VIOLATED)
   dmem/RAM_reg[0][6]/D         0.53          0.34 f        -0.19  (VIOLATED)
   dmem/RAM_reg[0][7]/D         0.53          0.34 f        -0.19  (VIOLATED)
   dmem/RAM_reg[0][8]/D         0.53          0.34 f        -0.19  (VIOLATED)
   dmem/RAM_reg[0][9]/D         0.53          0.34 f        -0.19  (VIOLATED)
   dmem/RAM_reg[0][10]/D        0.53          0.34 f        -0.19  (VIOLATED)
   dmem/RAM_reg[0][11]/D        0.53          0.34 f        -0.19  (VIOLATED)
   dmem/RAM_reg[0][12]/D        0.53          0.34 f        -0.19  (VIOLATED)
```

Figure 4.30: Hold time violation

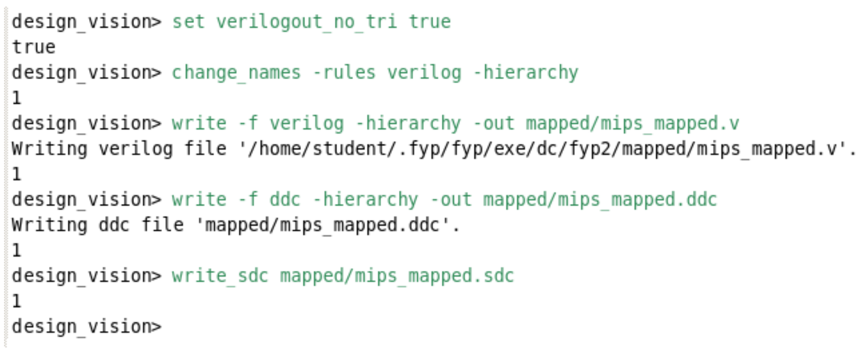


```
   min_capacitance

                          Required       Actual
   Net                   Capacitance   Capacitance      Slack
   -----------------------------------------------------------------
   reset (dont_touch)        1.00          0.00         -1.00  (VIOLATED)

   -----------------------------------------------------------------
   Total                   1                -1.00

   max_area

                          Required       Actual
   Design                   Area          Area           Slack
   -----------------------------------------------------------------
   top                       0.00       63196.40      -63196.40 (VIOLATED)


1
```

Figure 4.31: Result showing min capacitance and max area violation after mapping

4.2.7. Output Gate-level Netlist

When the design is analysed and verified to have met all the constraints applied to it, the gate-level netlist is output to be used in the back-end design. The violations of minimum capacitance and maximum area constraints are ignored. The optimized design is saved in ddc format and Verilog format to be used as a gate-level netlist. A sdc file is generated to act as a design constraint file with the record of all constraints applied to the design. Figure 4.32 shows the outing of the gate-level netlist is completed successfully. This makes a successful completion of logic synthesis and the end of the front-end design flow.



```
design_vision> set verilogout_no_tri true
true
design_vision> change_names -rules verilog -hierarchy
1
design_vision> write -f verilog -hierarchy -out mapped/mips_mapped.v
Writing verilog file '/home/student/.fyp/fyp/exe/dc/fyp2/mapped/mips_mapped.v'.
1
design_vision> write -f ddc -hierarchy -out mapped/mips_mapped.ddc
Writing ddc file 'mapped/mips_mapped.ddc'.
1
design_vision> write_sdc mapped/mips_mapped.sdc
1
design_vision>
```

Figure 4.32: Result for output gate-level netlist

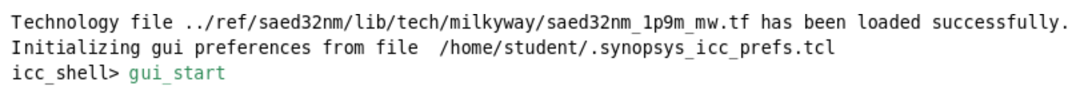
4.3. Physical design

In this phase, the optimised gate-level netlist generated in logic synthesis stage is transformed into layout using Synopsys ICC as the design tool.

4.3.1. ICC Setup and Gate-netlist import

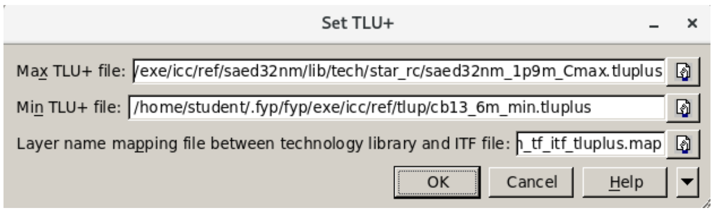
A setup file is used to set up the library in ICC. The setup file specifies the target library, synthetic library, and link library. Since there are no I/O pads in this design, the I/O library is not loaded into the software. Similar to the library setup in Design Compiler, saed32lvt_ss0p95v125c.db is used as the target library and listed in the link library for ICC to ensure the same technical information as logic synthesis is used in the physical design. The synthetic library is dw_foundation.sldb.

A milkyway library named MW_MIPS is created using the setup file to act the design library. A tech file is loaded into the design library as shown in Figure 4.33. The setup of tluplus files is also done manually using the GUI as shown in Figure 4.34. After the setup, the ICC is ready to load the optimized netlist. The *ddc* netlist is imported into ICC. The loading of libraries is done when the design is imported into the tool. Figure 4.35 shows the design is imported successfully into ICC and the loading of libraries is completed. The tool creates an initial design cell after reading in the netlist as shown in Figure 4.36. Since the design does not consist of any macro cells and I/O pads, only standard cells are generated in the terms of purple rectangles. There are 12954 standard cells in the design. The *list_libs* command is executed to check and make sure all the required libraries are loaded into the tool. Figure 4.37 shows all the required libraries have been loaded successfully and the tool is ready to perform physical design.



```
Technology file ../ref/saed32nm/lib/tech/milkyway/saed32nm_1p9m_mw.tf has been loaded successfully.
Initializing gui preferences from file  /home/student/.synopsys_icc_prefs.tcl
icc_shell> gui_start
```

Figure 4.33: Load tech file during design library creation



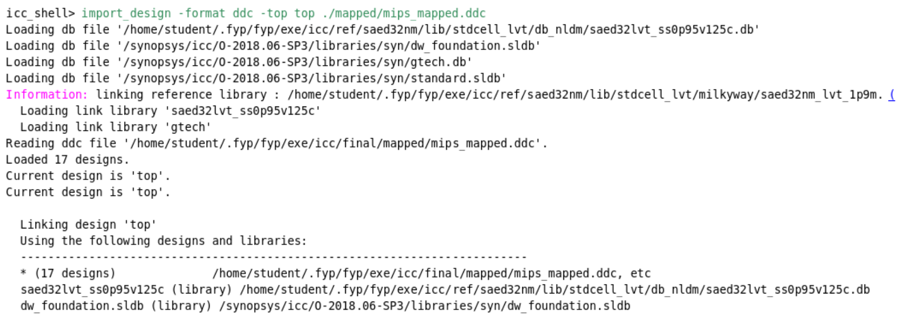
Set TLU+

Max TLU+ file: /exe/icc/ref/saed32nm/lib/tech/star_rc/saed32nm_1p9m_Cmax.tluplus

Min TLU+ file: /home/student/.fyp/fyp/exe/icc/ref/tlup/cb13_6m_min.tluplus

Layer name mapping file between technology library and ITF file: n_tf_itf_tluplus.map

Figure 4.34: Setup of tluplus files



```
icc_shell> import_design -format ddc -top top ./mapped/mips_mapped.ddc
Loading db file '/home/student/.fyp/fyp/exe/icc/ref/saed32nm/lib/stdcell_lvt/db_nldm/saed32lvt_ss0p95v125c.db'
Loading db file '/synopsys/icc/O-2018.06-SP3/libraries/syn/dw_foundation.sldb'
Loading db file '/synopsys/icc/O-2018.06-SP3/libraries/syn/gtech.db'
Loading db file '/synopsys/icc/O-2018.06-SP3/libraries/syn/standard.sldb'
Information: linking reference library : /home/student/.fyp/fyp/exe/icc/ref/saed32nm/lib/stdcell_lvt/milkyway/saed32nm_lvt_1p9m. (
  Loading link library 'saed32lvt_ss0p95v125c'
  Loading link library 'gtech'
Reading ddc file '/home/student/.fyp/fyp/exe/icc/final/mapped/mips_mapped.ddc'.
Loaded 17 designs.
Current design is 'top'.
Current design is 'top'.

  Linking design 'top'
  Using the following designs and libraries:
  --------------------------------------------------------------------------
  * (17 designs)              /home/student/.fyp/fyp/exe/icc/final/mapped/mips_mapped.ddc, etc
  saed32lvt_ss0p95v125c (library) /home/student/.fyp/fyp/exe/icc/ref/saed32nm/lib/stdcell_lvt/db_nldm/saed32lvt_ss0p95v125c.db
  dw_foundation.sldb (library) /synopsys/icc/O-2018.06-SP3/libraries/syn/dw_foundation.sldb
```

Figure 4.35: Import of design and loading of libraries

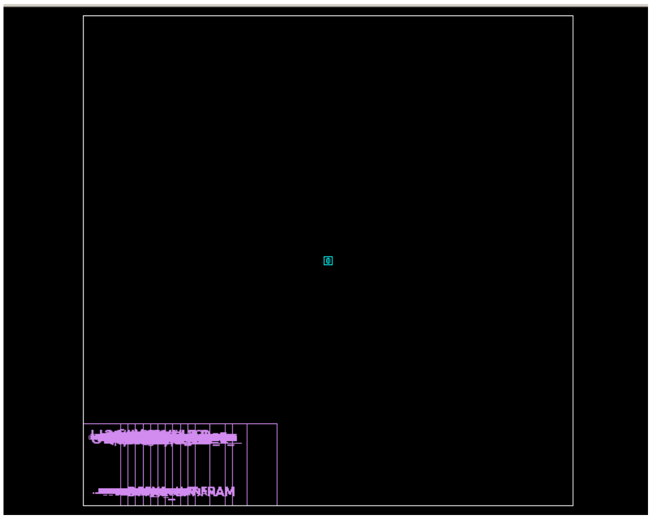
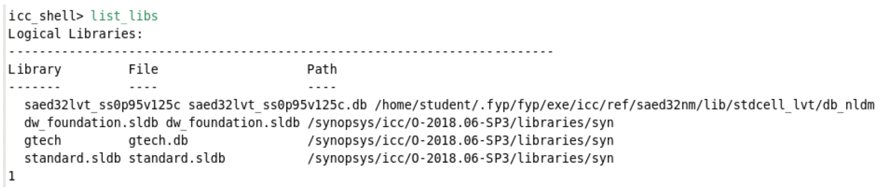


Figure 4.36: Initial view of the top module after loading the design



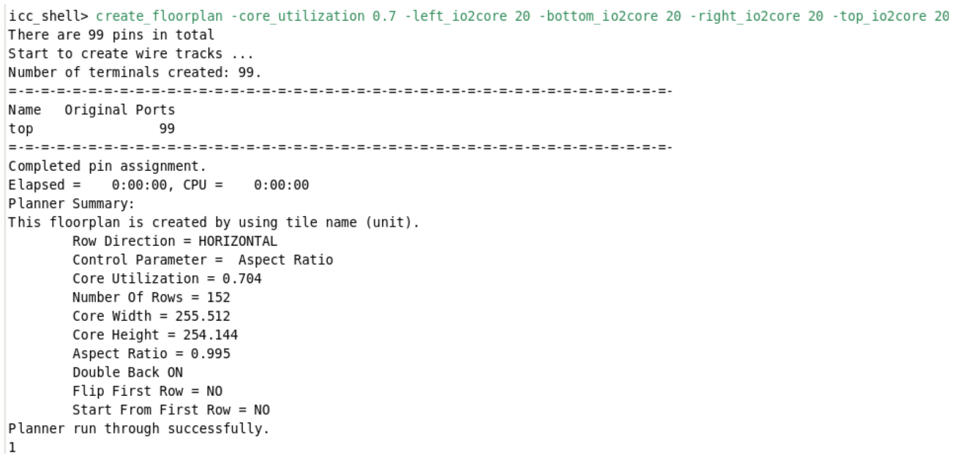
```
icc_shell> list_libs
Logical Libraries:
-------------------------------------------------------------------------
Library         File                    Path
-------         ----                    ----
  saed32lvt_ss0p95v125c saed32lvt_ss0p95v125c.db /home/student/.fyp/fyp/exe/icc/ref/saed32nm/lib/stdcell_lvt/db_nldm
  dw_foundation.sldb dw_foundation.sldb /synopsys/icc/O-2018.06-SP3/libraries/syn
  gtech         gtech.db                /synopsys/icc/O-2018.06-SP3/libraries/syn
  standard.sldb standard.sldb           /synopsys/icc/O-2018.06-SP3/libraries/syn
1
```

Figure 4.37: Checking on libraries loading

4.3.2. Floorplanning

The first stage of physical design is floorplanning After setting up the libraries and the loading of design, the physical design is started with the floorplanning stage. Floorplanning acts as the foundation of physical design in which the quality of a floorplan will affect placement and routing process. In this stage, the chip area is defined and the area for routing is determined.

Before creating a floorplan, logical connections for ground and power are created using *the derive_pg_connection* command. It is to ensure the ground net and power net are connected logically to pins including ground, power, and tie-off pins. The command is executed when there are changes in ground and power connection after floorplanning, placement, routing, and chip-finishing. The floorplan is generated by applying the command *create_floorplan* and some parameters are set. The setting to create the floorplan and the summary of the floorplan is as shown in Figure 4.38. The core utilization ratio is the parameter to determine the area reserves for placement. The core utilization ratio is set as 0.7 in which 70% of the chip area will be used for placement and the remaining 30% is available for routing. It is important to reserve sufficient area to avoid routing congestions later. The distance between the core area and the terminals or pads is set at 20 µm for all four sides of the design. By applying the setting, the tool generates an optimum area based on the design automatically as shown in Figure 4.39. The standard cells are placed along the right sides of the floorplan. The square inside the floorplan is the core of the chip. Based on the summary of the floorplan, the actual core utilization ratio is 0.704 or 70.4%. It is slightly higher than the pre-defined value of 0.7. The tool is generating a floorplan with an optimum area as close as possible to the predefined value.



```
icc_shell> create_floorplan -core_utilization 0.7 -left_io2core 20 -bottom_io2core 20 -right_io2core 20 -top_io2core 20
There are 99 pins in total
Start to create wire tracks ...
Number of terminals created: 99.
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Name   Original Ports
top               99
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Completed pin assignment.
Elapsed =    0:00:00, CPU =    0:00:00
Planner Summary:
This floorplan is created by using tile name (unit).
        Row Direction = HORIZONTAL
        Control Parameter =  Aspect Ratio
        Core Utilization = 0.704
        Number Of Rows = 152
        Core Width = 255.512
        Core Height = 254.144
        Aspect Ratio = 0.995
        Double Back ON
        Flip First Row = NO
        Start From First Row = NO
Planner run through successfully.
1
```

Figure 4.38: Setting to create floorplan and summary of the floorplan

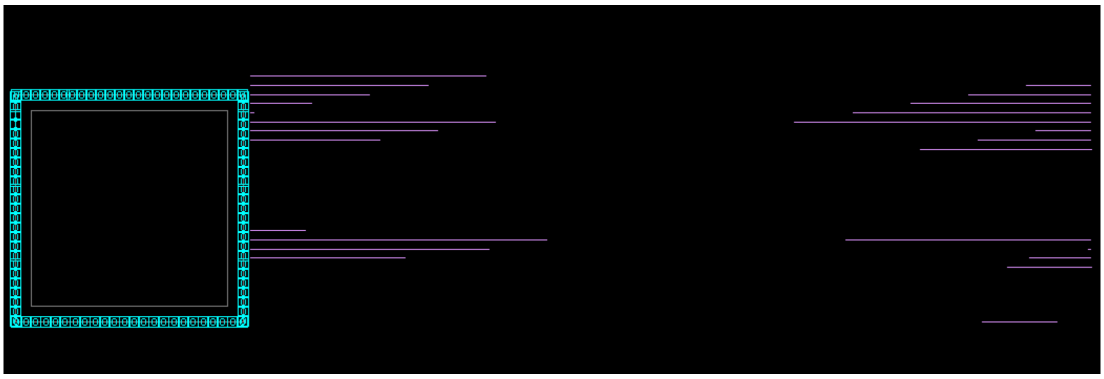
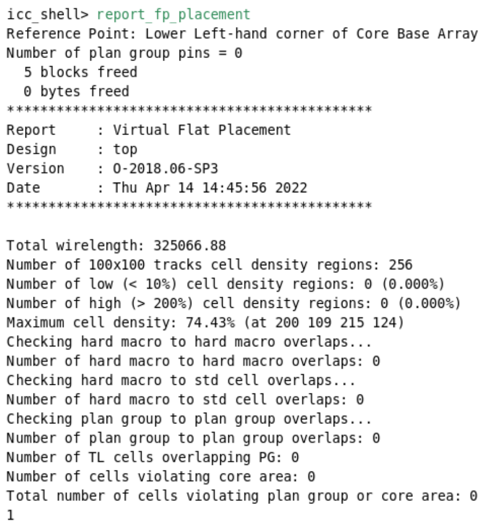


Figure 4.39: Floorplan generated by the tool

After the initial floorplan is generated, the next step is virtual flat placement to place the standard cells legally in the core area. This is to enhance the congestion and timing after creating the floorplan. There are some parameters are set before performing virtual flat placement. The congestion effort is set to high to reduce the congestion issues and the pin routing aware is set to true in order to avoid the DRC error using *set_fp_placement_strategy* command. Then, virtual flat placement is performed. A report of virtual flat placement is generated as shown in Figure 4.40. Based on the result, there is no overlapping of cells, and no cells are violating the core area. Hence, all the standard cells are placed in the predefined area which is 70% of the core area.



```
icc_shell> report_fp_placement
Reference Point: Lower Left-hand corner of Core Base Array
Number of plan group pins = 0
  5 blocks freed
  0 bytes freed
*********************************************
Report     : Virtual Flat Placement
Design     : top
Version    : O-2018.06-SP3
Date       : Thu Apr 14 14:45:56 2022
*********************************************

Total wirelength: 325066.88
Number of 100x100 tracks cell density regions: 256
Number of low (< 10%) cell density regions: 0 (0.000%)
Number of high (> 200%) cell density regions: 0 (0.000%)
Maximum cell density: 74.43% (at 200 109 215 124)
Checking hard macro to hard macro overlaps...
Number of hard macro to hard macro overlaps: 0
Checking hard macro to std cell overlaps...
Number of hard macro to std cell overlaps: 0
Checking plan group to plan group overlaps...
Number of plan group to plan group overlaps: 0
Number of TL cells overlapping PG: 0
Number of cells violating core area: 0
Total number of cells violating plan group or core area: 0
1
```

Figure 4.40: Summary of virtual flat placement

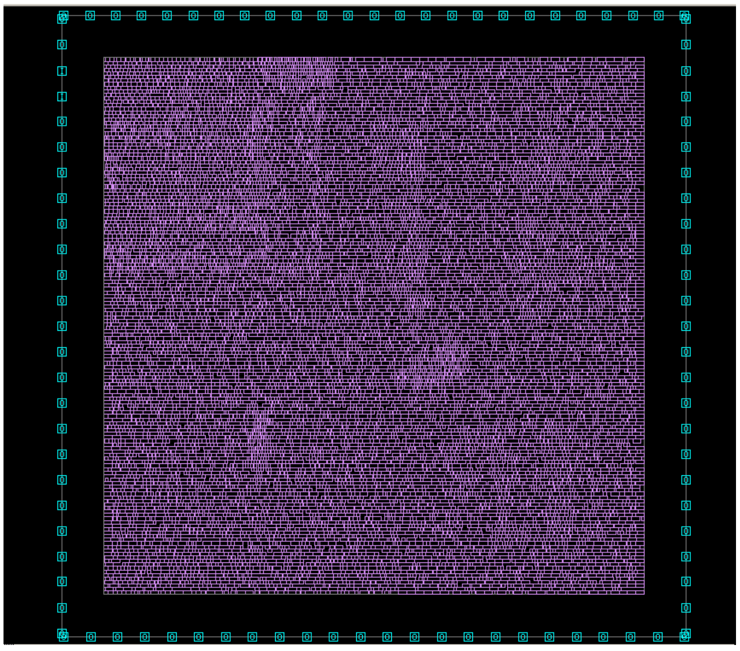
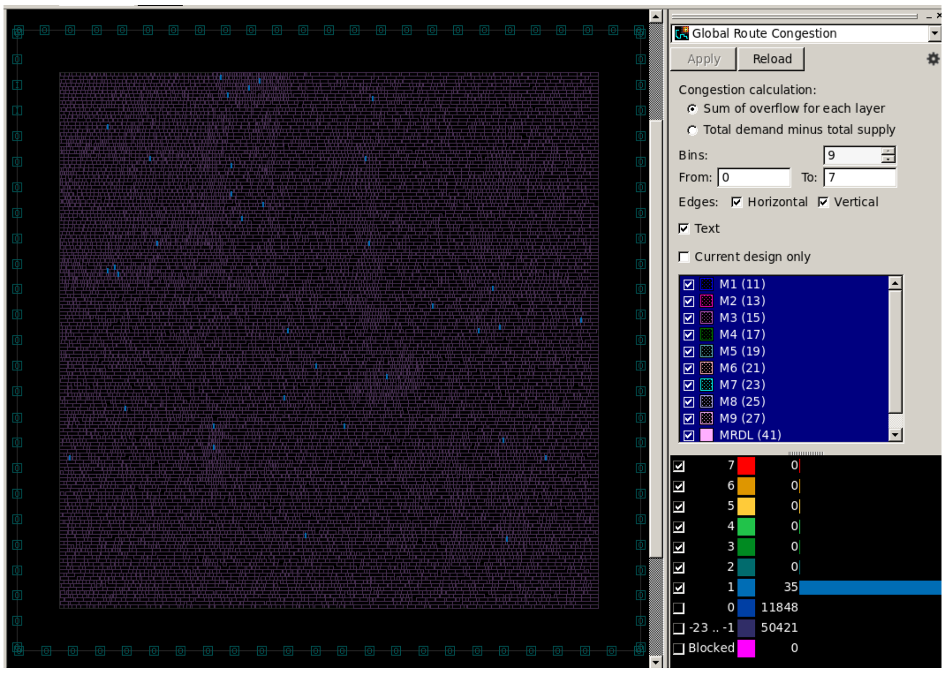


Figure 4.41: Result of virtual flat placement

The congestion of the design is checked by generating a global route congestion map after performing virtual placement. The congestion map is the visualization of the placement quality. The overflow of the cells is determined and highlighted using different colours. The brighter the colour, the larger the overflow. The global route congestion map is shown in Figure 4.42. From the map, there are a few cells highlighted in blue, indicating the design does not have serious congestion issues. An enlarged map is shown in Figure 4.43. The tool calculates the congestion by comparing the supply and demand of nets based on the global routing cell (GRC). The number of tracks available for routing is the supply while the demand is the quantity of nets to cross the edge of GRC. When there the demand is greater than the supply, overflow occurs. A large overflow will lead to serious congestion issues during routing. The congestion report based on GRC is generated as shown in Figure 4.44. The design consists of 31152 GRC while 2 of them are having overflows. The overflow GRC is 0.006% of the total GRC. Since the number of overflow GRC is only 2 with overflow below 10, the congestion is acceptable as the routablity is still high. Hence, the design is about to proceed to the next stage.



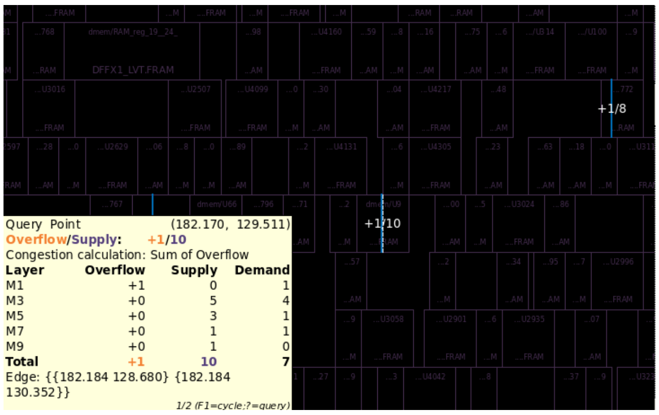
Congestion calculation:
[✓ Sum of overflow for each layer]
Total demand minus total supply

Bins: 9
From: 0 To: 7
Edges: ✓ Horizontal ✓ Vertical
✓ Text
Current design only

✓ M1 (11)
✓ M2 (13)
✓ M3 (15)
✓ M4 (17)
✓ M5 (19)
✓ M6 (21)
✓ M7 (23)
✓ M8 (25)
✓ M9 (27)
✓ MRDL (41)

✓	7	🟥	0
✓	6	🟧	0
✓	5	🟨	0
✓	4	🟩	0
✓	3	🟩	0
✓	2	🟦	0
✓	1	🟦	35
☐	0	🟦	11848
☐	-23 .. -1	🟪	50421
☐	Blocked	🟪	0

Figure 4.42: Global route congestion map



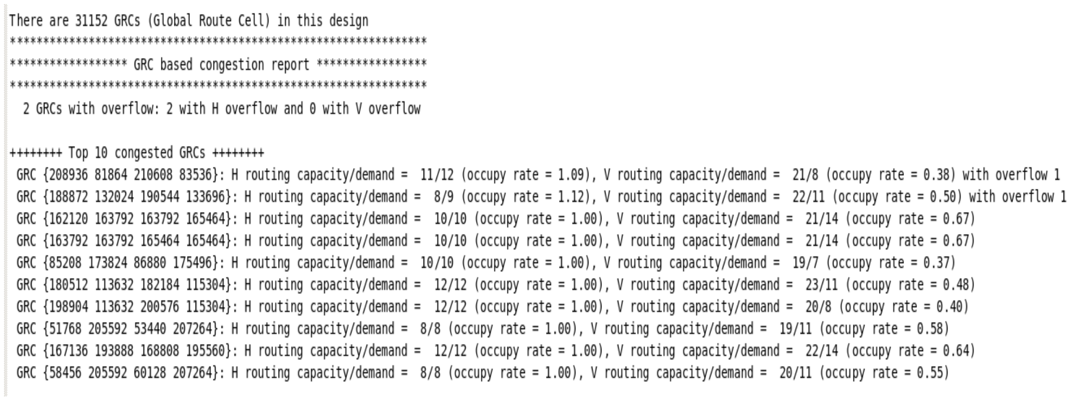
Query Point (182.170, 129.511)
Overflow/Supply: +1/10
Congestion calculation: Sum of Overflow

Layer	Overflow	Supply	Demand
M1	+1	0	1
M3	+0	5	4
M5	+0	3	1
M7	+0	1	1
M9	+0	1	0
Total	+1	10	**7**

Edge: {{182.184 128.680} {182.184 130.352}}

1/2 (F1=cycle;?=query)

Figure 4.43: Enlarged congestion map with congestion calculation



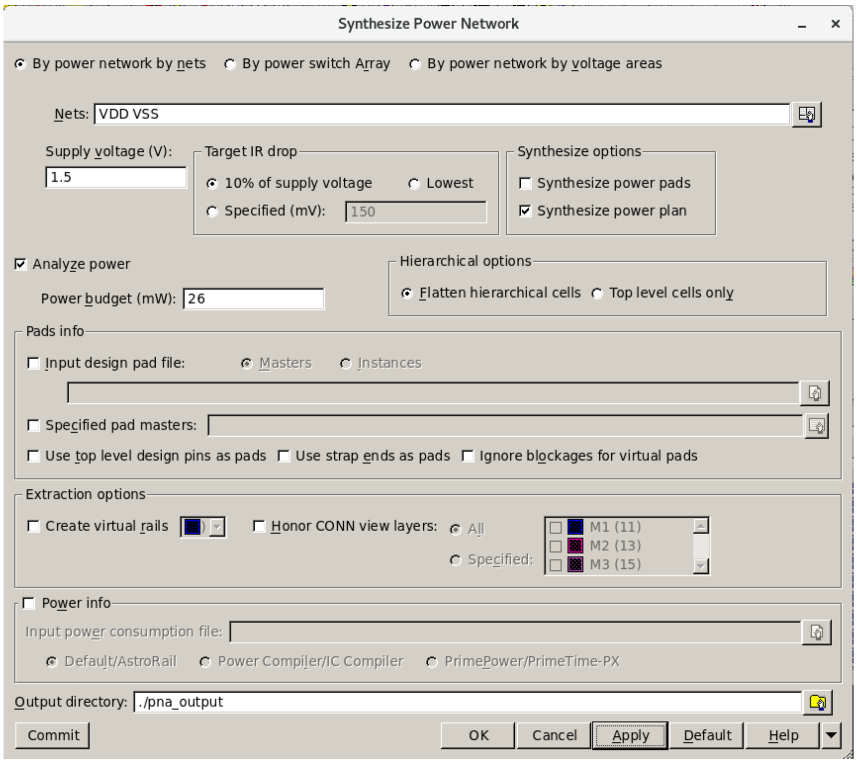
```
There are 31152 GRCs (Global Route Cell) in this design
****************************************************************
***************** GRC based congestion report *****************
****************************************************************
  2 GRCs with overflow: 2 with H overflow and 0 with V overflow

++++++++ Top 10 congested GRCs ++++++++
 GRC {208936 81864 210608 83536}: H routing capacity/demand =  11/12 (occupy rate = 1.09), V routing capacity/demand =  21/8 (occupy rate = 0.38) with overflow 1
 GRC {188872 132024 190544 133696}: H routing capacity/demand =  8/9 (occupy rate = 1.12), V routing capacity/demand =  22/11 (occupy rate = 0.50) with overflow 1
 GRC {162120 163792 163792 165464}: H routing capacity/demand =  10/10 (occupy rate = 1.00), V routing capacity/demand =  21/14 (occupy rate = 0.67)
 GRC {163792 163792 165464 165464}: H routing capacity/demand =  10/10 (occupy rate = 1.00), V routing capacity/demand =  21/14 (occupy rate = 0.67)
 GRC {85208 173824 86880 175496}: H routing capacity/demand =  10/10 (occupy rate = 1.00), V routing capacity/demand =  19/7 (occupy rate = 0.37)
 GRC {180512 113632 182184 115304}: H routing capacity/demand =  12/12 (occupy rate = 1.00), V routing capacity/demand =  23/11 (occupy rate = 0.48)
 GRC {198904 113632 200576 115304}: H routing capacity/demand =  12/12 (occupy rate = 1.00), V routing capacity/demand =  20/8 (occupy rate = 0.40)
 GRC {51768 205592 53440 207264}: H routing capacity/demand =  8/8 (occupy rate = 1.00), V routing capacity/demand =  19/11 (occupy rate = 0.58)
 GRC {167136 193888 168808 195560}: H routing capacity/demand =  12/12 (occupy rate = 1.00), V routing capacity/demand =  22/14 (occupy rate = 0.64)
 GRC {58456 205592 60128 207264}: H routing capacity/demand =  8/8 (occupy rate = 1.00), V routing capacity/demand =  20/11 (occupy rate = 0.55)
```

Figure 4.44: Congestion report based on GRC

In this stage, power network synthesis (PNS) is performed. Before performing PNS, power network constraints are applied to the design to control the number and the width of the power rings and metal strap. The tool calculates the number of straps required in PNS to evenly distribute the power according to the constraints. Since there are no I/O pads in the design, the design does not have any ground pad or power pad. In order to perform PNS, the power and ground pads are essential. Thus, virtual power pads for VSS and VDD are created using the GUI of the tool to solve the problem. The PNS is performed using the GUI and the setting is as shown in Figure 4.45. The essential parameters including the value of supply voltage, name of power net, target IR drop and power budget must be set to synthesize the power network. The power net is named VDD and VSS in this project. For supply voltage, it is set as 1.5 V while the 10% of supply voltage is set as the target IR drop. In other words, the desired IR drop is 0.15 V.

According to the power report in the logic synthesis stage, the design dissipated about 23.7 mW. Hence, the power budget for the design is set as 26 mW with a 10% of safety margin to ensure there is sufficient power to drive the processor. The maximum IR drop is examined after the PNS. As shown in Figure 4.46, the summary of PNS reported a maximum instance IR drop of 9.356 mV. The maximum drop in voltage is smaller than the target value of 150 mV. The IR drop is acceptable. Hence, the power network is committed based on the PNS result. The PNS heat map is shown in Figure 4.47. The virtual power pads located around the chip are highlighted in light purple colour. The voltage drop at the center of the chip indicated in red colour is the worst. The voltage from the power rings drops along the power strap due to the resistance of the metal wire. Since the distance between the power rings and the center of the chip is the longest, the chip center experiences the most voltage drop. The layout after committing the power network is shown in Figure 4.48.

After that, the pins in instances are connected to the ground and power using *preroute_instances* command. The *set_preroute_drc_strategy* command is used to prevent blockage of the signal pin access edge in this stage which can lead to DRC error. The ground and power pins of the standard cells are then connected to the power strap and rings. At the same time, the tool is instructed to remove the unconnected or floating segments from the design. Before proceeding to legalize the placement, the tool is set to avoid placement of cells under the power nets to avoid DRC errors and congestion by setting a complete blockage to the power straps. The legalization of placement is performed to adjust the locations of the standard cells which are violating the power strap. Figure 4.49 shows the result of placement legalization. The location of 8250 cells in the core of the chip is modified and 5492 cells are rotated. Next, actual global routing is performed, and the congestion report based on GRC is generated again to analyse the actual congestion. The report is shown in Figure 4.50. According to the report, there is no overflow in GRC after the global routing indicating there are no congestion issues in the design. The last step in floorplanning is timing optimization and fixing of violations in design rules using the command. Based on the result shown in Figure 4.51, 1884 cells have experienced modification in placement to optimize the timing and fix the design rule.



Synthesize Power Network

[✓ By power network by nets] By power switch Array By power network by voltage areas

Nets: VDD VSS

Supply voltage (V): 1.5

Target IR drop: [✓ 10% of supply voltage] Lowest Specified (mV): 150

Synthesize options: Synthesize power pads [✓ Synthesize power plan]

[✓ Analyze power]

Power budget (mW): 26

Hierarchical options: [✓ Flatten hierarchical cells] Top level cells only

Pads info

Input design pad file: [✓ Masters] Instances

Specified pad masters:

Use top level design pins as pads Use strap ends as pads Ignore blockages for virtual pads

Extraction options

Create virtual rails Honor CONN view layers: [✓ All] Specified: M1 (11) M2 (13) M3 (15)

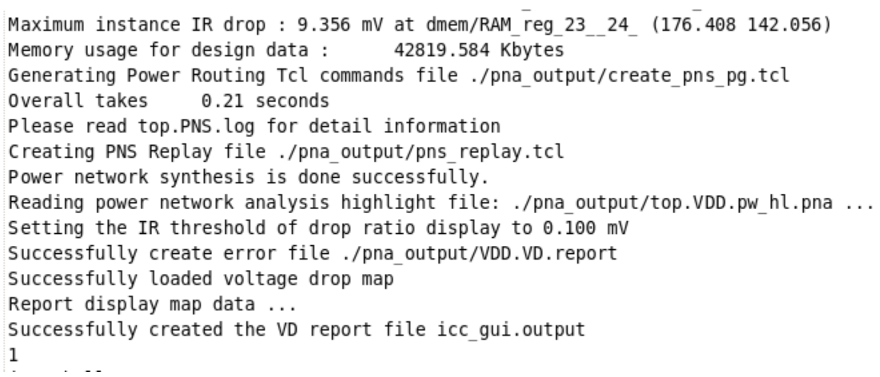
Power info

Input power consumption file:

[✓ Default/AstroRail] Power Compiler/IC Compiler PrimePower/PrimeTime-PX

Output directory: ./pna_output

Figure 4.45: Setting applied to PNS



```
Maximum instance IR drop : 9.356 mV at dmem/RAM_reg_23__24_ (176.408 142.056)
Memory usage for design data :      42819.584 Kbytes
Generating Power Routing Tcl commands file ./pna_output/create_pns_pg.tcl
Overall takes     0.21 seconds
Please read top.PNS.log for detail information
Creating PNS Replay file ./pna_output/pns_replay.tcl
Power network synthesis is done successfully.
Reading power network analysis highlight file: ./pna_output/top.VDD.pw_hl.pna ...
Setting the IR threshold of drop ratio display to 0.100 mV
Successfully create error file ./pna_output/VDD.VD.report
Successfully loaded voltage drop map
Report display map data ...
Successfully created the VD report file icc_gui.output
1
```

Figure 4.46: Maximum IR drop reported in PNS summary

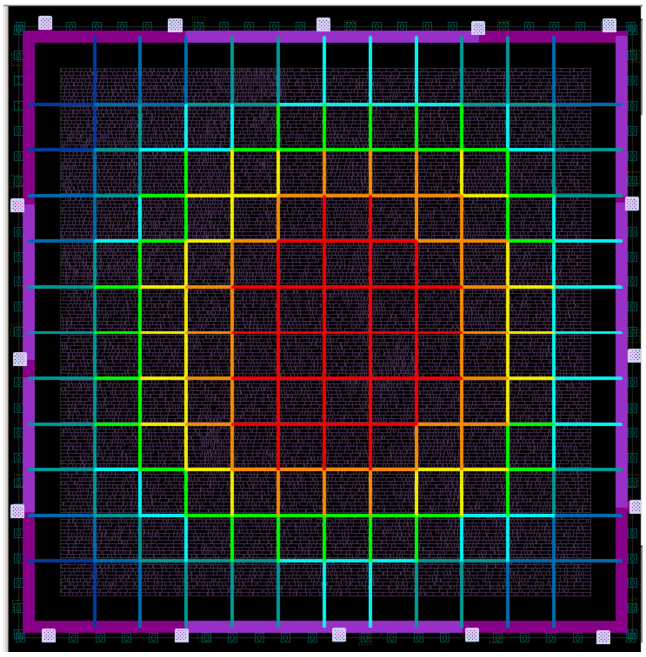


Figure 4.47: Heat map for PNS

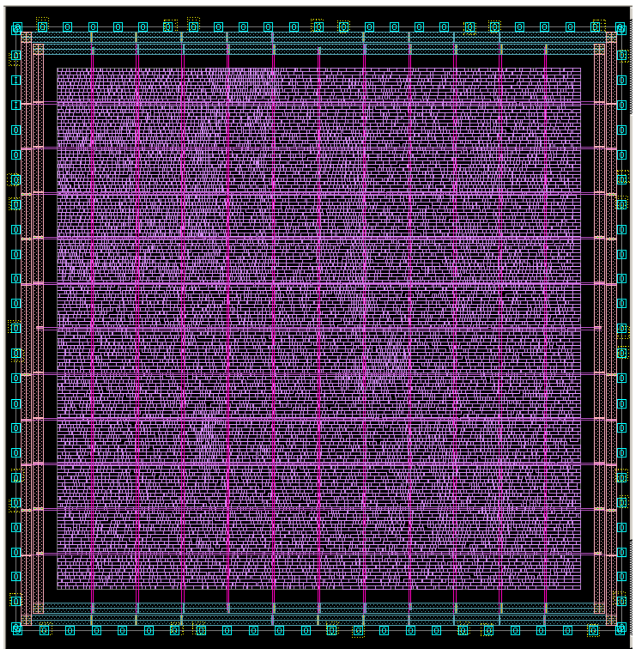
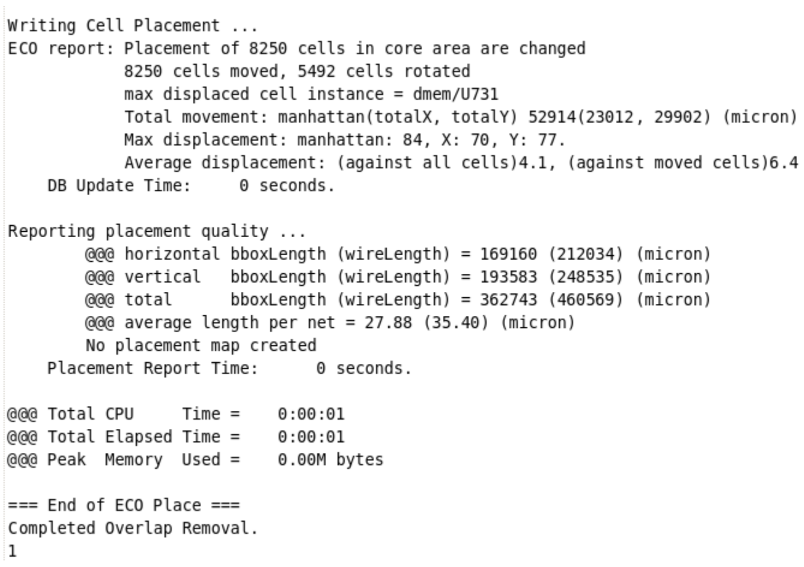
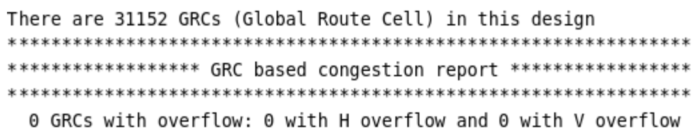


Figure 4.48: Chip layout after committing the power network



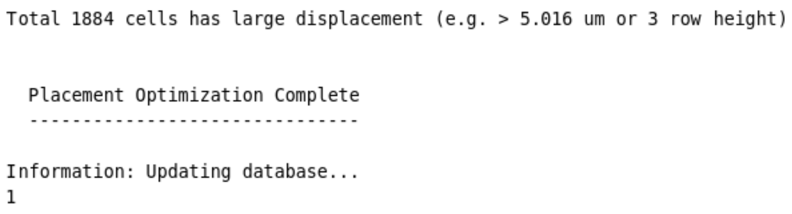
```
Writing Cell Placement ...
ECO report: Placement of 8250 cells in core area are changed
            8250 cells moved, 5492 cells rotated
            max displaced cell instance = dmem/U731
            Total movement: manhattan(totalX, totalY) 52914(23012, 29902) (micron)
            Max displacement: manhattan: 84, X: 70, Y: 77.
            Average displacement: (against all cells)4.1, (against moved cells)6.4
    DB Update Time:     0 seconds.

Reporting placement quality ...
        @@@ horizontal bboxLength (wireLength) = 169160 (212034) (micron)
        @@@ vertical   bboxLength (wireLength) = 193583 (248535) (micron)
        @@@ total      bboxLength (wireLength) = 362743 (460569) (micron)
        @@@ average length per net = 27.88 (35.40) (micron)
        No placement map created
    Placement Report Time:      0 seconds.

@@@ Total CPU     Time =    0:00:01
@@@ Total Elapsed Time =    0:00:01
@@@ Peak  Memory  Used =    0.00M bytes

=== End of ECO Place ===
Completed Overlap Removal.
1
```

Figure 4.49: Result of placement legalization



```
There are 31152 GRCs (Global Route Cell) in this design
****************************************************************
****************** GRC based congestion report *****************
****************************************************************
  0 GRCs with overflow: 0 with H overflow and 0 with V overflow
```

Figure 4.50: Congestion report based of GRC after global routing

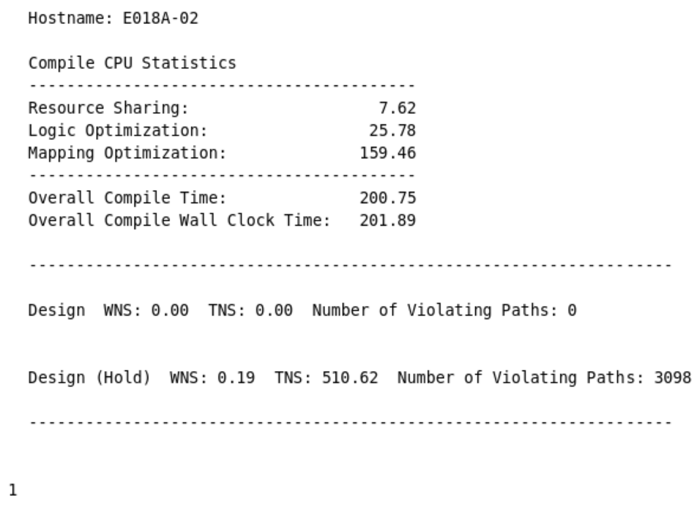


```
Total 1884 cells has large displacement (e.g. > 5.016 um or 3 row height)


  Placement Optimization Complete
  -------------------------------

Information: Updating database...
1
```

Figure 4.51:Result for timing optimization and fixing of design rule violation

4.3.3. Placement, CTS, and Routing

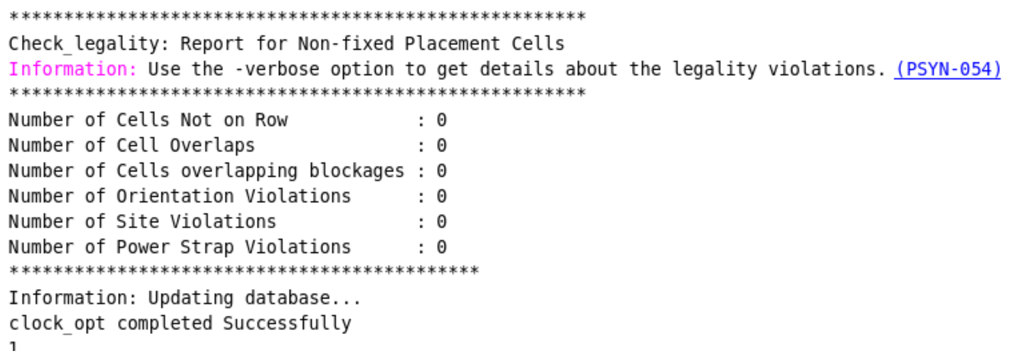
After completing floorplanning and PNS, the next stage in physical design is placement. A few parameters are set for placement optimization is made. In order to obtain optimum chip area, the area recovery option is set to high to allow the tool to perform cell area recovery during placement optimization which helps to reduce congestion issues. Besides, the power option is set high to allow power optimization during placement. The CTS option is also enabled so that the tool can perform CTS when optimizing the placement of the cells. The summary of placement optimization as shown in Figure 4.52 shows the design is having hold time violation with 3098 paths. CTS can be performed to solve the violations.



```
  Hostname: E018A-02

  Compile CPU Statistics
  ----------------------------------------
  Resource Sharing:                   7.62
  Logic Optimization:                25.78
  Mapping Optimization:             159.46
  ----------------------------------------
  Overall Compile Time:             200.75
  Overall Compile Wall Clock Time:  201.89

  -------------------------------------------------------------------

  Design  WNS: 0.00  TNS: 0.00  Number of Violating Paths: 0


  Design (Hold)  WNS: 0.19  TNS: 510.62  Number of Violating Paths: 3098

  -------------------------------------------------------------------


1
```

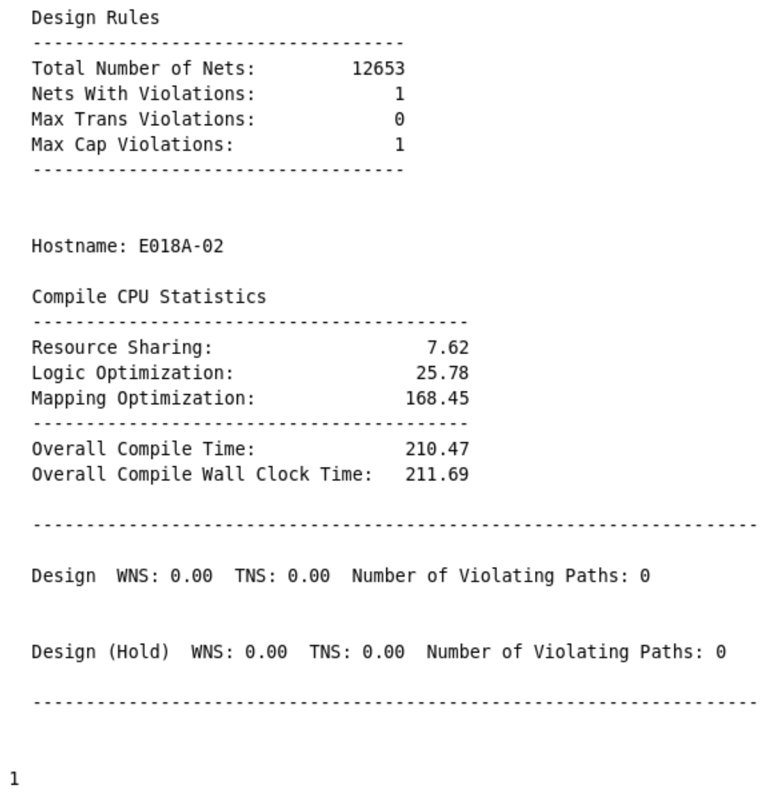
Figure 4.52: Summary of placement optimization

After placement, CTS is carried out to fix the hold time violation by inserting a buffer into the clock path. The area recovery option is set high to optimize the area. The CTS result shows the CTS is completed successfully and there is no overlapping of cells, blockage, and violation at this stage. RC extraction is done and a QoR report is generated. As shown in Figure 4.54, the hold time violations are fixed and there is violating path after CTS. However, 1 out of 12653 nets is found to violate the design rule. The design rule violation can be fixed with the route of the clock tree and detailed routing. A clock tree report summary is generated using the tool as shown in Figure 4.55. The clock skew is 0.0269 ns, and the longest path has a delay of 0.1112 ns. There are 29 buffers inserted into the clock path to produce a balance clock delay and minimize the clock skew during CTS. By reducing clock skew, the clock signal can travel faster to a register, the data is held stable after the clock active edge for a longer time in a register. Hence, the hold violations are solved. The clock tree is routed using global routing. Figure 4.56 shows the total DRC violations after the global routing of the clock tree are 0 and the clock tree is built using 12607 nets. A visualization of the clock tree is shown in Figure 4.57.



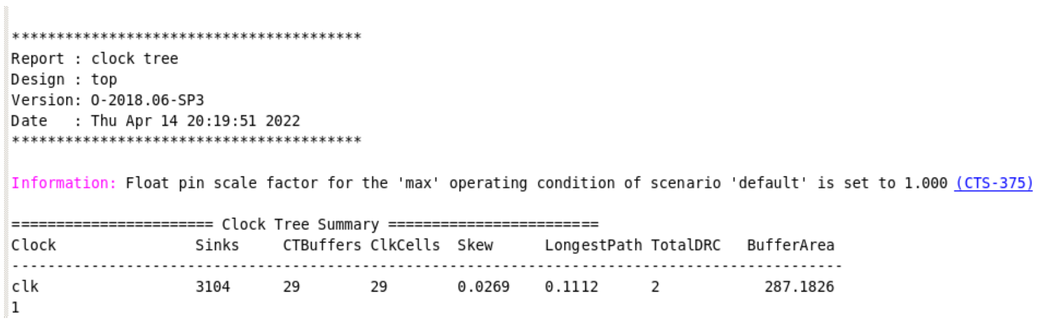
```
*******************************************************
Check_legality: Report for Non-fixed Placement Cells
Information: Use the -verbose option to get details about the legality violations. (PSYN-054)
*******************************************************
Number of Cells Not on Row            : 0
Number of Cell Overlaps               : 0
Number of Cells overlapping blockages : 0
Number of Orientation Violations      : 0
Number of Site Violations             : 0
Number of Power Strap Violations      : 0
*********************************************
Information: Updating database...
clock_opt completed Successfully
1
```

Figure 4.53: CTS result



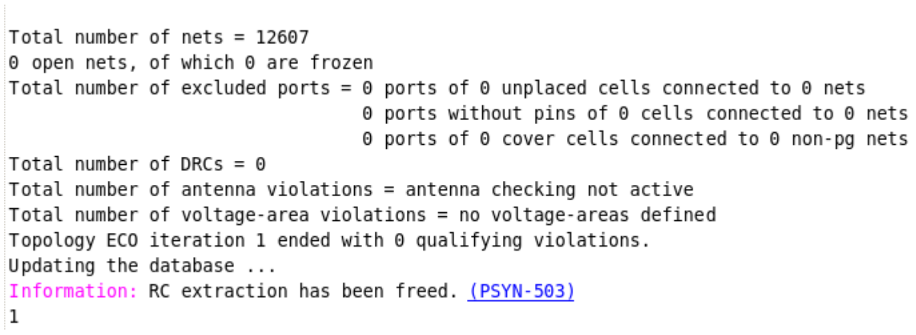
```
            Design Rules
            -----------------------------------
            Total Number of Nets:        12653
            Nets With Violations:            1
            Max Trans Violations:            0
            Max Cap Violations:              1
            -----------------------------------


            Hostname: E018A-02

            Compile CPU Statistics
            -----------------------------------------
            Resource Sharing:                    7.62
            Logic Optimization:                 25.78
            Mapping Optimization:              168.45
            -----------------------------------------
            Overall Compile Time:              210.47
            Overall Compile Wall Clock Time:   211.69

            --------------------------------------------------------------------

            Design  WNS: 0.00  TNS: 0.00  Number of Violating Paths: 0


            Design (Hold)  WNS: 0.00  TNS: 0.00  Number of Violating Paths: 0

            --------------------------------------------------------------------


          1
```

Figure 4.54: Part of QoR report after CTS



```
****************************************
Report : clock tree
Design : top
Version: O-2018.06-SP3
Date   : Thu Apr 14 20:19:51 2022
****************************************

Information: Float pin scale factor for the 'max' operating condition of scenario 'default' is set to 1.000 (CTS-375)

====================== Clock Tree Summary =======================
Clock                Sinks     CTBuffers ClkCells  Skew      LongestPath TotalDRC   BufferArea
---------------------------------------------------------------------------------------------
clk                  3104      29        29        0.0269    0.1112      2            287.1826
1
```

Figure 4.55: Clock tree summary



```
Total number of nets = 12607
0 open nets, of which 0 are frozen
Total number of excluded ports = 0 ports of 0 unplaced cells connected to 0 nets
                                 0 ports without pins of 0 cells connected to 0 nets
                                 0 ports of 0 cover cells connected to 0 non-pg nets
Total number of DRCs = 0
Total number of antenna violations = antenna checking not active
Total number of voltage-area violations = no voltage-areas defined
Topology ECO iteration 1 ended with 0 qualifying violations.
Updating the database ...
Information: RC extraction has been freed. (PSYN-503)
1
```

Figure 4.56: Status of clock tree routing

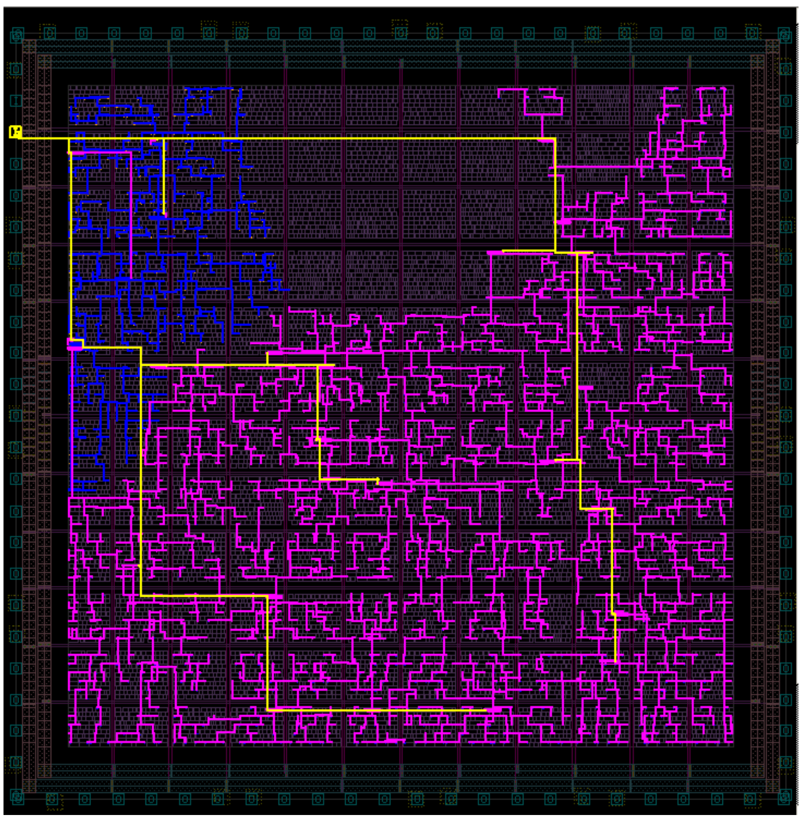
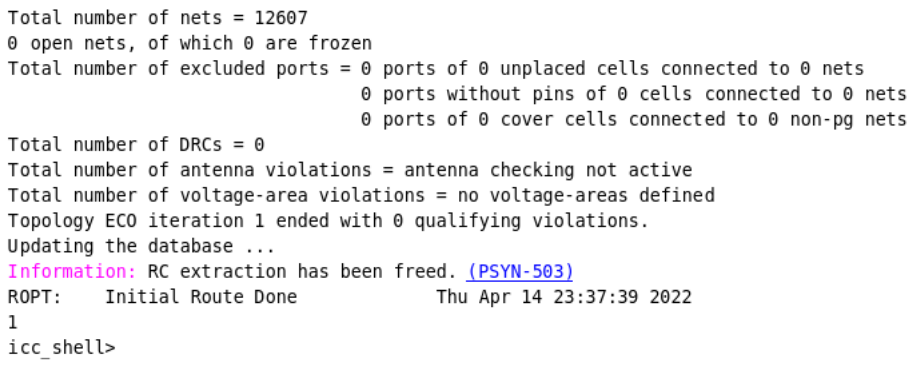


Figure 4.57: Visualisation of the clock tree

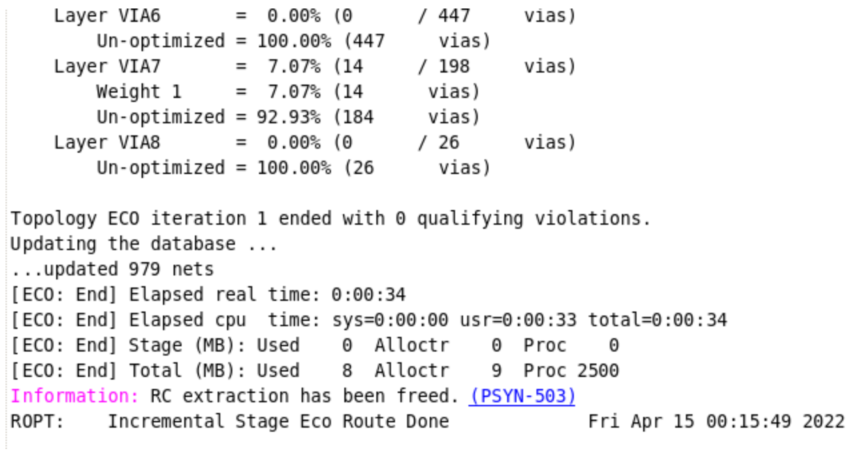
The design is ready for detailed routing after performing CTS. The detailed routing is separated into two stages. The first stage is the initial route for assignment of the track, global routing, and detail routing. There is no optimization in the initial routing. Figure 4.58 shows the result of the initial routing. The initial routing is successful and the sum of DRC errors is 0, showing no congestion issues exist in the design. As a result, 30% of the chip area is sufficient for routing. If there are DRC errors after initial routing, detailed routing and ECO routing should be performed to solve the violations. However, a floorplan should be generated using a smaller core utilization ratio to provide more spacing for routing when the DRC errors still occur after detailed routing and ECO routing. For this project, the post route optimization is hence performed on the design as the design is free from congestion issues after the initial route. The area recovery and power options are set high to further optimize these two parameters. The tool had updated 979 nets during the optimization as shown in Figure 4.59.

The LVS is performed to verify the LVS error. Figure 4.60 shows the LVS result after the initial routing. As shown in the result, there is a minor LVS error for 20 floating ports in the design. The floating ports are the ports without connecting to any nets. Based on the detail of floating ports as shown in Figure 4.61, the floating ports are the unused output ports of the registers. For a register, there are 2 output ports Q and QN in which QN is the inverting result of Q. Since some of the registers is using only output, the floating ports violation occurs. However, this violation will not affect the design functionality. Hence, the floating port violations can be ignored. There are no floating nets, shorted nets, or open nets in the layout. With no electrical equivalent error in the layout, the layout can be said to have the same interconnection as the optimized gate-level netlist produced during the logic synthesis stage. Meanwhile, the nets are routed correctly as defined in the gate-level netlist since there is no must joint error in the layout. Hence, the LVS result is acceptable.



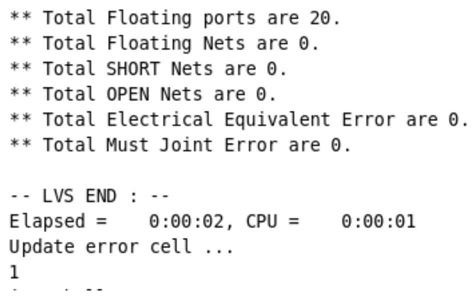
```
Total number of nets = 12607
0 open nets, of which 0 are frozen
Total number of excluded ports = 0 ports of 0 unplaced cells connected to 0 nets
                                 0 ports without pins of 0 cells connected to 0 nets
                                 0 ports of 0 cover cells connected to 0 non-pg nets
Total number of DRCs = 0
Total number of antenna violations = antenna checking not active
Total number of voltage-area violations = no voltage-areas defined
Topology ECO iteration 1 ended with 0 qualifying violations.
Updating the database ...
Information: RC extraction has been freed. (PSYN-503)
ROPT:    Initial Route Done            Thu Apr 14 23:37:39 2022
1
icc_shell>
```

Figure 4.58: Result of initial routing.



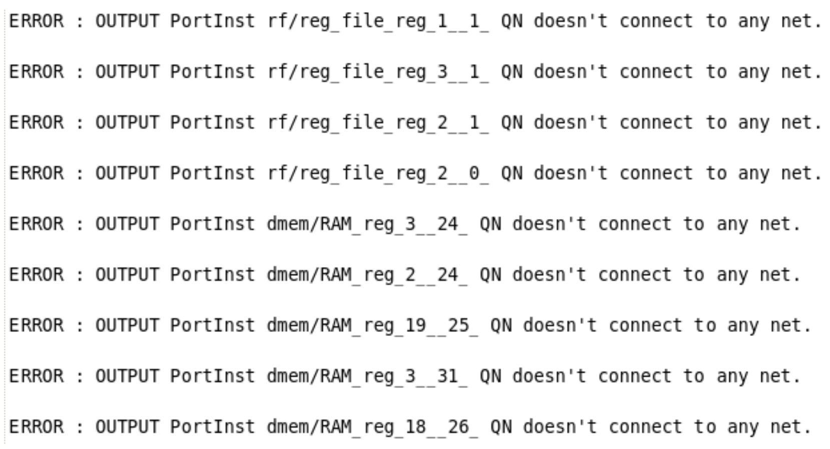
```
    Layer VIA6       =  0.00% (0      / 447     vias)
        Un-optimized = 100.00% (447     vias)
    Layer VIA7       =  7.07% (14     / 198     vias)
        Weight 1     =  7.07% (14      vias)
        Un-optimized = 92.93% (184     vias)
    Layer VIA8       =  0.00% (0      / 26      vias)
        Un-optimized = 100.00% (26      vias)

Topology ECO iteration 1 ended with 0 qualifying violations.
Updating the database ...
...updated 979 nets
[ECO: End] Elapsed real time: 0:00:34
[ECO: End] Elapsed cpu  time: sys=0:00:00 usr=0:00:33 total=0:00:34
[ECO: End] Stage (MB): Used    0  Alloctr    0  Proc    0
[ECO: End] Total (MB): Used    8  Alloctr    9  Proc 2500
Information: RC extraction has been freed. (PSYN-503)
ROPT:    Incremental Stage Eco Route Done            Fri Apr 15 00:15:49 2022
```

Figure 4.59: Result of postroute optimization



```
** Total Floating ports are 20.
** Total Floating Nets are 0.
** Total SHORT Nets are 0.
** Total OPEN Nets are 0.
** Total Electrical Equivalent Error are 0.
** Total Must Joint Error are 0.

-- LVS END : --
Elapsed =    0:00:02, CPU =    0:00:01
Update error cell ...
1
```

Figure 4.60: LVS after initial route



```
ERROR : OUTPUT PortInst rf/reg_file_reg_1__1_ QN doesn't connect to any net.

ERROR : OUTPUT PortInst rf/reg_file_reg_3__1_ QN doesn't connect to any net.

ERROR : OUTPUT PortInst rf/reg_file_reg_2__1_ QN doesn't connect to any net.

ERROR : OUTPUT PortInst rf/reg_file_reg_2__0_ QN doesn't connect to any net.

ERROR : OUTPUT PortInst dmem/RAM_reg_3__24_ QN doesn't connect to any net.

ERROR : OUTPUT PortInst dmem/RAM_reg_2__24_ QN doesn't connect to any net.

ERROR : OUTPUT PortInst dmem/RAM_reg_19__25_ QN doesn't connect to any net.

ERROR : OUTPUT PortInst dmem/RAM_reg_3__31_ QN doesn't connect to any net.

ERROR : OUTPUT PortInst dmem/RAM_reg_18__26_ QN doesn't connect to any net.
```

Figure 4.61: Detail of the floating ports

4.3.4. Chip Finishing and Tape Out

Chip finishing is the last stage in the physical design before tape out. The tool is used to reduce the critical area in this stage.

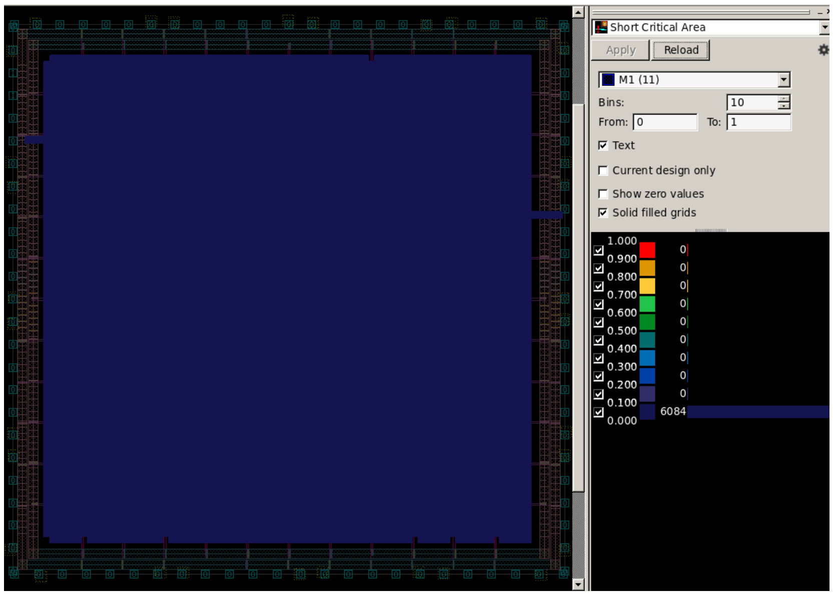
First, the short critical area map is generated using the tool as shown in Figure 4.62. The maximum threshold value is set as 1. From the map, it can be seen the short critical area is small in the overall layout. Similarly, the open critical area map with the highest threshold of 1 is generated as shown in Figure 4.63. The map distribution shows the maximum open critical area ratio is below 40%. Spreading of wires is performed to enhance the critical area for shorts while widening of wires is carried out to reduce open critical area. Figure 4.64 shows that metal 3 is pushed off from the track due to wire spreading. The dotted line is the track for routing. The effect of widening wire is shown in Figure 4.65. Some part of the Metal 2 becomes wider, and the nets are pushed away from the original track. By pushing the nets away from the original track, the space between the metal traces increases. Hence, the critical area reduces. The DRC and LVS are performed again to ensure the design is still free of error, excepting the minor floating port error in LVS. Next, the redundant vias are inserted into the layout. The redundant vias had increased by 3. The DRC and LVS are checked again. The final DRC and LVS results are shown in Figure 4.68 and Figure 4.69. There is no DRC error and additional LVS error.

The final timing, QoR, area, and power reports are generated and analysed. The timing report is generated before taping out to make sure the design is free from setup violations. As shown in Figure 4.70, the data arrives faster than the required time in the path with a maximum delay with a margin of 0.09 ns. There are no setup violations in the final design. Besides, the timing performance increases by 0.05 ns from the optimized gate-level netlist with a margin of 0.04 ns. It is because the clock delay is stabilised in CTS. In the logic synthesis stage, the clock delay is an estimation of 0.20 ns and there is an uncertainty of 0.34 ns applied to the design. The post-CTS design has a stabilized clock delay of 0.06 ns only and the uncertainty is removed. Hence, the time required by the data increases by 0.20 ns from 3.22 ns to 3.42 ns as compared to the optimized netlist. Due to the reduction of clock skew after CTS, the final data arrival time had also increased by 0.16 ns from 3.18 ns to 3.34 ns. Since the increase in data arrival time is smaller than the data required time, the slack or margin improves. The QoR report is also generated to examine the hold time performance of the final layout as shown in Figure 4.73. The report shows there is no path to experiencing hold time violation. Hence, the design is proved to be free from timing issues.

Aside from that, the area report of the final design is generated through ICC. The final area reports are shown in Figure 4.71. The total chip area had increased by 15.28 % to 72852.436196 µm^2. In order to avoid DRC errors, the metal nets are set to have a minimum distance between each other in the design. Besides, the power nets are included in the physical design. The cells and route tracks leave a distance from them to prevent design rule error as the power nets are set to have complete blockage for routing during preroute stage. The metal nets used for routing, power rings, and power nets have a certain width. Thus, in physical design, the chip area generated in the floorplan stage reserve a 30% area for routing. This causes the total chip area to increase as the area is fitting all the cells and the metal nets used for routing. Next, the power analysis of the final design is done based on the power report as shown in Figure 4.72. The power dissipated by net switching increases significantly from 134.4276 µW to 1.2951 mW. This causes the total dynamic power to rise as well. Due to the significant increase in net switching power, its contribution to total dynamic power increases by about 15 %. The cell internal power increases slightly to 5.7733 mW while the cell leakage power reduces slightly to 18.0631 mW. As a result, the total power dissipated by the design, in the end, is 25.131 mW with an increase of 7.28 %

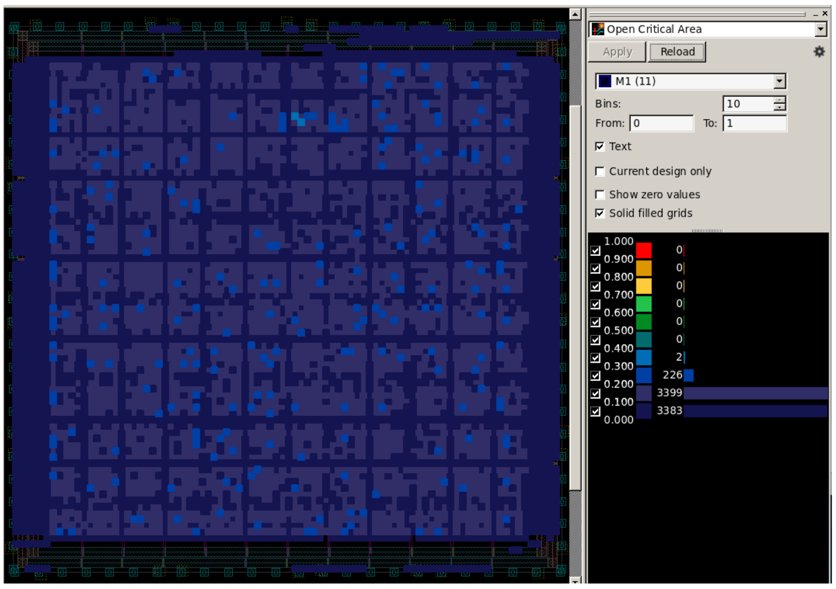
from the previous area. There are more switching nets when the clock tree nets increase. This leads to the rise in switching power as more nets will be discharging or charging the internal capacitance. The internal capacitance of the nets grows with the number of nets. Hence, the switching power became larger. The total power is within the power budget of 26 mW set in PNS. Hence, the design is still able to support the power consumption.

Finally, the design is ready for tape out. The GDSII file is produced by the tool to be used for fabrication. For this project, the design is not suitable to be fabricated duet to the educational libraries used in the software. The result of outputting the GDSII file is shown in Figure 4.74. The final layout of the design is shown in Figure 4.75.


Short Critical Area
Apply Reload
M1 (11)
Bins: 10
From: 0 To: 1
☑ Text
☐ Current design only
☐ Show zero values
☑ Solid filled grids

```
1.000
0.900  0
0.800  0
0.700  0
0.600  0
0.500  0
0.400  0
0.300  0
0.200  0
0.100  0
0.000  6084
```

Figure 4.62: Short critical area of the design



Open Critical Area

Apply Reload

M1 (11)

Bins: 10

From: 0 To: 1

☑ Text

☐ Current design only

☐ Show zero values

☑ Solid filled grids

1.000	🟥	0
0.900	🟧	0
0.800	🟨	0
0.700	🟩	0
0.600	🟩	0
0.500	🟦	0
0.400	🟦	2
0.300	🟦	226
0.200	🟦	3399
0.100	🟦	3383
0.000		

Figure 4.63: Open critical area

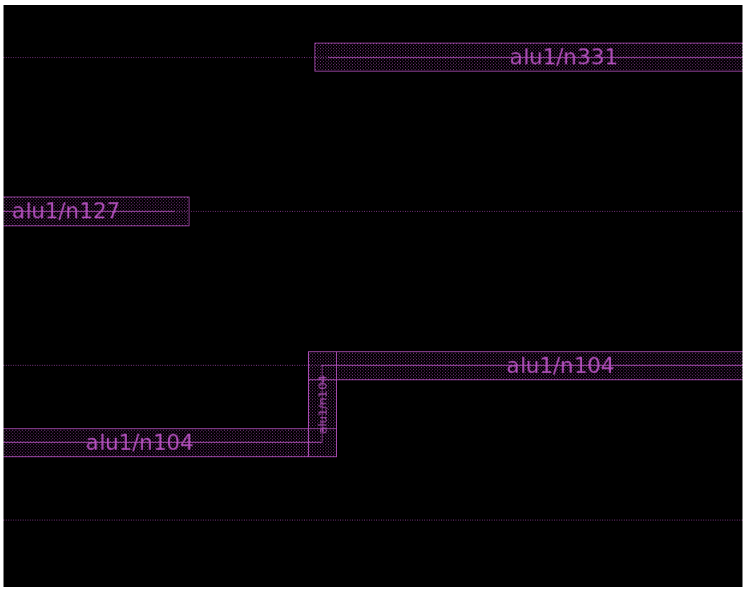


Figure 4.64: Metal 3 to be pushed off from the track

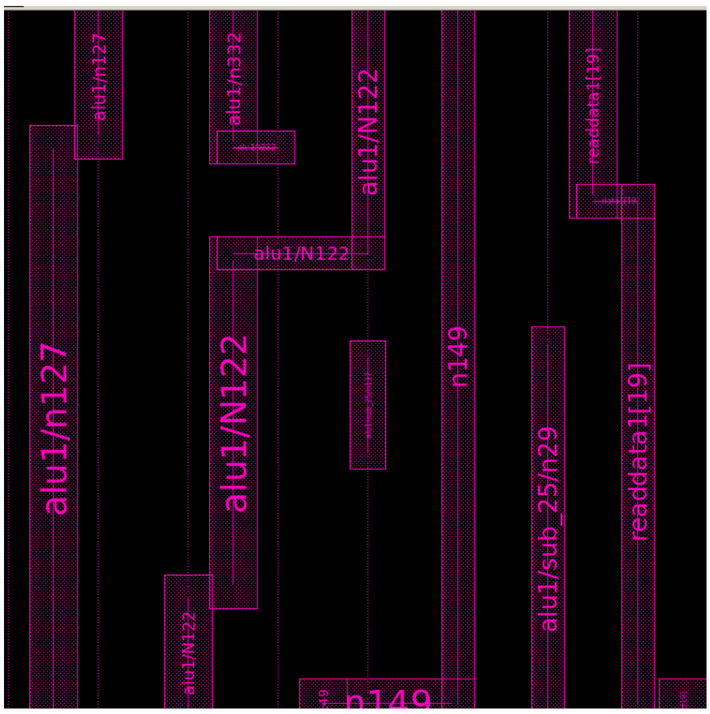
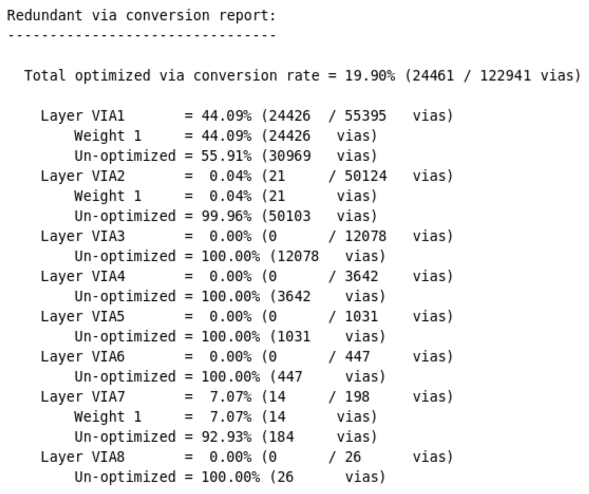
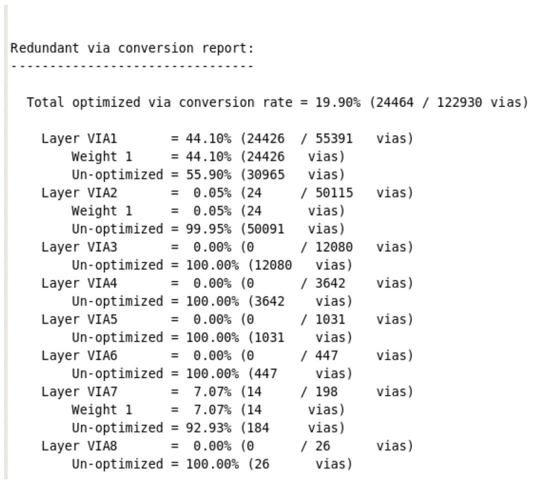


Figure 4.65: Metal 2 to become widen and pushed off from track



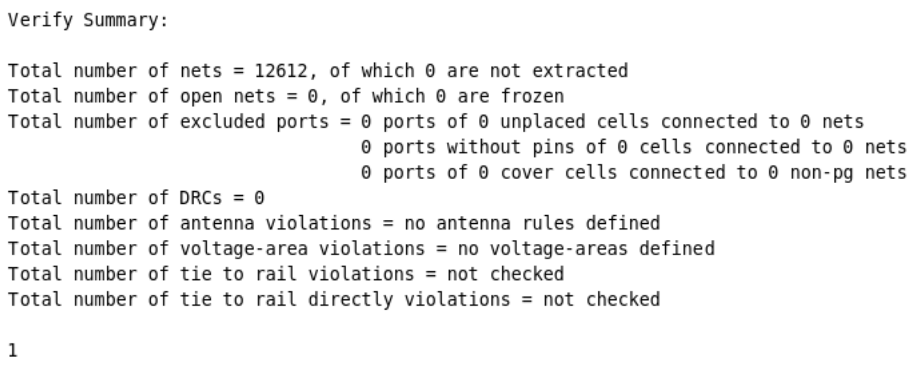
```
Redundant via conversion report:
--------------------------------

  Total optimized via conversion rate = 19.90% (24461 / 122941 vias)

    Layer VIA1       = 44.09% (24426   / 55395   vias)
        Weight 1     = 44.09% (24426   vias)
        Un-optimized = 55.91% (30969   vias)
    Layer VIA2       =  0.04% (21      / 50124   vias)
        Weight 1     =  0.04% (21      vias)
        Un-optimized = 99.96% (50103   vias)
    Layer VIA3       =  0.00% (0       / 12078   vias)
        Un-optimized = 100.00% (12078   vias)
    Layer VIA4       =  0.00% (0       / 3642    vias)
        Un-optimized = 100.00% (3642    vias)
    Layer VIA5       =  0.00% (0       / 1031    vias)
        Un-optimized = 100.00% (1031    vias)
    Layer VIA6       =  0.00% (0       / 447     vias)
        Un-optimized = 100.00% (447     vias)
    Layer VIA7       =  7.07% (14      / 198     vias)
        Weight 1     =  7.07% (14      vias)
        Un-optimized = 92.93% (184     vias)
    Layer VIA8       =  0.00% (0       / 26      vias)
        Un-optimized = 100.00% (26      vias)
```

Figure 4.66: Redundant conversion report before chip finishing



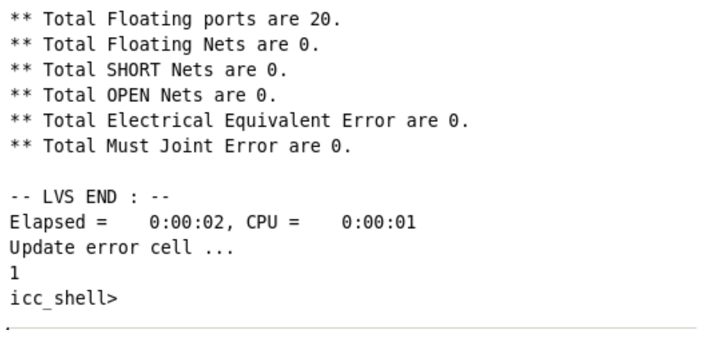
```
Redundant via conversion report:
--------------------------------

  Total optimized via conversion rate = 19.90% (24464 / 122930 vias)

    Layer VIA1       = 44.10% (24426  / 55391   vias)
        Weight 1     = 44.10% (24426   vias)
        Un-optimized = 55.90% (30965   vias)
    Layer VIA2       =  0.05% (24     / 50115   vias)
        Weight 1     =  0.05% (24      vias)
        Un-optimized = 99.95% (50091   vias)
    Layer VIA3       =  0.00% (0      / 12080   vias)
        Un-optimized = 100.00% (12080   vias)
    Layer VIA4       =  0.00% (0      / 3642    vias)
        Un-optimized = 100.00% (3642    vias)
    Layer VIA5       =  0.00% (0      / 1031    vias)
        Un-optimized = 100.00% (1031    vias)
    Layer VIA6       =  0.00% (0      / 447     vias)
        Un-optimized = 100.00% (447     vias)
    Layer VIA7       =  7.07% (14     / 198     vias)
        Weight 1     =  7.07% (14      vias)
        Un-optimized = 92.93% (184     vias)
    Layer VIA8       =  0.00% (0      / 26      vias)
        Un-optimized = 100.00% (26      vias)
```

Figure 4.67: Redundant conversion report after chip finishing



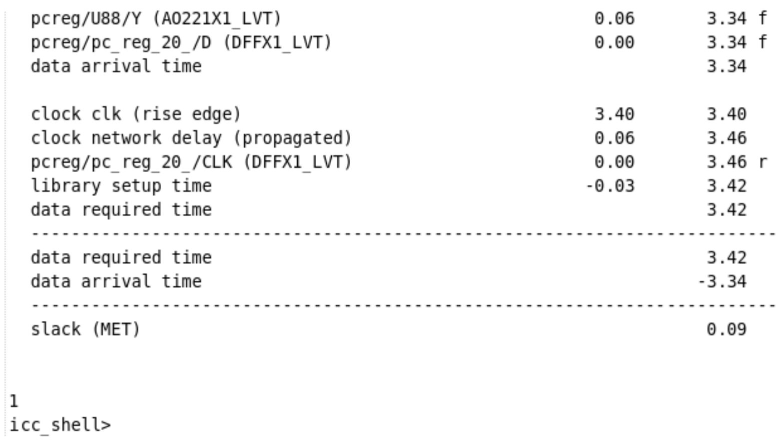
```
Verify Summary:

Total number of nets = 12612, of which 0 are not extracted
Total number of open nets = 0, of which 0 are frozen
Total number of excluded ports = 0 ports of 0 unplaced cells connected to 0 nets
                                 0 ports without pins of 0 cells connected to 0 nets
                                 0 ports of 0 cover cells connected to 0 non-pg nets
Total number of DRCs = 0
Total number of antenna violations = no antenna rules defined
Total number of voltage-area violations = no voltage-areas defined
Total number of tie to rail violations = not checked
Total number of tie to rail directly violations = not checked

1
```

Figure 4.68: Final DRC verification result



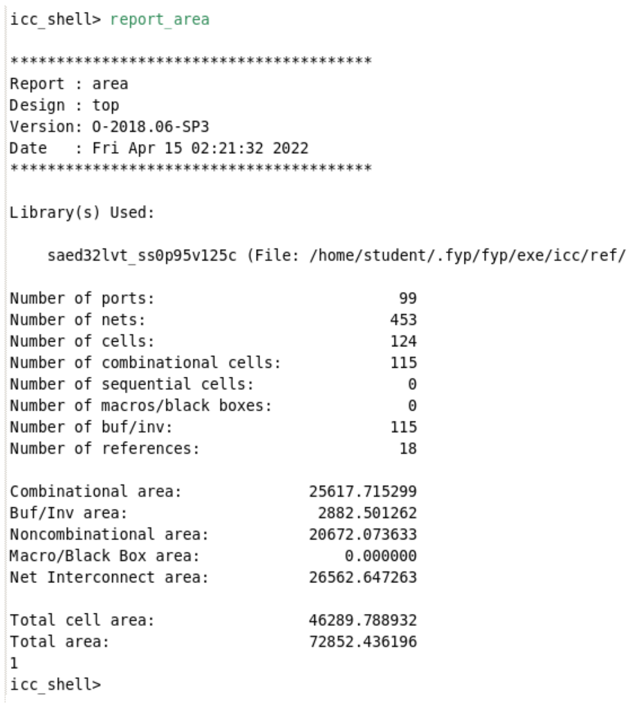
```
** Total Floating ports are 20.
** Total Floating Nets are 0.
** Total SHORT Nets are 0.
** Total OPEN Nets are 0.
** Total Electrical Equivalent Error are 0.
** Total Must Joint Error are 0.

-- LVS END : --
Elapsed =    0:00:02, CPU =    0:00:01
Update error cell ...
1
icc_shell>
```

Figure 4.69: Final LVS verification result



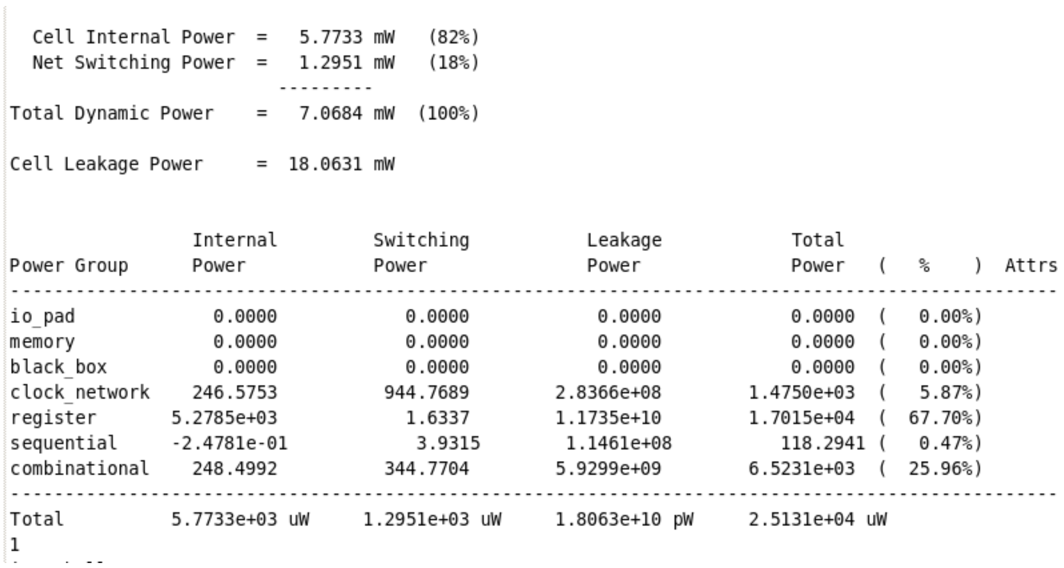
```
  pcreg/U88/Y (AO221X1_LVT)                              0.06       3.34 f
  pcreg/pc_reg_20_/D (DFFX1_LVT)                         0.00       3.34 f
  data arrival time                                                 3.34

  clock clk (rise edge)                                  3.40       3.40
  clock network delay (propagated)                       0.06       3.46
  pcreg/pc_reg_20_/CLK (DFFX1_LVT)                       0.00       3.46 r
  library setup time                                    -0.03       3.42
  data required time                                                3.42
  ---------------------------------------------------------------------------
  data required time                                                3.42
  data arrival time                                                -3.34
  ---------------------------------------------------------------------------
  slack (MET)                                                       0.09


1
icc_shell>
```

Figure 4.70: Final timing report



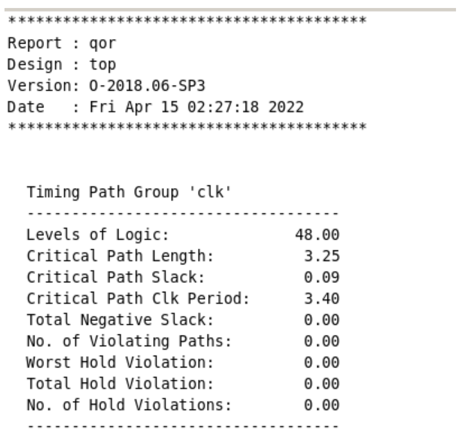
```
icc_shell> report_area

****************************************
Report : area
Design : top
Version: O-2018.06-SP3
Date   : Fri Apr 15 02:21:32 2022
****************************************

Library(s) Used:

    saed32lvt_ss0p95v125c (File: /home/student/.fyp/fyp/exe/icc/ref/

Number of ports:                           99
Number of nets:                           453
Number of cells:                          124
Number of combinational cells:            115
Number of sequential cells:                 0
Number of macros/black boxes:               0
Number of buf/inv:                        115
Number of references:                      18

Combinational area:             25617.715299
Buf/Inv area:                    2882.501262
Noncombinational area:          20672.073633
Macro/Black Box area:               0.000000
Net Interconnect area:          26562.647263

Total cell area:                46289.788932
Total area:                     72852.436196
1
icc_shell>
```

Figure 4.71: Final area report



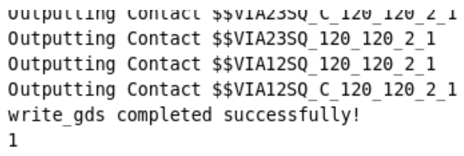
```
  Cell Internal Power  =   5.7733 mW   (82%)
  Net Switching Power  =   1.2951 mW   (18%)
                         ---------
Total Dynamic Power    =   7.0684 mW  (100%)

Cell Leakage Power     =  18.0631 mW


                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )  Attrs
--------------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000            0.0000             0.0000  (   0.00%)
memory             0.0000           0.0000            0.0000             0.0000  (   0.00%)
black_box          0.0000           0.0000            0.0000             0.0000  (   0.00%)
clock_network    246.5753         944.7689        2.8366e+08         1.4750e+03  (   5.87%)
register       5.2785e+03           1.6337        1.1735e+10         1.7015e+04  (  67.70%)
sequential     -2.4781e-01          3.9315         1.1461e+08          118.2941 (   0.47%)
combinational    248.4992         344.7704        5.9299e+09         6.5231e+03  (  25.96%)
--------------------------------------------------------------------------------------------------
Total          5.7733e+03 uW     1.2951e+03 uW     1.8063e+10 pW     2.5131e+04 uW
1
```

Figure 4.72: Final power report



```
****************************************
Report : qor
Design : top
Version: O-2018.06-SP3
Date   : Fri Apr 15 02:27:18 2022
****************************************


  Timing Path Group 'clk'
  -----------------------------------
  Levels of Logic:             48.00
  Critical Path Length:         3.25
  Critical Path Slack:          0.09
  Critical Path Clk Period:     3.40
  Total Negative Slack:         0.00
  No. of Violating Paths:       0.00
  Worst Hold Violation:         0.00
  Total Hold Violation:         0.00
  No. of Hold Violations:       0.00
  -----------------------------------
```

Figure 4.73: QoR report showing hold time violation information



```
Outputting Contact $$VIA23SQ_C_120_120_2_1
Outputting Contact $$VIA23SQ_120_120_2_1
Outputting Contact $$VIA12SQ_120_120_2_1
Outputting Contact $$VIA12SQ_C_120_120_2_1
write_gds completed successfully!
1
```

Figure 4.74: Result of outputting GDSII file

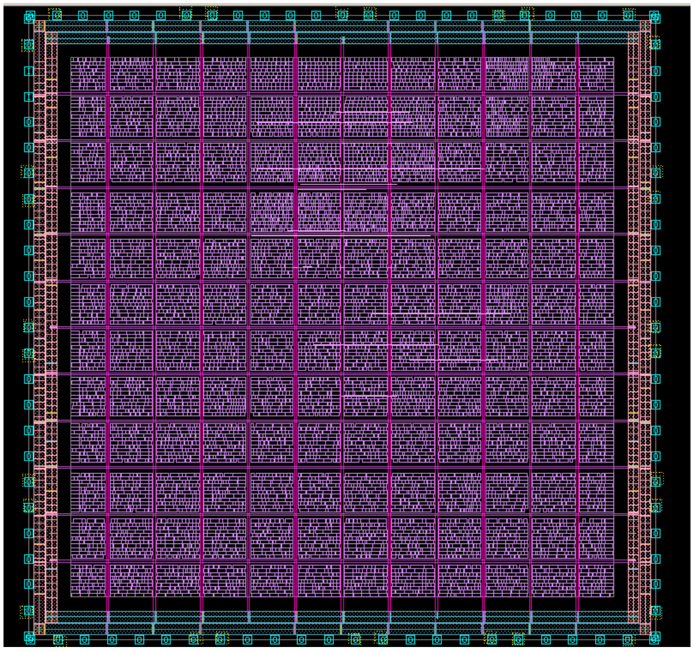


Figure 4.75: Final layout

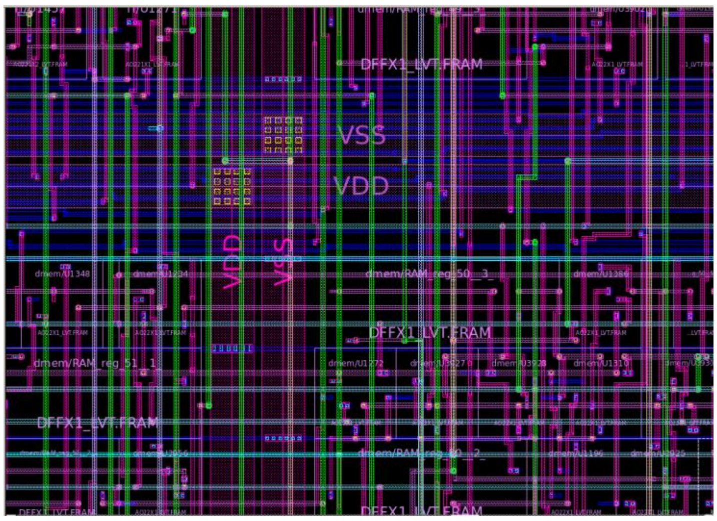


Figure 4.76: Enlarged view of the layout

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1. Conclusion

The objectives are achieved. In this project, the front-end and back-end design flow from RTL coding to tape out is completed successfully. A 32-bit MIPS processor with support for 29 instructions is developed and its functionality is verified. The complete VLSI design flow is implemented using EDA tools provided by Synopsys. The Synopsys DC is used to implement the logic synthesis stage in the front-end while the Synopsys ICC is to implement the physical design processes for back-end design flow. In logic synthesis, DC is used to map and optimize the design using the 32 nm libraries. At the same time, the software provided the automation for STA to analyse the setup time performance in this stage. Besides, the tool is used to generate reports to analyse the area and power of the design after mapping and optimization. This greatly reduced the time required for logic synthesis as the DC can map the Boolean equations in the RTL source codes to the gates in the provided libraries. While for ICC, it is used to carry out all the stages in back-end design with a built-in STA analyzer to analyse the timing. The algorithm of the software reduces the time consumption to provide a design layout that is free from timing issues. There are plenty of logic gates in the design. Manual execution of complete VLSI design flow is too complex for a human. Hence, the implementation of the VLSI design flow requires the help of the EDA tool to reduce the time to produce an SoC.

After completing the front-end design flow, the mapped and optimized design is verified to have no setup violation. The hold violations that exist in this stage are left to be solved in the physical design. The design is driven by a clock with a frequency of around 294 MHz. The timing constraints are applied to set the input delay, output delay, setup uncertainty and hold uncertainty during logic synthesis stage to obtain a more accurate timing performance. The optimized gate-level netlist is checked to ensure its functionality. The design is modified and optimized without changing the functionality.

The optimized gate-level netlist is used to perform physical design. The hold time violations are fixed in this stage through CTS. At the same time, the setup time performance is improved slightly to the margin of 0.09 ns. However, the power dissipated by the design becomes larger due to the increase in net switching power in physical design. The high frequency of the clock used in the design had caused the switching power to increase. Besides, the large number of registers used for RAM and ROM in the design had contributed to the large power consumption. In conclusion, the power performance of the design is scarified to optimize the timing performance.

5.2. Recommendation

One of the recommendations is to further improve the area performance and efficiency of the design using smaller transistor technology. The technology used in this project is 32 nm while 5 nm is the latest technology in the industry. There is room for improvement in terms of reducing transistor size. By using a smaller transistor, the chip area can be reduced as the same amounts of transistors can be placed in a smaller area. Thus, the cost to produce a chip can be reduced. Besides, by having more transistors in the chip, the processor becomes more powerful to execute instructions faster. A reduction in transistor size leads to a drop in its internal capacitance. The power required to drive the transistor is hence smaller. Thus, the smaller transistor is more power-efficient.

The margin for setup time for this design is 0.09 ns. The is a possibility for the design to have setup violations during the manufacturing process. The design can be improved by taking the margin for setup into account. In this project, the design is not available for fabrication. However, it is important for a designer to have the practice to leave a safety margin for the manufacturing of the product. The margin can be improved by adjusting the timing constraints applied to the design during DC.

Besides, RAM and ROM should be designed as macro cells. The memory elements designed as macro cells can be optimized separately to achieve better timing and power performance. In the current design, the RAM and ROM are optimized as a group with the other submodules. This causes the register to dissipate large power and results in high power consumption in overall. By generating macrocells for RAM and ROM, they are optimized individually with specific power constraints applied to them. Thus, the power consumed by the RAM and ROM can be reduced, resulting in a lower power dissipation of the overall design. The macro cells can be generated using another EDA tool from Synopsys named Memory Compiler.

The current design is made without I/O pads. This causes the design to be unable to interface with other devices through external pins. In order to improve the design, the RTL codes should be modified to include the implementation of I/O pads to allow interfacing of external devices. Aside from that, the current design is a single cycle MIPS. The design can be modified into a pipelined design to improve efficiency. For a single cycle process, only 1 instruction is executed in a clock cycle. While for pipelined design, a single cycle processor is divided into several stages to execute instructions simultaneously. For a 5-stage pipelined processor, 5 instructions can be executed within a clock cycle in parallel. Hence, the processors can perform faster in terms of increasing the efficiency to execute the instructions.

REFERENCE

Abd-El-Barr, M., and El-Rewini, H., 2005. *Fundamentals of computer organization and architecture.* New Jersey: John Wiley & Sons, Inc.

Bhatnagar, H, 2002. *Advanced ASIC chip synthesis: using Synopsys Design Compiler Physical Compiler and Prime Time.* 2 nd ed. New York: Kluwer Academic Publishers.

Chen, W., K., 2009. *The circuits and filters handbook: computer aided design and design automation.* 3rd ed. Boca Raton: CRC Press.

Das, D., 2010. *VLSI design.* New Delhi: Oxford University Press.

Gianfagna, M., 2021. *What is electronic design automation?* [online] Available at: < https://www.synopsys.com/glossary/what-is-electronic-design-automation.html> [Accessed 9 August 2021].

Harris, D. M., and Harris, S. L., 2013. *Digital design and computer architecture.* 2nd ed. Waltham: Elsevier, Inc.

Hennessy, J., L., Jouppi, N., Przybylski, S., Rowen, C., Gross, T., Baskett, F., and Gill, J., 1982. MIPS: A microprocessor architecture. *ACM SIGMICRO Newsletter,* [e-journal] 13(4). pp 17 -22. https://dl.acm.org/doi/10.1145/1014194.800930

Jamil, T., 1995. RISC versus CISC. *IEEE Potentials,* [e-journal] 14(3). pp 13-16. https://doi.org/10.1109/45.464688

Kahng, A., B., Lienig, J., Markov, I., L., and Hu, J., 2011. *VLSI physical design: from graph partitioning to timing closure.* Dordrecht: Springer.

King, I., Wu, D., and Pogkas, D., 2021. How a chip shortage snarled everything from phones to cars. *Bloomberg.* [online] 29 March. Available at: < https://www.bloomberg.com/graphics/2021-semiconductors-chips-shortage/> [Accessed 9 August 2021].

Kishore, K., L., and Prabhakar, V., S., V., 2009. *VLSI design*. New Delhi: International Publishing House Pvt. Ltd.

Kowalski, T., J., Geiger, D., J., Wolf, W., H., and Fichtner, W., 1985. The VLSI design automation assistant: from algorithms to silicon. *IEEE Design & Test of Computers,* [e-journal] 2(4). pp 33-43. https://doi.org/10.1109/MDT.1985.294721

Lamie, E., L., 2009. *Real-time embedded multithreading: using ThreadX and MIPS.* Oxford: Elsevier

Luo, T., C., Leong, E., Chao, M., C., T., Fisher, P., A. and Chang, W., H., 2010. Mask versus schematic – an enhanced design-verification flow for first silicon success. *2020 IEEE International Test Conference,* [e-journal] pp 1-9. pp. 1-9. https://doi.org/10.1109/TEST.2010.5699238

Patterson, D., A., and Hennssy, J., L., 2014. *Computer organization and design: the hardware/software interface.* 5th ed. Waltham: Elsevier, Inc.

Ramachandran, S., 2007. *Digital VLSI systems design: a design manual of implementation of projects on FPGAs and ASICs using Verilog.* Dordrecht: Springer.

SIA, 2021. *Global semiconductor sales increase 1.9% month-to-month in April; Annual sales projected to increase 19.7% in 2021, 8.8% I 2022.* [online] Available at: < https://www.semiconductors.org/global-semiconductor-sales-increase-1-9-month-to-month-in-april-annual-sales-projected-to-increase-19-7-in-2021-8-8-in-2021/> [Accessed 9 August 2021].

Silbert, S., 2012. *$125 MIPS-based Smart Tab 1 brings Jelly Bean on budget to India.* [online] Available at: https://www.engadget.com/2012-07-31-125-mips-based-smart-tab-1-jelly-bean.html [Accessed 9 August 2021].

Takahashi, D., 2005. Forty years of Moore's Law. *The Seattle Times.* [online] 18 April. Available at: <https://www.seattletimes.com/business/forty-years-of-moores-law/> [Accessed 9 August 2021].

TSMC, 2020. *Logic Technology*. [online] Available at: <https://www.tsmc.com/english/dedicatedFoundry/technology/logic> [Accessed 9 August 2021].

Voica, A., 2016. *Five most iconic devices to use MIPS CPUs.* [online] Available at: < https://www.mips.com/blog/five-most-iconic-devices-to-use-mips-cpus/> [Accessed 9 August 2021].

Walker, D., M., H., 1992. Critical area analysis. *Proceedings International Conference on Wafer Scale Integration.* [e-journal] pp 281-290. https://doi.org/10.1109/ICWSI.1992.171820

Xiu, L., M., 2008. *VLSI circuit design methodology demystified: A conceptual taxonomy.* New Jersey: John Wiley & Sons, Inc.

APPENDICES

APPENDIX A: Instruction loaded into the design

```
20020005 //addi $2, $0, 5
2003000C //addi $3, $0, 12
2067fff7 //addi $7, $0, -9
00E22025 //or $4, $7, $2
00642824 //and $5, $3, $4
00A72826 //xor $5, $5, $7
00A43020 //add $6, $5, $4
10C70010 //beq $6, $7, end
0064202A //slt $4, $3, $4
10800001 //beq $4, $0, around
20050000 //addi $5, $0, 0
00E2202A //slt $4, $7, $3
00853820 //add $7, $4, $5
00E23822 //sub $7, $7, $2
AC670044 //sw $7, 68 ($3)
8C020050 //lw $2, 80 ($0)
30489616 //andi $8, $2, 38422
35099600 //ori $9, $8, 38400
392A9614 //xori $10, $9, 38420
000A6100 //sll $12, $10, 4
3C0B0004 //lui $11, 4
016C6022 //sub $12, $11, $12
08000018 //J end
```

```
20020001 //addi $2, 0, $1
AC0C0054 //sw $12, 84(0)
22120008 //addi s2 s0 8
3C11F300 //lui s1 0xf300
22310022 //addi s1 s1 0x0022
2A57000A //slti s7 s2 10
16E40002 //bne s7, $4, 2
16320001 //bne s1, s2, 2
3C11F300 //lui s1 0xf300
0C00002A //jal 42
02519807 //SRAv s3 s1 s2 (s3 = -851968)
02F3A006 //SRLV $s4 $s3 $s7 (s4 = 2147057664)
0254A004 //SllV $s4 $s4 $s2 (s4 = -109051904)
AC140058 //sw s4, 88(r0) (ram[22] = 2]
1A800005 //blez s4 0x0005
1E800001 //bgtz s4, 1
AC16005C //sw s6, 92(r0) (ram[23] = 0]
02D5B827 //nor s7, s6,s5 ( ans = -153)
AE170060 //sw s7, 96(r0) (ram[24] = -153)
03E00008 //jr ra return to pc == 44
0011B083 //sra s6 s1 0x0002
0016B280 //sll s6 s6 0x000A
0016B382 //srl s6 s6 0x000E
1AC00001 //blez s6 0x0001
3C11F300 //lui s1 0xf300
22150098 //addi s5 s0 152
1EA00001 //bgtz s5 0x0001
3C11F300 //lui s1 0xf300
02A0F809 //jalr ra s5
00000000
```

APPENDIX B: Testbench for verification

```
`timescale 1ns/1ps
module tb_fyp();
  logic clk;
  logic reset, error;
  logic [2:0] check;

  logic [31:0] readdata2, writedata, dataaddr;
  logic memwrite;

  // instantiate device to be tested
  top dut (clk, reset, readdata2, writedata, dataaddr, memwrite);

  // initialize test
  initial
    begin
      check <= 0; error <= 0;
      reset <= 1; # 22; reset <= 0;
    end

  // generate clock to sequence tests
  always
    begin
      clk <= 1; # 5; clk <= 0; # 5;
    end

  // check results
  always @(negedge clk)
    begin
      if (memwrite) begin
    if (dataaddr === 80 & readdata2 === 3) begin
      $display("Arithmetic calculation succeeded");
      check = check +3'b001;
    end
    else if (dataaddr === 84 & readdata2 === 261792) begin
      $display("Logical calculation succeeded");
      check = check +3'b001;
    end
     else if (dataaddr === 88 & readdata2 === -109051904) begin
      $display("Variable rotation succeeded");
      check = check +3'b001;
    end
    else if (dataaddr === 92 & readdata2 === 0) begin
      $display("Rotation succeeded");
    check = check +3'b001;
    end
    else if (dataaddr === 96 & readdata2 === -153) begin
    $display("Logical NOR calculation succeeded");
    check = check +3'b001;
    end
    else begin
      error = 1'b1;
      end
  end

  if(error)begin
   $display("\nError!!");
  $display ("Time %d, dmen addr(*2) = %b, mem write data = %b",
  $realtime, dataaddr, readdata2);
  #20;
  $stop;
end
else if (check == 3'b101 && error == 0)begin
  $display("\nSimulation Sucess!!");
```

```
      #20;
      $finish;
    end
  end
endmodule
```

APPENDIX C: Setup file for DC

```
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
#  Library Setup
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

set search_path    "$search_path scripts ../ref/saed32nm/lib/stdcell_lvt/db_nldm"
set target_library "saed32lvt_ss0p95v125c.db"
set_app_var synthetic_library dw_foundation.sldb
set link_library   "* saed32lvt_ss0p95v125c.db $synthetic_library"
set symbol_library "generic.sdb"

############ Do NOT edit below this line ############
####################################################

echo "\n\nSettings:"
echo "search_path:       $search_path"
echo "link_library:      $link_library"
echo "target_library:    $target_library"
echo "symbol_library:    $symbol_library"


define_design_lib DEFAULT -path ./analyzed

# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
#  History
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

history keep 1000
```

```
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
#  Aliases
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

alias h history
alias rc "report_constraint -all_violators"
alias rt report_timing
alias ra report_area
alias page_on {set sh_enable_page_mode true}
alias page_off {set sh_enable_page_mode false}
alias fr "remove_design -designs"

# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
#  Alib for compile_ultra
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

# set alib_library_analysis_path [get_unix_variable HOME]
set alib_library_analysis_path ..

#read the files
analyze -format sverilog { top.sv sl2.sv alu32.sv pc.sv imem.sv regfile.sv  aludec.sv
dmem.sv maindec.sv signext.sv}

#link top
elaborate top

#compile variable
set write_name_nets_same_as_ports true
set compile_advanced_fix_multiple_ports_nets true
set compile_delete_unloaded_sequential_cells true

echo "\n\nI am ready...\n"
```

APPENDIX D: Setup file for ICC

```
set LIB_ROOT ../ref/saed32nm/lib
set hdlin_enable_upf_naming_compatibility true

set LVT_LIB " \
        saed32lvt_ss0p95v125c.db \
        "



set LVT_LIB_SEARCH_PATH "$LIB_ROOT/stdcell_lvt/db_nldm"

set search_path "$search_path \
                $LVT_LIB_SEARCH_PATH "

set_app_var target_library $LVT_LIB
set_app_var synthetic_library dw_foundation.sldb
set_app_var link_library "* $target_library $synthetic_library"
define_design_lib default -path ./work

set mw_design_library MW_MIPS
set mw_reference_library "$LIB_ROOT/stdcell_lvt/milkyway/saed32nm_lvt_1p9m "

set TECH_FILE "$LIB_ROOT/tech/milkyway/saed32nm_1p9m_mw.tf"
set MAP_FILE "$LIB_ROOT/tech/star_rc/saed32nm_tf_itf_tluplus.map"
set                                                      TLUPLUS_MAX_FILE
"$LIB_ROOT/tech/star_rc/saed32nm_1p9m_Cmax.tluplus"
set                                                      TLUPLUS_MIN_FILE
"$LIB_ROOT/tech/star_rc/saed32nm_1p9m_Cmin.tluplus"

set MW_POWER_NET      "VDD"
set MW_POWER_PORT      "VDD"
set MW_GROUND_NET      "VSS"
set MW_GROUND_PORT     "VSS"
set MIN_ROUTING_LAYER  "M1"
set MAX_ROUTING_LAYER  "M6"
set upf_create_implicit_supply_sets false

if { [shell_is_in_topographical_mode] || ($::synopsys_program_name=="icc_shell") }
{
    if {![file isdirectory $mw_design_library ]} {
      create_mw_lib   -technology $TECH_FILE \
                      -mw_reference_library $mw_reference_library \
                      $mw_design_library
    } else {
      set_mw_lib_reference $mw_design_library \
        -mw_reference_library $mw_reference_library
    }

open_mw_lib $mw_design_library
set_tlu_plus_files -max_tluplus $TLUPLUS_MAX_FILE \
                       -min_tluplus $TLUPLUS_MIN_FILE \
                       -tech2itf_map $MAP_FILE
}
```