

**OPTIMIZATION OF AN INTEGRATED CIRCUIT DEVICE BY
IMPROVING ITS VLSI DESIGN FROM RTL TO GDSII**

THEE KANG WEI

**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**

January 2016

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 : _____

Name : _____

ID No. : _____

Date : _____

APPROVAL FOR SUBMISSION

I certify that this project report entitled **“OPTIMIZATION OF AN INTEGRATED CIRCUIT DEVICEBY IMPROVING ITS VLSI DESIGN FROM RTL TO GDSII”** was prepared by **THEE KANG WEI** 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 : _____

Supervisor: Dr. Yeap Kim Ho

Date : _____

The copyright of this report belongs to the author under the terms of the copyright Act 1987 as qualified by Intellectual Property Policy of Universiti Tunku Abdul Rahman. Due acknowledgement shall always be made of the use of any material contained in, or derived from, this report.

© 2016, Thee Kang Wei. All right reserved.

ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, Dr. Yeap Kim Ho for his invaluable advice, guidance and his enormous patience throughout the development of the research.

Besides, I would like to thank to my project Co-supervisor, Mr. Cheah Hun Wah for his useful advices and guidance throughout the development of the project.

Moreover, I would like to thank to Mr. Sree for his technical guidance throughout the development of the project.

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

**OPTIMIZATION OF AN INTEGRATED CIRCUIT DEVICE BY
IMPROVING ITS VLSI DESIGN FROM RTL TO GDSII**

ABSTRACT

VLSI design flow from RTL to GDSII consists of two phases, namely front-end design and back-end design. In this project, the front-end design and back-end design were done in order to improve and optimize an 8051 microcontroller-based core. Logic synthesis, physical design, physical verification and others are done by using EDA tools, namely Synopsys Design Compiler and Synopsys IC Compiler. EDA tools provide the design automations for IC design process which can reduce the design TAT. In order to reduce the design cost, the chip-area is reduced as small as possible. The performance of the chip is improved by 10 times of its original clock frequency. Most of the violations that exist in the design are fixed. The optimized gate-level netlist is generated by Design Compiler in ddc format. The final layout is generated by IC Compiler. The layout and netlist have passed the verifications like static timing analysis and others. Lastly, the GDSII file is streamed out from IC Compiler.

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	xv
LIST OF APPENDICES	xvi

CHAPTER

1	INTRODUCTION	1
	1.1 Background	1
	1.2 Problem Statements	4
	1.3 Aims and Objectives	5
2	LITERATURE REVIEW	6
	2.1 VLSI Design Flow	6
	2.2 Digital Design Using VHDL	9
	2.3 Logic Synthesis	10
	2.3.1 Logic Synthesis Tools with Design Automation	10
	2.3.2 Synopsys Design Compiler	12
	2.4 Physical Design	13
	2.4.1 Partitioning	15

2.4.2	Floor-planning	16
2.4.3	Placement	16
2.4.4	Routing	17
2.4.5	Physical Verification	17
2.4.6	Tools for Physical Design	19
3	METHODOLOGY	20
3.1	VLSI Design Methodology	20
3.1.1	Register Transfer Level (RTL) Design	23
3.1.2	Logic Synthesis	24
3.1.3	Static Timing Analysis	25
3.1.4	Floor-plan	25
3.1.5	Power Network Synthesis	26
3.1.6	Clock Tree Synthesis	26
3.1.7	Placement and Routing	27
3.1.8	Chip Finishing	27
3.1.9	Physical Verification	28
3.2	Design Compiler	29
3.2.1	Data Setup	31
3.2.2	Technology Mapping and Optimization	31
3.2.3	Design Checking	31
3.2.4	Post-synthesis Output Data	31
3.3	IC Compiler	32
3.3.1	Data Setup	34
3.3.2	Floor-planning	35
3.3.3	Placement	36
3.3.4	Clock Tree Synthesis	37
3.3.5	Routing	37
3.3.6	Chip Finishing	37
4	RESULTS AND DISCUSSIONS	39
4.1	Logic Synthesis	39
4.1.1	Design Compilation and Translation	39

4.1.2	Checking Design	42
4.1.3	Static Timing Analysis before Mapping and Optimization	43
4.1.4	Area of the Design before Mapping and Optimization	44
4.1.5	Power Analysis before Mapping and Optimization	45
4.1.6	Mapping and Optimization	47
4.1.7	Analysis after Mapping and Optimization Process	48
4.1.8	Outputting Gate-Level Netlist	53
4.2	Physical Design	54
4.2.1	Library Setup	54
4.2.2	Floor-planning	55
4.2.3	Placement and Routing	64
4.2.4	Design Rule Checking Fixing and Final Verifications	68
5	CONCLUSION AND RECOMMENDATIONS	75
5.1	Conclusion	75
5.2	Recommendations	76
	REFERENCES	78
	APPENDICES	81

LIST OF TABLES

TABLE	TITLE	PAGE
	Table 3.1: Examples of Commands Used for Design Compiler	30
	Table 3.2: Script Files that are Included into Design Compile	30
	Table 3.3: Examples of Command that Used for IC Compiler	33
	Table 3.4: Script Files that are Included into IC Compiler	34
	Table 4.1: Modes or Options of the Mapping and Optimization Process	48
	Table 5.1: Summary of the Design Improvement	76

LIST OF FIGURES

FIGURE	TITLE	PAGE
Figure 1.1:	Design Flow for the project (Kang et al., 2015)	3
Figure 2.1:	Top-down Very-large-scale Integration (VLSI) Design Flow (Das, 2010)	8
Figure 2.2:	Logic Synthesis Flow by using Electronic Design Automation Tools (Chirag, 2012)	11
Figure 2.3:	Graphical User Interface of Synopsys Design Compiler (Yun & Ha, 2009)	12
Figure 2.4:	Synopsys Design Compiler dc_shell user interface (Steven & Zhao, 2010)	13
Figure 2.5:	Physical Design Flow (Das, 2010)	14
Figure 2.6:	Graphical User Interface of IC Compiler (Derek, 2016)	19
Figure 3.1:	Very-large-scale Integration (VLSI) Front-end Design Flow	21
Figure 3.2:	Very-large-scale Integration (VLSI) Back-end Design Flow	22
Figure 3.3:	Hierarchical Level of the Core Unit (Oregano System, 2012)	24
Figure 4.1:	Result of Automated Library Setup	40
Figure 4.2:	Result of Loading Design	40
Figure 4.3:	Results of Compilation	40
Figure 4.4:	Result of Linking Process	41
Figure 4.5:	Result of the Libraries Checking	41

Figure 4.6: Design Issue	42
Figure 4.7: Static Timing Analysis Report before Mapping and Optimization	44
Figure 4.8: Area of the Design before Mapping and Optimization	45
Figure 4.9: Power Report before Mapping and Optimization	46
Figure 4.10: Status of the Mapping and Optimization Process	48
Figure 4.11: Timing Report with Timing Issues	51
Figure 4.12: Timing Report without Timing Issues	52
Figure 4.13: Area Report Generated by the Tool	52
Figure 4.14: Power Report of the Design	52
Figure 4.15: Report of the Constraint Checking	53
Figure 4.16: Result of the Checking Design	53
Figure 4.17: Status of the Outputting Netlist	53
Figure 4.18: Status of Design Library Setup	54
Figure 4.19: Setting Tluplus Files	55
Figure 4.20: Status of the Loading Design	55
Figure 4.21: Initial Top Cell View of the Design	58
Figure 4.22: Creating Floor-plan	59
Figure 4.23: Created Floor-plan	59
Figure 4.24: Virtual Placement	60
Figure 4.25: Virtual Placement Report	60
Figure 4.26: Chip Summary	60
Figure 4.27: Congestion Report	61
Figure 4.28: Congestion Map with GRCs	61
Figure 4.29: Virtual Power Pad on the Top Hierarchy of the Design	62

Figure 4.30: PNS setting	62
Figure 4.31: PNA Power Map	63
Figure 4.32: PNS Report for VDD Nets	63
Figure 4.33: PNS Report for VSS Nets	63
Figure 4.34: Checking Design Report	65
Figure 4.35: Options for Placement	65
Figure 4.36: Status of Placement	65
Figure 4.37: Layout after Placement and Power Nets Routing	66
Figure 4.38: CTS Status	66
Figure 4.39: Clock Tree Summary	66
Figure 4.40: Result of Clock Tree Routing (Global Routing)	66
Figure 4.41: Summary of the Initial Routing	67
Figure 4.42: Summary of Detail Routing and Optimization	67
Figure 4.43: Summary of ECO Routing	67
Figure 4.44: Congestion Heat Map	70
Figure 4.45: Congestion Heat Map with Ratios	71
Figure 4.46: Power Map of the Design	71
Figure 4.47: Result of the ECO Routing	72
Figure 4.48: Result of LVS Verification	72
Figure 4.49: Final Layout	72
Figure 4.50: Centre of the Layout	73
Figure 4.51: Final Timing Report	73
Figure 4.52: Final Area Report	74
Figure 4.53: Final Power Report	74
Figure 4.54: Status of Stream Out Process	74

LIST OF SYMBOLS / ABBREVIATIONS

CAD	Computer-Aided Design
CTS	Clock Tree Synthesis
ddc	Optimized Netlist File Format (include constraints)
def	Scan Chain Information File Format
DRC	Design Rule Checks
EDA	Electronic Design Automation
GDSII	Graphic Database System II
GUI	Graphical User Interface
HDL	Hardware Description Language
IC	Integrated Circuit
Intel Corp.	Intel Corporation
IR	Voltage (Current * Resistance)
I/O	Input and Output
LVS	Layout Versus Schematic
MHz	Mega Hz (Frequency)
MOSFET	Metal-oxide-semiconductor Field-effect Transistor
PNS	Power Network Synthesis
P/G	Power and Ground
RTL	Register Transfer Level
sdc	Design Constraint Output File Format
SoC	System-on-Chip
STA	Static Timing Analysis
Synopsys, Inc.	Synopsys Incorporate
TAT	Turn-around Time
TLU+	RC Model File Format
VHDL	VHSIC Hardware Description Language
VLSI	Very-large-scale Integration

LIST OF APPENDICES

APPENDIX	TITLE	PAGE
APPENDIX A:	Project Timeline	81
APPENDIX B:	Synopsys Start-up File for Design Compiler	82
APPENDIX C:	Synopsys Start-up File for IC Compiler	84

CHAPTER 1

INTRODUCTION

1.1 Background

Based on the report from the Semiconductor Industry Association (SIA) in 2007, the global sales of semiconductor reached at the level in term of billions dollar with the amount of 255.6 billion US dollar (Jiang *et al.*, 2008). From the report, we know that semiconductor is profitable and it provided a very high profit margin for semiconductor based companies. In real life, computers, smartphones, home appliances, personal digital assistants devices and entertainment devices, rely on integrated circuit or IC chip to operate. (Jiang *et al.*, 2008). So, semiconductor industry plays a significant role in our daily lives.

In this technology era, every field consist of relationships and formulae that are useful and beneficial for technology development (Ethan, 2006). The founder of Intel Corporation, Gordon Moore, predicted the density of the transistors incorporated in an IC will approximately double every two years. This prediction is known as Moore's Law and it is still valid until today (Intel Corp, no date). The foundry has been continuously inventing advanced IC process technology in approximately every 24 months by significant scaling down the Metal Oxide Field Effect transistors (MOSFETs) in order to improve chip performance and also to reduce chip area as stated by Moore's Law (Samar, 2013). Due to the continuous scaling down of MOSFETs, the dimensions of MOSFETs are gradually approaching their physical restriction. Consequently, the development of advanced semiconductor

technology according to Moore's Law is being slowed down recently (Samar, 2013). However, Moore's Law is still an influential concept in current semiconductor industry development (Ethan, 2006).

Process of developing an IC by packing millions of transistors into a single chip, with smaller die size, is known as Very-Large Scale Integration or VLSI (Kang *et al.*, 2015). As VLSI technology becomes advanced, a complex system can be embedded into a single chip, known as System-on-Chip (SoC) (Chang & Kim 2006). At the meantime, multi-core SoC can be implemented by recent technology.

In order to design a SoC, extremely long design turnaround time (TAT) is required (Ryota *et al.* 1985). The full-custom design approach requires even more TAT (Ryota *et al.* 1985). Because of this, Engineering Design Automation (EDA) tools are invented to overcome the problems. Shorter TAT in designing a complex SoC by using EDA tools (Ryota *et al.* 1985). EDA tools provide automatic design approaches for IC design which can minimize the required TAT (Ryota *et al.* 1985). For instance, the design approaches are auto place and route, gate-level optimization, STA, CTS and etc. Therefore, EDA tools are important in IC design industry.

The VLSI design flow can be used as the guide for the project. The design flow of the project is shown in Figure 1.1. RTL stands for Register Transfer Level. In this stage, the behaviour and the function of the design are described in Hardware Description Language (HDL) format. For logic synthesis, it is a process of converting the RTL source codes into optimized gate level netlist (Kang *et al.*, 2015). For physical design, it is the process of converting the optimized gate-level netlist into geometrical representation of the design which is known as layout (Kang *et al.*, 2015). GDSII file, a binary file, consists of cell references and the geometry parameters of the cells (Steven, 1994). All labelled texts, geometric shapes of the cell and other information about the layout are represented by Graphical Database System (GDSII) file (Steven, 1994).

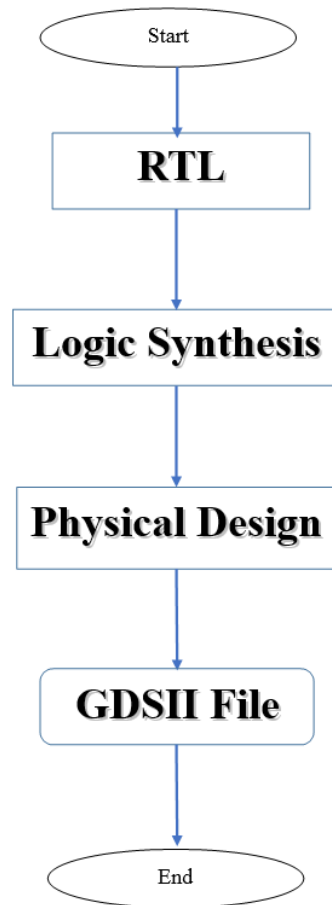


Figure 1.1: Design Flow for the project (Kang et al., 2015)

1.2 Problem Statements

The problem statements of the project are stated below:

- For designing an IC from RTL to GDSII, longer TAT is required.
- Low performance of 8051 microcontroller is low due to the low clock frequency.
- Design cost for an IC is higher.
- Timing closure of the design becomes worse when increasing the clock frequency.

Before the EDA tools are invented, all designs are done from scratch. The designers need to draw logic gates of the design on the papers and then translate the logic gate into layout by drawing the layout on the papers. This is a time consuming process for IC design. Therefore, longer TAT is required for designing an IC without using EDA tools.

The older version of 8051 microcontroller has low clock frequency and hence causes low performance. Typically, the clock frequency of the older microcontroller is about 12 MHz (Intel Corp., n.d.). Since the clock frequency is lower, the performance of the chip is quite restricted. When the chip area is larger, the fabrication cost is also higher and hence the design cost of the IC will be increased,

After increasing the clock frequency, the timing closure of the design become worse and eventually the timing violations like setup violations and hold violations will appear.

1.3 Aims and Objectives

The objectives of the project are shown as follows:

- i) To complete the VLSI design flow from RTL to GDSII.
- ii) To reduce the design TAT by using EDA tools with design automation.
- iii) To reduce the design cost by chip-area optimization.
- iv) To improve the performance by increasing the clock frequency.
- v) To improve the timing closure after the clock frequency is increased.

In this project, I will go through the full IC design flow. I will go through logic synthesis, physical design, physical verification and others. EDA tools provide the design automations for IC design process which can reduce the design TAT. In order to reduce the design cost, the chip-area is reduced as small as possible. As mentioned earlier, the clock frequency of the original 8051 microcontroller is around 12 MHz. In this project, I will try to increase the frequency as high as possible in order to improve the performance of the chip. Some of the blocks like RAM and ROM are taken out from the design. The RAM and ROM blocks are not be implemented into layout. After increasing the frequency, timing violations will appear. Those violations will be fixed in order to improve the timing closure after the clock frequency has increased.

CHAPTER 2

LITERATURE REVIEW

2.1 VLSI Design Flow

VLSI design flow is a series of steps that is used to translate and synthesize the specifications of a system into a chip (Das, 2010). The design flow is shown in Figure 2.1. This VLSI design flow is known as top-down VLSI design flow (Das, 2010). To begin with, the flow starts with the system specifications. The specifications of the design such as speed, power and area are specified. The next phase is functional design. In this phase, the design is specified in the behavioural model and then translated into RTL (Das, 2010). After the system specification is done, the next phase is functional verification. In this phase, the design is checked in terms of functionality (Das, 2010). Further verification for RTL is done in order to check the syntax and functionality of the RTL (Das, 2010). If the functionality of the design does not meet the requirements, the designer will perform the functional design again in order to achieve the design requirements (Das, 2010). After the functional verification phase, the subsequent phase is logic design. In this phase, the designers design the logic networks of the system by following the functional descriptions of the design (Das, 2010). At the meantime, the I/O characteristic is used at the functional design level (Das, 2010). Basically the RTL is converted into logic-level representation in this phase. The logic-level networks are then stored as logic-level netlist (Das, 2010). Furthermore, the logic verification is used to verify the correctness of the netlist (Das, 2010). The following phases are circuit design and circuit verification. The logic-level netlist is translated into electronic network by

using transistors (Das, 2010). Further verification is done for the circuit-level representation. Once the circuit design and verification are completed, the next step is physical design. The circuit-level netlist is translated into physical layout (Das, 2010). Further verification is done for physical layout such as DRC, LVS check and others (Das, 2010). The final step is fabrication and testing. The physical layout is converted into a finished silicon chip (Das, 2010). Further verifications are then done to test the functionality and characteristic of the chip (Das, 2010). These sequences of design flow illustrate the top-down design approach for the IC design (Das, 2010).

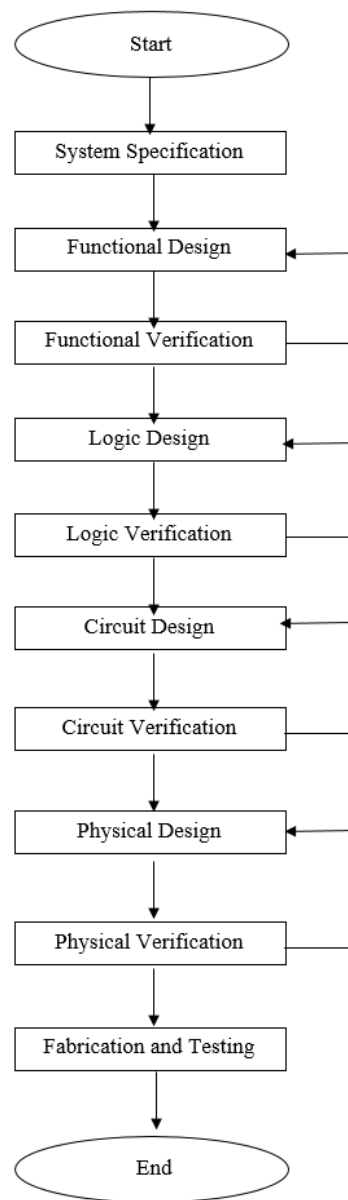


Figure 2.1: Top-down Very-large-scale Integration (VLSI) Design Flow (Das, 2010)

2.2 Digital Design Using VHDL

Usually, the functional descriptions and behavioural descriptions of the system are expressed in HDL. There are two popular HDL languages, namely VHDL and Verilog HDL (Das, 2010).

VHDL languages can be used to model a digital system at RTL (Das, 2010). In this level, the functional descriptions and behavioural descriptions of the system are coded into VHDL format. VHDL has 5 basic constructs, namely sequential statements, concurrent statement, netlist statements, timing specifications and waveform generation (Das, 2010).

The hardware abstraction of VHDL has two views, namely external view and internal view (Das, 2010). External view is represented by **Entity** (Das, 2010). Meantime, the internal view is represented by **Architecture**. **Entity** is used for only port specification design (Das, 2010). It models all input and output ports of the design (Das, 2010). On the other hands, **Architecture** representing the actual internal specifications of the hardware like logic functions and interconnects (Das, 2010). The relationship between input and output ports are represented by Architecture (Das, 2010).

Generally, four different architecture styles can be used for VHDL modelling (Das, 2010). Namely, structural modelling, behaviour modelling, dataflow modelling and mixed modelling (Das, 2010). In behavioural modelling, the behavioural representation in the form of truth table can be used for behavioural modelling style of the system (Das, 2010). Besides, the specifications of the system are described in the form of Boolean equations is known as dataflow modelling (Das, 2010). In structural modelling, the interconnections of the system which consists of basic building blocks and logic gates are described (Das, 2010).

2.3 Logic Synthesis

Logic synthesis is a process of translating the high-level description of the design in RTL into an optimized gate-level netlist (Das, 2010). Logic synthesis is a set of techniques that is used for VLSI design industry (Bernard *et al.*, 1989). Recently, CAD approach is popular for electronic system design (Das, 2010). Of course, logic synthesis can be done by using CAD approach.

2.3.1 Logic Synthesis Tools with Design Automation

Recently, several logic synthesis tools are being developed based on the CAD approach. Logic synthesis tools are the EDA software that provides design automations for IC designers (Das, 2010). Usually, the logic synthesis expressions are included into the algorithms of the tools (Bernard *et al.*, 1989). These expressions refer to automate design procedure that is used to generate gate-level Boolean networks for physical design (Bernard *et al.*, 1989).

Generally, logic synthesis tools provide design automations which are able to reduce design TAT significantly (Das, 2010). For instance, the design automation is automatic translating the RTL description of the design into gate-level netlist (Das, 2010). Besides, the tools provide automatic chip area, power and power optimization (Das, 2010).

Typically, a VLSI chip consists of hundreds or even thousands of logic gates. Consequently, manual synthesis at all levels becomes impractical due to huge number of logic gates (Das, 2010). Nevertheless, using EDA tools for logic synthesis is more convenient and practical.

Figure 2.2 shows the logic synthesis flow by using EDA tools. To begin with, the RTL description of the design like RTL codes in VHDL format is translated into un-optimized internal representation. Basically, the un-optimized internal

representation is a type of data structure that is used to store the information of the design (Das, 2010). Next, the logic optimization is performed by the tools based on the Karnaugh map optimization approach (Das, 2010). Before technology mapping and optimization, the design constraints and technology library are applied to the tools. Design constraints are known as design goals such as operating condition, chip area, clock frequency, clock uncertainty, load and others (Das, 2010). Technology library is also known as standard cell library. Standard cell library is a type of database library which consists of logic gate and macro cells like flip-flop, adder and multiplexer (Das, 2010). After that, technology mapping and optimization are performed by the tools. Generally, technology mapping is a process of connecting the selected elements from the standard cell library in such a way as to achieve the predefined synthesis goals and functionality of the design (Eugenio & Pablo, 1995). On the other hand, optimization is a process of applying the logic transformations to the design based on the design constraints (Eugenio & Pablo, 1995). Eventually, the optimized gate-level netlist is generated by the tools.

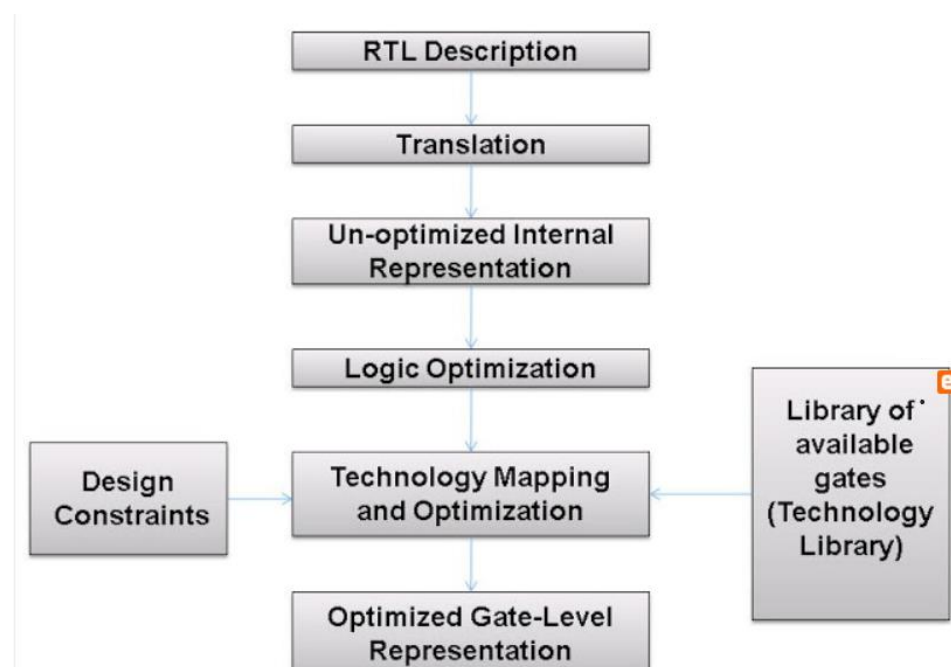


Figure 2.2: Logic Synthesis Flow by using Electronic Design Automation Tools
(Chirag, 2012)

2.3.2 Synopsys Design Compiler

The popular synthesis tool is Design Compiler from Synopsys Inc. In the Synopsys synthesis product family, Design Compiler is the core synthesis engine. It consists of two user interface, known as dc_shell; a command line interface and GUI (Thomas & William, 2010). Figure 2.3 shows the GUI of the Design Compiler. Figure 2.4 shows the dc_shell user interface. Synopsys Design Compiler can be used to constrain a design for area and timing, apply synthesis techniques to achieve area and timing closure, analyse the results and generate design data that works with physical design or layout tools (Synopsys Inc, 2010).

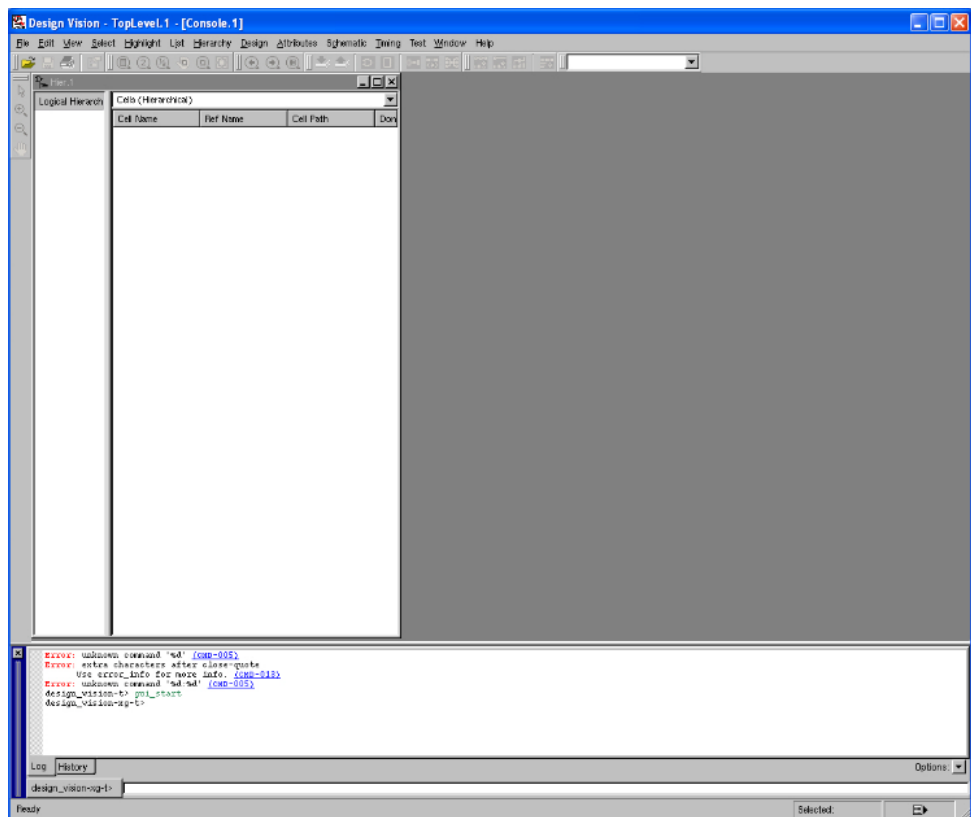
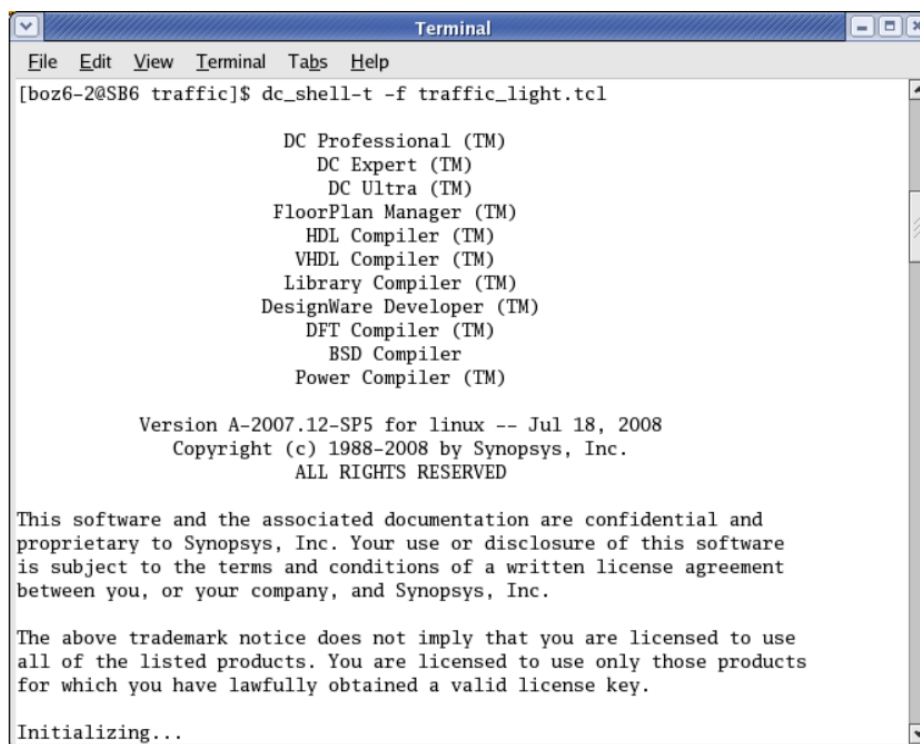


Figure 2.3: Graphical User Interface of Synopsys Design Compiler (Yun & Ha, 2009)



```
Terminal
File Edit View Terminal Tabs Help
[boz6-2@SB6 traffic]$ dc_shell-t -f traffic_light.tcl

      DC Professional (TM)
      DC Expert (TM)
      DC Ultra (TM)
      FloorPlan Manager (TM)
      HDL Compiler (TM)
      VHDL Compiler (TM)
      Library Compiler (TM)
      DesignWare Developer (TM)
      DFT Compiler (TM)
      BSD Compiler
      Power Compiler (TM)

      Version A-2007.12-SP5 for linux -- Jul 18, 2008
      Copyright (c) 1988-2008 by Synopsys, Inc.
      ALL RIGHTS RESERVED

This software and the associated documentation are confidential and
proprietary to Synopsys, Inc. Your use or disclosure of this software
is subject to the terms and conditions of a written license agreement
between you, or your company, and Synopsys, Inc.

The above trademark notice does not imply that you are licensed to use
all of the listed products. You are licensed to use only those products
for which you have lawfully obtained a valid license key.

Initializing...
```

Figure 2.4: Synopsys Design Compiler dc_shell user interface (Steven & Zhao, 2010)

2.4 Physical Design

Physical design is a process of converting gate-level netlist into geometric representation which is known as layout (Das, 2010). The layout consists of standard cells, routings, clock trees, power nets and other information (Das, 2010). Figure 2.5 shows the physical design flow.

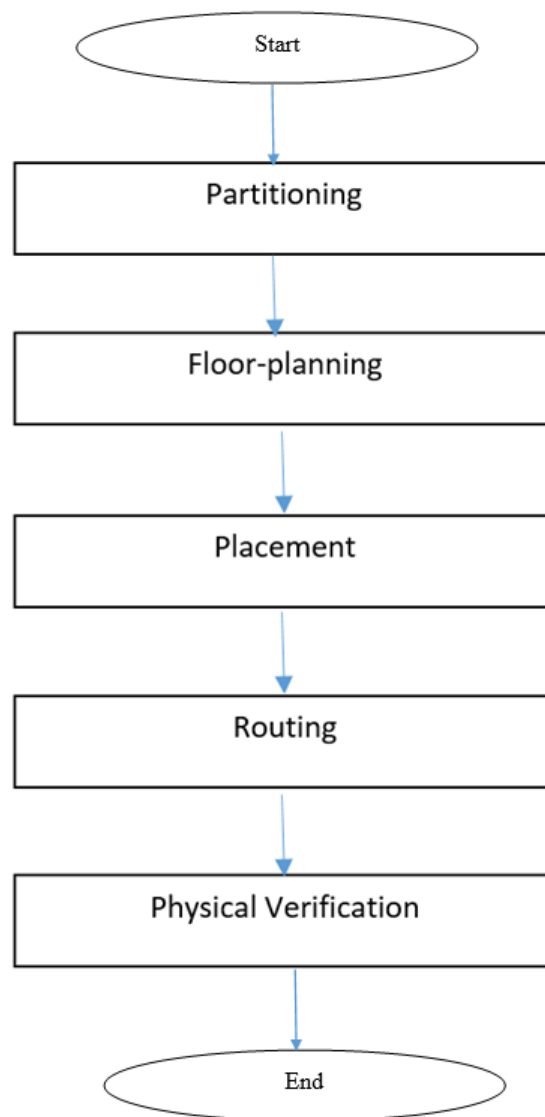


Figure 2.5: Physical Design Flow (Das, 2010)

2.4.1 Partitioning

A VLSI chip consists of thousands of logic gates. Due to the complexity of the design, the layout of the entire design cannot be handled at the same time. Normally, the complex design is partitioned into smaller subsystem in order to minimize the complexity of the design during physical design (Chen, 2009). The implementation of entire design cannot be performed at the same time due to the limitation on memory space and computation power (Jeffery *et al.*, 2009). Thus, partitioning plays an important role in physical design stage.

Generally, decomposing the design into a set of smaller subcircuits or blocks is known as partitioning (Jeffery *et al.*, 2009). After decomposition, each subcircuit or block can be designed simultaneously and independently. An interface specification that is used for connecting all the subcircuits or blocks is generated (Jeffery *et al.*, 2009). The interconnects that is used to connect the blocks or subsystems are required. The collection of those interconnects is known as netlist (Jeffery *et al.*, 2009).

There are a few factors that need to be considered for partitioning. Namely, size of the blocks or subcircuits, the number of blocks or subcircuits and the number of interconnects between the subcircuits or blocks (Jeffery *et al.*, 2009).

The partitioning process for standard cell designs is non-hierarchical (Jeffery *et al.*, 2009). Generally, the partitioning algorithms is used to partition the circuit into a set of unconnected sub-circuits such that each sub-circuit corresponds to a cell in a standard cell design library (Jeffery *et al.*, 2009). The non-hierarchical form is used for standard cell designs in term of partitioning procedure (Jeffery *et al.*, 2009).

2.4.2 Floor-planning

In the floor-planning stage, the chip is planned in such a way to accommodate all design components like standard cells, macros, memory units and its interconnects within a minimum area (Das, 2010). The area of the chip that is used for accommodate those design components can be calculated (Jeffery *et al.*, 2009). The interconnects of the chip and the location of the standard cells need to be determined in order to minimized of the chip area (Jeffery *et al.*, 2009).

2.4.3 Placement

After the floor-planning stage, the area accommodated by each component of the chip and the number of terminals are defined (Jeffery *et al.*, 2009). The process of arranging the components of the chip on the specified area is known as placement (Jeffery *et al.*, 2009). The steps for placement include determining the position of each components on the chip, determining the minimum area arrangement for the components on the chip which can allows the completion of routing without any violations and lastly place those components on the chip according to defined positions (Jeffery *et al.*, 2009). After the placement, all of the components of the design like standard cells, macros and IPs are placed on to the defined core area accordingly with minimum area (Jeffery *et al.*, 2009).

In a standard cell library, the height of each standard cell is defined equally (Jeffery *et al.*, 2009). This means that the standard cells have same height compared to others standard cells in the standard cell library (Jeffery *et al.*, 2009). In standard cell design style, all cells are placed in rows by minimizing the heights of all the channels and the width of the widest row (Jeffery *et al.*, 2009). The total area of the chip is the summation of required placement area and required routing area (Jeffery *et al.*, 2009). The process of routing area estimation which determine the channel height, play an important role in determining the overall chip area (Jeffery *et al.*,

2009). The empty spaces over the standard cell rows are then used for routing (Jeffery *et al.*, 2009).

2.4.4 Routing

Routing is a process of determining the geometrical representation which is known as layout of all the nets of the chip (Jeffery *et al.*, 2009). This process is used for interconnecting the components on the chip based on the netlist (Jeffery *et al.*, 2009). The area which is not accommodated by the components is partitioned into rectangular regions known as channel and switchbox (Jeffery *et al.*, 2009). The channel and switchbox are utilized by the routers in order to complete the routing between the all components on the chip (Jeffery *et al.*, 2009). Further optimization can be done based on the constraints (Jeffery *et al.*, 2009).

Standard cell rows are determined with uniform lengths channels (Jeffery *et al.*, 2009). In order to minimize the height of the channel, all the nets in the channel must be routed (Jeffery *et al.*, 2009). Further optimization can be done based on the routing constraints (Jeffery *et al.*, 2009). For instance, the constraints are restricting length of the critical nets within some acceptable limits and minimizing the length of the longer nets (Jeffery *et al.*, 2009).

2.4.5 Physical Verification

Physical verification is a process of verifying the IC design for manufacturability and electrical connectivity rules (Rajesh *et al.*, 2014). The three major steps in physical verification are DRC, LVS checks and parasitic extraction (Chang *et al.*, 1999).

The physical verification can be done by automation (Rajesh *et al.*, 2014). The scripts required by the EDA tools to perform physical verification are generated by the algorithms when developing the layout and runset files of DRC and LVS

checks (Rajesh *et al.*, 2014). These files are then executed in the terminal command line in order to automate the physical verification process (Rajesh *et al.*, 2014). The result also are generated by automation (Rajesh *et al.*, 2014).

Design Rule Checker (DRC) is a process of analysing the design by a given set of design rules (Das, 2010). The DRC is performed in order to ensure that the design is compromised with a set of manufacturing process rules (Chang *et al.*, 1999). For instance, the rule can be the minimum space between adjacent lines (Chang *et al.*, 1999).

Basically, technology library which consists of a set of design rules is provided by the foundry (Rajesh *et al.*, 2014). The DRC can be done by the design automation that provided by EDA tools (Rajesh *et al.*, 2014).

On the other hand, Layout versus Schematic (LVS) check is a process of comparing the netlist derived from the layout with the original netlist (Chang *et al.*, 1999). Basically, the original netlist comes from design synthesis phase (Chang *et al.*, 1999). The netlist derived from layout is basically generated after the routing process is done (Chang *et al.*, 1999). Generally, the LVS check is done by the tools automatically.

Besides, parasitic extraction is a process of obtaining the parasitic elements of the design from the geometrical information and material properties of the design (Das, 2010). For instance, the parasitic elements are resistance, capacitance and inductance (Das, 2010). This process basically can be done by the design automation that provided by the tools. The tools will automatic obtaining the parasitic elements from the layout and generating the report for parasitic verification.

2.4.6 Tools for Physical Design

The EDA tools provide the design automations such as auto placement, auto routing, physical verification and others. The TAT for physical design can be reduced by using EDA tools.

Synopsys IC Compiler can be used for hierarchical chip-level design planning known as floor-planning, placement, CTS, routing and physical verification (Synopsys Inc, 2007.). IC compiler takes an optimized gate-level netlist with design constraints file as the input and produces layout as an output (Derek, 2016). Figure 2.6 shows the graphical user interface (GUI) of the IC compiler.

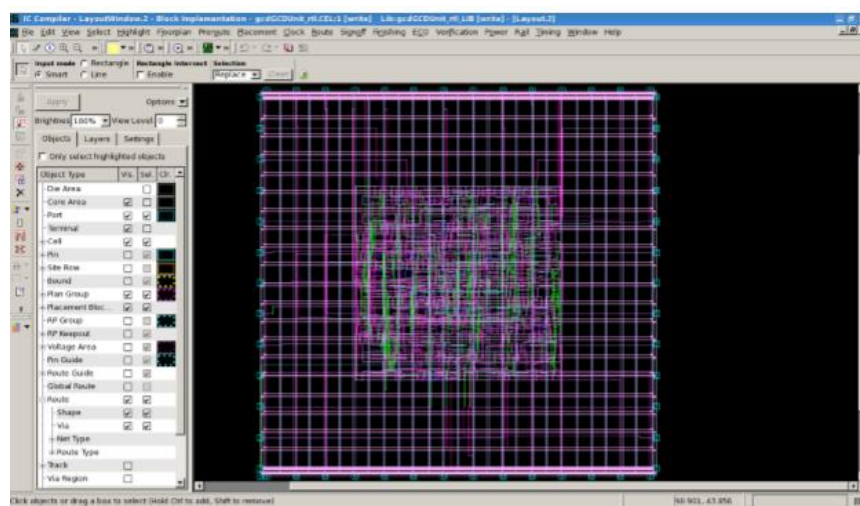


Figure 2.6: Graphical User Interface of IC Compiler (Derek, 2016)

CHAPTER 3

METHODOLOGY

3.1 VLSI Design Methodology

VLSI design process is expensive especially in the design phase and fabrication phases. Consequently, the absolute discipline in the design phase is critical and significant. A proven design methodology is required in order to ensure the design is succeed at the first attempt. Therefore, VLSI design methodology is important and critical in IC design industry. Figure 3.1 shows the design flow for front-end design. Figure 3.2 shows the design flow for back-end design.

The front-end design process consists of system specifications / behavioural design, RTL design, design verification, logic synthesis and STA. In this project, the RTL source codes are retained from the Oregano System. The RTL codes are verified by the Oregano System. The system specifications / behavioural design, RTL design and design verification are done by the Oregano system. Therefore, the system specifications / behavioural design, RTL design and design verification methodology will not be taken for this project.

The back-end design is the design methodology for layout generation. Basically, the back-end design is known as physical design. In fact, physical design process consists of Floor-plan, PNS, CTS, placement & routing, STA, chip finishing, post-layout verification, physical verification and finally tape out. In this project, Synopsys IC Compiler is used for physical design. The methodology for physical design process will be discussed in following subsections.

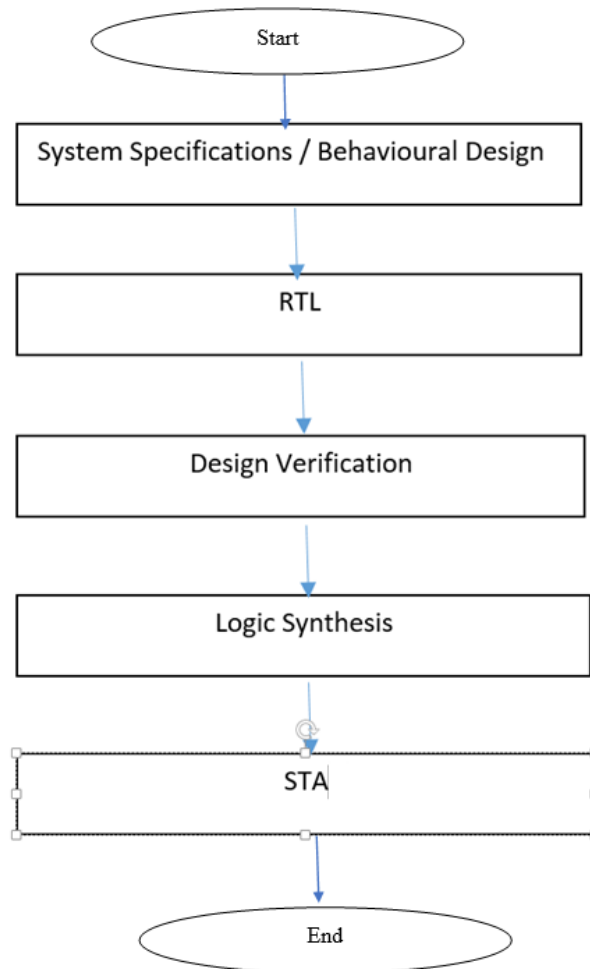


Figure 3.1: Very-large-scale Integration (VLSI) Front-end Design Flow

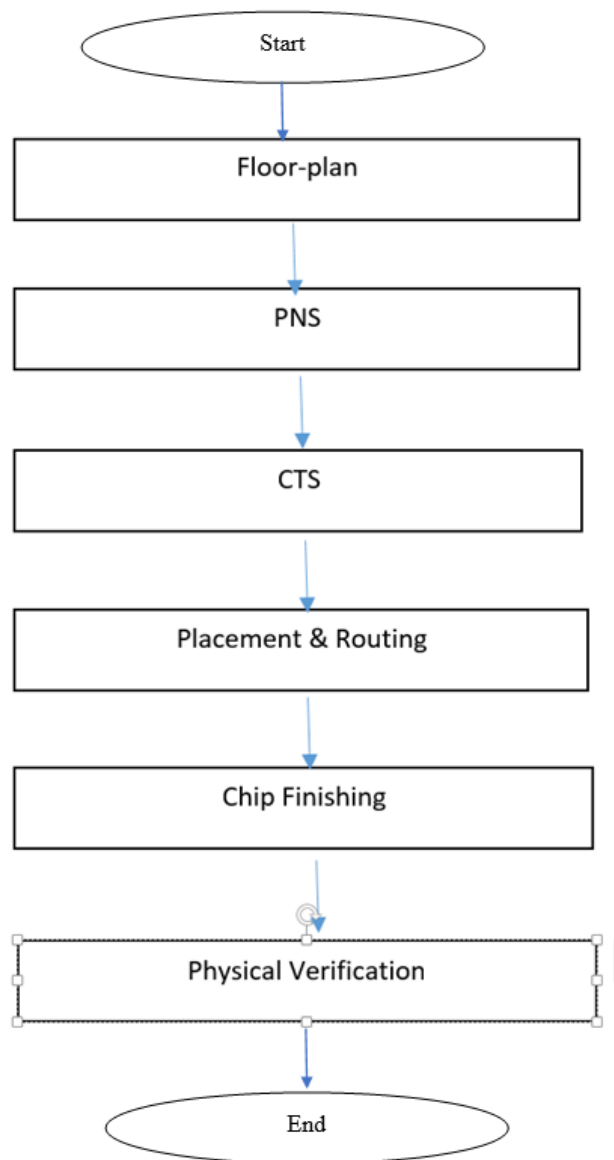


Figure 3.2: Very-large-scale Integration (VLSI) Back-end Design Flow

3.1.1 Register Transfer Level (RTL) Design

The RTL codes are provided by the Oregano System. The syntax of the codes and functionality of the design are verified by the Oregano System.

The design is coded using the VHDL format. The designers coded the design separately. In this project, only the core of the design will be implemented. Since the RTL codes for the core are designed separately, thus the RTL codes need to be combined by according to design hierarchical level. Figure 3.2 shows the hierarchical level of the core unit. The codes are combined by following the hierarchical level of the design that is shown in Figure 3.2. Note that only mc8051_core is implemented. Therefore, top hierarchy is mc8051_core.

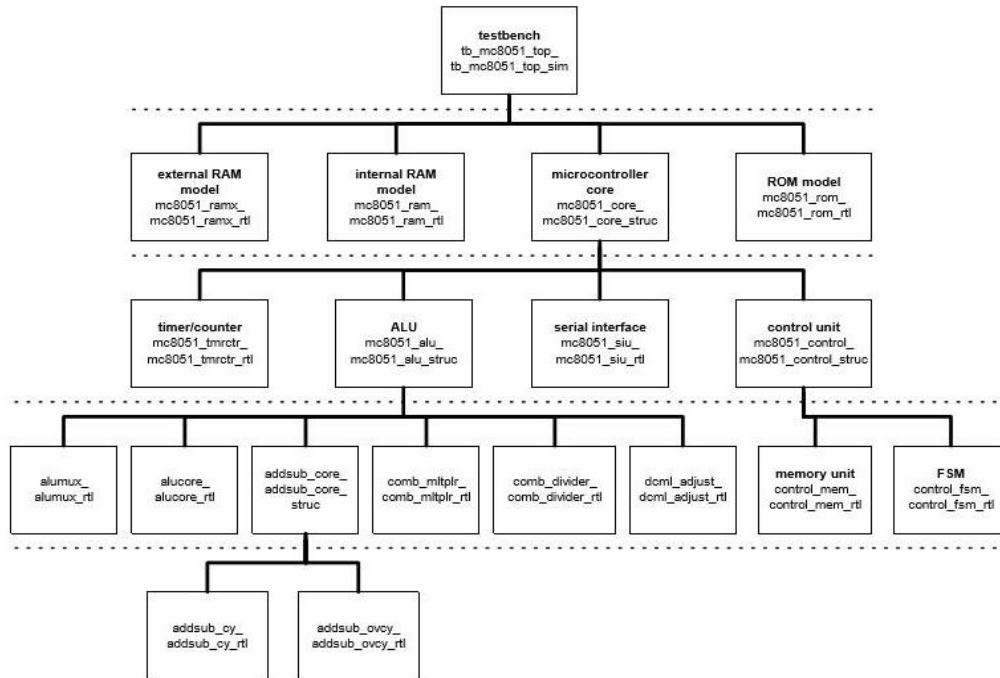


Figure 3.3: Hierarchical Level of the Core Unit (Oregano System, 2012)

3.1.2 Logic Synthesis

The RTL codes are imported to the EDA tool which is Design Compiler from Synopsys Inc. The RTL codes are then synthesized into optimized gate-level netlist based on the design constraints. The design constraints can be timing constraints and area constraints. The technology process of the logic gates, flip-flops and wire model is specified in the target cell library by the foundry. So, the target cell library must be provided for the tool in order to perform technology mapping and optimization. The gate-level netlist can be optimized for speed, performance, area and power through the design constraints.

3.1.3 Static Timing Analysis

STA is performed on fully synchronous designs to validate the timing performance of the design. The speed and timing constraints must be provided to the tool as an input for STA. For instance, the design constraints can be clock period, input delay, output delay, load capacitance, wire capacitance, wire load model, clock uncertainty and clock transition.

A wire load model is used for synthesis and STA in order to achieve early timing closure. Besides, the operating condition that provided by the foundry need to be specified in design constraints. STA checks all possible timing violation paths under worst case operating condition. The STA that is provided by the tool is quite efficient and accurate. Once the design is free from timing violations, next steps which is physical design will be carried on. The next section will discuss about the back-end design methodology for this project.

3.1.4 Floor-plan

In floor-planning stage, core utilization ratio is set to the appropriate value in order to give margins for routability. The utilization ratio is the ratio between height of the core area and the length of the core area. The extra space requirements for power networks and clock tree routing need to be taken into consideration during the setup of core utilization ratio. Usually, few iterations are required for the process.

The core utilization ratio setup is important for weighting the trade-off between the routability of the chip and chip area. Setting the core utilization ratio too high resulting the increasing of chip area and hence causes the design cost to increase. Besides, the timing closure for critical paths becomes worse due to long routing can be happened if the ratio is set too high.

3.1.5 Power Network Synthesis

There must be sufficient power and ground pins in order to drive up the chip to its operating voltage level. The sufficient power and ground nets can minimize the power the ground noise from interfering. Placement constraints script can be used to perform the placement of these power and ground pads strategically at each side of the chip. Ground bounce simulation is done to determine the sufficient pairs of power and ground pads for the chip.

If there are more than one power domain, the power distribution among these power domains is significant in floor-planning and placement phases. The tool synthesizes the power distribution networks of the chip based on the power budget specifications for each domain. The IR drop analysis can be done by using the tool. The purpose for IR analysis is to prevent the supply voltage drops below the designed range. In order to avoid the supply voltage drops below the designed range, the width of the power and ground buses must be designed by sufficient value. Adjustments can be applied to floor-plan and power networks by several iterations until the IR drop constraints are met.

3.1.6 Clock Tree Synthesis

The clock tree is synthesized by design synthesis close to an ideal clock for a fully synchronous design. During CTS, the tools will buffer the clock tree networks to a sufficient level of transition like fast rise and fall edges. Meantime, the tool tries to balance the clock branches so that clock skew is under controlled in order to prevent any timing violations. Clock skew control is not so important in clock tree buffers generation.

3.1.7 Placement and Routing

The standard cells of the design can be placed into core area by using the tool. The placement can be done either using manual placement or auto placement. For manual placement, the designer places the standard cells into core area by using drag and drop function on the GUI of the tool. In this case, the designer needs to decide the locations of the standard cell to place. Manual is insufficient for large design. It is impractical to place hundreds or thousands of cells into the core area manually. Generally, auto place is more sufficient and practical for large design even for small design. In order to use auto placement, the designer need to include the placement constraints and routing constraints into the tool in order to perform auto placement. The tool will place those standard cells into the core area sufficient based on the constraints. The tool will leave some space for routing based on the constraints. Consequently, the chip area can be optimized by using auto placement. Further optimization for placement can be done to improve routability.

It is impossible to route thousands to millions of nets on the chip. Thus, auto routing is provided by the tool in order to automate the routing process. The tool will find out the best possible ways to route the design by least number of DRC, timing and LVS violations. Those violations can be fixed by re-place the standards cells and re-route the signal and power nets. The re-place and re-route process might be taken in several iterations in order the clean those violations. The following subsection will discuss about the chip finishing.

3.1.8 Chip Finishing

Chip finishing is the additional steps required for the chip tape out preparation. The antenna check can be done by the tool based on the antenna rules that provided by foundry. The antenna violations are caused by the longer length of the interconnects with the same metal layers. This will cause damage to the chip due to the collected charges on the particular metal layers during fabrication. Therefore, the antenna

check must be performed before tape out. The antenna fixing can be done by the tool automatically. The tool breaks the long metal lines by inserting a next higher-level metal layer. Basically, this is known as layer-jumper approach. For top level metal, the protection diodes are added by the tool to fix the violations.

The fill cells are added into the remaining unused area in standard cell rows. The purpose of this process is to meet the continuity of the N-well and P-well and power buses. Additional metals and polys fill layers are added to the un-routed areas in order to establish the density requirements that set by the foundry. IO fill cells are added to the remaining empty space corresponding to the IO cells in order to complete the IO power ring of the chip which can provides ESD protection. Lastly, redundant vias are added wherever there are spaces to prevent the possibility of manufacturing issues. Some vias might become too narrow and might not be functioning properly.

3.1.9 Physical Verification

DRC can be performed by the tool to ensure the layout is met the design rule requirements that set by foundry. The DCR violations can be fixed by re-routing the violated nets. Basically, the fixing process can be done manually or automatically. For auto fix, the fixing script needs to be provided to the tool.

LVS is the process of checking the equivalence between the physical layout against the optimized gate-level netlist to ensure the logical descriptions of the design are not changed during the physical implementation process like placement, routing and others.

In the other hand, parasitic extraction is a process of calculating the capacitance, inductance and resistance for parasitic elements in the design. The tool will calculate those parameters by referring to the parasitic library that provided by the foundry.

Lastly, the tool generates the GDSII file that is used for fabrication.

3.2 Design Compiler

Synopsys Design Compiler consists of two user interface, known as `dc_shell` command line interface and GUI. GUI is used for design purpose with graphical visualization. The `dc_shell` command line is used to invoke the command for the compiler to perform specified tasks. Table 3.1 shows the some examples of the command that are used for Design Compiler. Table 3.2 shows the script files that are included into Design Compiler.

There are four main steps to use Synopsys Design Compiler to perform logic synthesis. The four main steps are known as data setup, technology mapping and optimization, design checking and post-synthesis output data. The following subsection will discuss the data setup methodology.

Table 3.1: Examples of Commands Used for Design Compiler

Commands	Description
dc_shell -gui	Invoke Design Compiler
write -f ddc -hier -out mc8051_core.ddc	Save the netlist as ddc format (used for physical design)
report_design	Report operating condition and wire load
report_timing	Generate timing report for STA
check_timing	Summarize all timing paths
compile -scan -boundary -map high	Technology mapping and optimization with high effort
report_constraint -all_violators	Summarize violated constraints
source mc8051_core.con	Applied design constraint script to the compiler
read_ddc mc8051_core.ddc	Read ddc file into compiler
compile_ultra	Invoke powerful optimization algorithms
write_sdc mc8051_core.sdc	Save the constraints as sdc format for third party tool
write_verilog mc8051_core.v	Save the netlist in Verilog format
group_path	Group the timing path

Table 3.2: Script Files that are Included into Design Compile

Scripts	Description
.synopsys_dc.setup	Used for data setup like library setup , search path setup, import design
mc8051_core.con	Consists of design constrains in tcl command based

3.2.1 Data Setup

In this stage, all related library like target library, link library and symbol library need to be included into Design Compiler. The design also needs to be imported into Design Compiler. These steps can be done automatically by using a script file. Once the compiler is loaded, the script file is executed automatically by the compiler. All related libraries and search paths are set. The design is imported into compiler.

3.2.2 Technology Mapping and Optimization

Before performing mapping and optimization, the design constraints file must be applied to the Design Compiler. The design constraints can be coded into a script file by using tcl command format. In order to instruct the compiler to perform mapping and optimization, the command is inserted into dc_shell command line interface. The compiler performs the technology and optimization automatically based on the libraries and design constraints respectively.

3.2.3 Design Checking

At this stage, design checking include STA and constraint check. The design checking process can be done by tool automatically. A series of commands is used to generate timing report, constraint report, timing path report and I/O report.

3.2.4 Post-synthesis Output Data

The optimized gate-level netlist is saved into several file format like ddc, and verilog format. The constraints are saved into sdc format. The scan chain information is

saved into def format. These all output data are then used for physical design in IC Compiler or third party tools.

3.3 IC Compiler

Synopsys IC Compiler consists of two user interface, known as `icc_shell` command line interface and GUI. GUI is used for design purpose with graphical visualization. The `icc_shell` command line is used to invoke the command for the compiler to perform specified tasks. Table 3.3 shows the examples of the command that are used for IC Compiler. Table 3.4 shows the script files that are included into IC Compiler.

There are six main steps to use Synopsys IC Compiler to perform logic synthesis. The six main steps are known as data setup, floor-planning, placement, CTS, routing and chip finishing. The following subsection will discuss the data setup methodology.

Table 3.3: Examples of Command that Used for IC Compiler

Commands	Description
initialize_floorplan	Initialize floor-plan
derive_pg_connection	Make logical P/G pin connection
create_pad_rings	Create P/G pad rings
set_dont_touch_placement	Fix the placement (does not change in optimization)
analyse_fp_rail	Perform IR drop computation
check_physical_design	Check the readiness of placement
check_physical_constraints	Report the validity of the constraints
place_opt	Perform timing-driven and congestion-driven placement and logic optimization
create_placement_blockage	Create placement blockage
psyopt	Performs Incremental logic optimization
set_clock_tree_options	Set global CTS options
check_clock_tree	Verification on clock tree
clock_opt	Performs synthesis & balancing of individual clock tree networks, timing & DRC optimization of non-clock logic and routing of clock tree network
report_clock_tree	Analysing CTS result
set_fix_hold	Enable hold time fixing
set_delay_calculation – arnoldi	Perform net delay calculation by using Arnoldi model
set_route_zrt_detail_options –antenna true	Enable concurrent antenna fixing during signal detailed route
route_opt	Initialize routing
report_critical_area	Analyse critical area
insert_zrt_redundant_vias	Insert redundant vias
insert_metal_filler	Insert metal fillers
write_parasitics	Parasitic extraction
write_stream –cell mc8051_core.gdsii	Output netlist in GDSII format
save_mw_cel	Save the design cell

Table 3.4: Script Files that are Included into IC Compiler

Script	Description
.synopsys_dc.setup	Used for data setup, create design library and import design
opt_ctrl.tcl	Timing and optimization controls
zic_timing.tcl	Check zero-interconnect timing constraints
derive_pg.tcl	Create logical P/G connections
ndr.tcl	Define non-default clock routing rules
cts_setup.tcl	Set CTS options
antenna .tcl	Set up antenna rule
routing_setting.tcl	Some settings for routing

3.3.1 Data Setup

In data setup, the related libraries need to be included into IC Compiler like logic/timing library, physical library directories, technology file and RC model files. A design CEL need to be created in order to save the design during physical design. In order to perform physical design, the optimized gate-level netlist and design constraints files are imported into the compiler. The all steps can be done by using a series of commands. The all commands can be grouped into a script file known as .synopsys_dc.setup. Once the tool is started up, the tool will executed the commands that stored in the script file line by line. Those all settings can be done automatically by using the script file.

3.3.2 Floor-planning

Floor-planning stage consists of four main steps. These five main steps must be taken in order to complete the design floor-plan. These five main steps are create starting floor-plan, virtual flat placement, analyse/optimize congestion and synthesis power network.

Firstly, the physical-only pad cells are created by the tool. The locations of the cell must be specified. Next, the initial floor-plan are initialized. The core and periphery area are created. The core parameters are defined by specific values. Then, the pad filler cells are inserted and power and ground pad rings are created. These all steps can be done by using `icc_shell` command line, scripts and GUI. The subsequent paragraph will discuss about the virtual flat placement.

There are two steps in virtual flat placement process. The two steps are setting placement strategy parameters and performing virtual placement. These steps can be done by using command lines. The subsequent paragraph will discuss about congestion analysis and optimization.

The congestion is analysed layer by layer. Then, standard cell placement constraints are applied to the compiler by using `tcl` command. Additional placement constraints can be added like placement blockage constraints. The “fp placement strategy” is modified by using command line. The congestion-driven placement is performed. The high effort congestion strategy is invoked by using script file if the congestion is not acceptable. If the congestion is still not acceptable, the floor-plan must be modified by editing top-level pads or ports, modifying core aspect ratio and size and modifying power grid structure. The subsequent paragraph will discuss about power network synthesis (PNS).

The logical power and ground connections must be defined. Power network constraints are applied before PNS. The constraints are coded into a script file. Then, the tool will perform PNS. If the maximum IR drop is not acceptable, the re-constraint and re-synthesize steps need to be taken. After the IR drop is acceptable,

the new floor-plan is created with added P/G pads. The P/G pins are connected by the tool. The power rails along the standard cell placement rows are created. IR drop map is used to analyse the power network. After that, power net placement blockages are applied and then performs placement legalization. Timing checking is required to check the design in term of timing performance. These all steps can be done by using commands or GUI.

3.3.3 Placement

The placement can be done by referring to IC Compiler placement flow. The flow consists of placement setup & checks, placement & optimization and congestion & timing improvement.

In the placement setup and checks stage, the TLU+ files and timing & optimization controls script are re-applied to the compiler. The placement readiness is checked by using commands.

In the placement and optimization stage, the auto placement is activated by using commands. The tool will perform the placement and optimization based on the constraints.

Timing and congestion are analysed after the placement & optimization. If the timing violations or congestion are occurred, path grouping approach will be used for efficient optimization. After the paths are grouped by the clocks controlling their endpoints, the placement and optimization will be go through again. Basically, these can be done by using tcl commands.

3.3.4 Clock Tree Synthesis

CTS goals, timing constraints and timing control scripts are applied to the compiler before CTS is performed. Clock specifications are set and then optimizes the design for power. Basically, these can be done by using command line. The CTS is started by applying command to the tool. The certain analysis can be done by using CTS GUI. The CTS GUI consists of CTS browser, CTS schematic and clock arrival histogram. The timing report is generated in the command line terminal. Finally, the clock nets are routed by the tool.

3.3.5 Routing

The routing is performed by using command. The operations include global routing, track assignment, detail routing and concurrent optimization. Concurrent optimizations include timing, power and area optimizations. The concurrent antenna fixing during signal detailed routing is enable. The routing blockages are defined by using script. The redundant via will be inserted into the layout y using default via definitions from the technology file.

3.3.6 Chip Finishing

The critical area is reported by using the tool. The minimum jog length is controlled by using command. In order to fix the antenna violations by diode insertion approach, the diode insertion need to be turned on and then run incremental detail route. These can be done by using command. The filler cells in unused placement area are inserted by using script. There are two methods can be used for metal fill insertion that provided by the tool. First method is insert_metal_filler by IC Compiler and second method is signoff_metal_fill by Hercules based IC Compiler.

In final validation stage, the parasitic calculated can be done by using the compiler. The command is required. Lastly, writing the GDSII output can be done by using command. The tool will automatic write the output in the format of GDSII.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Logic Synthesis

In the logic synthesis stage, the RTL codes in VHDL format are translated into logic representation format known as the gate-level netlist. The logic optimization and STA are done in order to ensure that the timing, area, power and other constraints are met.

4.1.1 Design Compilation and Translation

Before using the Design Compiler, the `.synopsys_dc.setup` file (script file) is created in order to automate the library setup, import design, compilation and design translation processes. The contents of the script file are shown in appendices. Once the Design Compiler is started up, the compiler executes the instructions of the script file. Figure 4.1 shows the result of the library setup by using the script file. In this case, the link library is `saed90nm_max.db`. The target library is `saed90nm_max.db`. The symbol library is `saed90nm.sdb`. These libraries consist of the details of standard cells, macros, interconnects, technology processes, parasitic, timing and others.

```

Settings:
search_path: . /synopsys/sym2013.03-SPI/libraries/sym /synopsys/sym2013.03-SPI/minpower/sym /synopsys/sym2013.03-SPI/dw/syn_ver /synopsys/sym2013.03-SPI/dw/sim_ver unmapped unconstrained mapped rtl lib cons
link_library: * sse490m_max.db
target_library: sse490m_max.db
symbol_library: sse490m.sdb
Loading db file '/home/student/Desktop/FYP_updated/synthesis/lib/sse490m_max.db'
Loading db file '/synopsys/sym2013.03-SPI/libraries/sym/gtech.db'
Loading db file '/synopsys/sym2013.03-SPI/libraries/sym/standard.sldb'
Loading link library 'sse490m_max'
Loading link library 'gtech'
Loading vhdl file '/home/student/Desktop/FYP_updated/synthesis/rtl/mc8051_core.vhd'

```

Figure 4.1: Result of Automated Library Setup

After libraries setup, the next part is loading design. By applying the script file, the design is loaded into the compiler automatically. The design is in RTL code format (VHDL). The design is translated and compiled into unmapped and not optimized netlist. Figure 4.2 and Figure 4.3 showed the results of the loading design and translation and compilation respectively. From the results, the loading of the design into Design Compiler is successful. Besides, the compilations of the design such as mc8051_alu, mc8051_control, mc8051_siu, mc8051_tmrctr, addsub_core, alumux, alucore, comb_multiplier, comb_divider, decimal_adjustment, control_mem and control_fsm are successful. After translation and compilation, the current design is mc8051_core.

After compilation, the *link* command is used to link the design to the specified libraries in 3order to tell the compiler to use the specified libraries for the entire design. The standard cells and other information from the libraries are used for the design during design synthesis. Figure 4.4 showed the result from the compiler for the linking process.

```

Loading vhdl file '/home/student/Desktop/FYP_updated/synthesis/rtl/mc8051_core.vhd'
Running PRESTO HDLC

```

Figure 4.2: Result of Loading Design

```

-
Presto compilation completed successfully.
dc_shell> Current design is 'mc8051_core'.
Current design is 'mc8051_core'.

```

Figure 4.3: Results of Compilation


```

dc_shell> link

Linking design 'mc8051_core'
Using the following designs and libraries:
-----
* (24 designs)          /home/student/Desktop/FYP_updated/synthesis/rtl/mc8051_core.db, etc
saed90nm_max (library)  /home/student/Desktop/FYP_updated/synthesis/lib/saed90nm_max.db

```

Figure 4.4: Result of Linking Process

Before proceeding to the next stage, libraries checking is an important task. In order to ensure that the used libraries are correct, the *list_libs* command is used. In this case, the used libraries are correct. In order to prevent the redesign process, libraries checking process is important. Therefore, libraries checking process is necessary before gate-level optimization and mapping process. Redesigning process is required if the used libraries are wrong. Besides, the libraries checking process also helps the design to ensure that the libraries are applied to the compiler. If no libraries are applied, the compiler will use the built-in libraries. The built-in libraries are not suitable for the design because the libraries do not have any physical design data and eventually cause the physical design cannot be carried out. The designer needs to repeat the design synthesis by using suitable libraries for the design. This is a very time-consuming process. So, it is important to check the libraries before logic mapping and optimization. The result of the libraries checking is shown in Figure 4.5. In this case, the applied libraries are saed90nm_max.db and saed90nm.sdb. The libraries gtech.s5db, gtech.db and standard.sldb are the built-in libraries.

```

dc_shell> list_libs
Logical Libraries:
-----
Library      File          Path
-----
saed90nm_max saed90nm_max.db /home/student/Desktop/FYP_updated/synthesis/lib
gtech        gtech.db      /synopsys/syn2013.03-SP1/libraries/syn
standard.sldb standard.sldb  /synopsys/syn2013.03-SP1/libraries/syn

```

Figure 4.5: Result of the Libraries Checking

4.1.2 Checking Design

Before the mapping and optimization process, the design is checked. The result is shown in Figure 4.6. From the result, there were 26 problems on I/O ports and 56 problems on standard cells. The unconnected port is known as floating port. This is to say that the I/O ports are opened. No signal nets are connected to those particular ports. Some modules like ROM and RAM are removed from the design during RTL stage. The ports that are used to connect the ROM and RAM with the main modules are opened since the ROM and RAM are removed. On the other hand, the amount of feedthrough is 1. Feedthrough is the shorted net within a module or block. The input port of that particular module or block is connected directly to the output port of that particular module or block. It is different compared to the shorted output. Shorted output is the output port of the module or block connected directly to the input port of another module or block by a net. By referring to the RTL source code, the shorted connections can be found. Thus, the problem that causes the shorted connections is the name for both ports are different. The last issue is the cells do not drive any nets. The output ports of the cells are not connected by any nets. This issue also known as floating ports. All these problems can be solved automatically by the tool during mapping and optimization process. The tool will remove the unconnected ports during the mapping and optimization process. Besides, the tool will add buffers to the shorted nets and feedthrough nets during mapping and optimization process. Lastly, the tool will remove the cells that do not drive any nets. The nets that connect the output ports of the modules or blocks to the cells also will be removed by the tool.

Name	Total
Inputs/Outputs	26
Unconnected ports (LINT-28)	13
Feedthrough (LINT-29)	1
Shorted outputs (LINT-31)	12
Cells	56
Cells do not drive (LINT-1)	56

Figure 4.6: Design Issue

4.1.3 Static Timing Analysis before Mapping and Optimization

The STA result of the design is calculated by the tool. The tool consists of built-in static timing analyser. Static timing analyser is used to guide optimization decisions during mapping and optimization process and generate timing and timing related reports. The tool breaks the design into individual timing paths and the delay of each path is calculated. Then, all path delays are checked against timing constraints to determine if the constraints have been met. The tool analyses each timing path at least twice for single-cycle max-delay timing. The STA result is shown in Figure 4.7. The tool calculates the data arrival time of the design. Data arrival time is the time that the signal arrive at the input pin of the sequential elements like flip-flops. In this case, the data arrival time is about 20198.58 ns. In order to avoid timing violations, the data required time must be greater than 20198.58 ns. Data required time is the sum of the clock period with clock network delay minus the sum of the clock uncertainty with library setup time. For example, the data required time is 20200.00 ns (assume the propagated clock signal is ideal; no clock network delay, library setup time and uncertainty). The frequency of the clock is $1 / 20200 \text{ ns} = 49.5 \text{ kHz}$. The design only can works on 49.5 kHz or below. The smaller clock frequency causes the low performance of the design. The clock frequency will be improved later on. In this case, the slack is not calculated before the timing constraints are not applied.

```

Operating Conditions: WORST   Library: saed90nm_max
Wire Load Model Mode: enclosed

Startpoint: reset (input port)
Endpoint: ram_en_o (output port)
Path Group: (none)
Path Type: max

Des/Clust/Port      Wire Load Model      Library
-----
mc8051_core        ForQA                saed90nm_max
control_mem        ForQA                saed90nm_max

Point                                                    Incr      Path
-----
input external delay                                0.00      0.00 f
reset (in)                                          0.00      0.00 f
i_mc8051_control/reset (mc8051_control)            0.00      0.00 f
i_mc8051_control/i_control_mem/reset (control_mem) 0.00      0.00 f
i_mc8051_control/i_control_mem/I_333/Z (GTECH_NOT) 20198.55  20198.55 r
i_mc8051_control/i_control_mem/B_244/Z (GTECH_BUF) 0.00      20198.55 r
i_mc8051_control/i_control_mem/ram_en_o (control_mem) 0.00      20198.55 r
i_mc8051_control/ram_en_o (mc8051_control)          0.00      20198.55 r
ram_en_o (out)                                     0.02      20198.58 r
data arrival time                                  20198.58
-----
(Path is unconstrained)

```

Figure 4.7: Static Timing Analysis Report before Mapping and Optimization

4.1.4 Area of the Design before Mapping and Optimization

Before the mapping and optimization process, the design area is checked. The tool provides the automation in calculating the area of the design. The report that generated by the tool is shown in Figure 4.8. Based on the report, the total area of the design is $104984.540474 \mu\text{m}^2$. The design consists of 156 ports and 367 nets. The total area of the 367 nets is $5369.718050 \mu\text{m}^2$. The design consists of 89 cells (non-combinational cells) and 85 combinational cells. The total area of the cells is the sum of combinational cell area with the non-combinational cell which is $99614.822424 \mu\text{m}^2$ in this case. The macros or black boxes do not exist in the design. Moreover, the design has 85 inverters/buffers. Thus, the total design area is the sum of the cells area with the nets area.

The design is quite large. The area of the design can be reduced during mapping and optimization process with specific design constraints.

```

Library(s) Used:

    saed90nm_max (File: /home/student/Desktop/FYP_updated/synthesis/lib/saed90nm_max.db)

Number of ports:                156
Number of nets:                 367
Number of cells:                89
Number of combinational cells:  85
Number of sequential cells:     0
Number of macros/black boxes:   0
Number of buf/inv:              85
Number of references:           8

Combinational area:             77138.842108
Buf/Inv area:                   30517.862722
Noncombinational area:          22475.980316
Macro/Black Box area:           0.000000
Net Interconnect area:         5369.718050

Total cell area:                99614.822424
Total area:                     104984.540474

```

Figure 4.8: Area of the Design before Mapping and Optimization

4.1.5 Power Analysis before Mapping and Optimization

Power analysis is important for the design. Without power analysis, we do not know how much of the switching power, leakage power and internal power of the design. Every chip must be driven by a power source in order to make it functional. Thus, power analysis is important for the design.

The tool provides the automation for power analysis. The tool calculates all those power parameters like internal power, leakage power and switching power of the design. Eventually, the tool generates a report of the power analysis. The report is shown in Figure 4.9. The cell internal power dissipation is about 154.2252 μw . This means the power required to drive the cells of the design is 154.2252 μw . Since the technology of the cells is 90 nm, the power dissipation is smaller than those larger technology like 0.18 μm (180 nm). Besides, the net switching power is about 60.2696 μw . Since the design is a fully digital circuit, all of the cells are able to perform switching at each clock cycle (switch between 1 and 0). The power dissipation of a net during switching is called net switching power. The net switching power dissipation is directly proportional to the performance and total capacitance of a net. If the clock frequency is higher, the switching rate is also higher. As the switching rate increases, the net switching power dissipation increases also. On the other hand, the dynamic power of the design is the sum of the cell internal power

with the net switching power. In this case, the total dynamic power is 214.4947 μw . The dynamic power is the amount of power that required to drive the entire design and also known as total power dissipation of the design. On the other hand, the cell leakage power of the design is about 323.0420 μw . The cell leakage power is inversely proportional to the technology process. The cell leakage power is larger when the smaller technology is used like 90 nm in this case compared to larger technology process like 180 nm. In this case, 323.0423 μw is wasted due to the cell leakage power. The amount of the cell leakage power depends on the channel length (technology process) and number of cells of the design. Thus, the total power required to power up the chip is the sum of the total dynamic power with the cell leakage power which is 537.5377 μw in this case. The power can be optimized during mapping and optimization process with specific design constraints.

```

Cell Internal Power = 154.2251 uW (72%)
Net Switching Power = 60.2696 uW (28%)
-----
Total Dynamic Power = 214.4947 uW (100%)

Cell Leakage Power = 323.0420 uW

```

Power Group	Internal Power	Switching Power	Leakage Power	Total Power	(%)	Attrs	Cell Count
io_pad	0.0000	0.0000	0.0000	0.0000	(0.00%)		0
memory	0.0000	0.0000	0.0000	0.0000	(0.00%)		0
black_box	0.0000	0.0000	0.0000	0.0000	(0.00%)		0
clock_network	0.0000	0.0000	0.0000	0.0000	(0.00%)		0
register	7.0817	2.9006	6.6575e+07	76.5572	(14.24%)		578
sequential	0.0000	0.0000	0.0000	0.0000	(0.00%)		0
combinational	147.1427	57.3691	2.5647e+08	460.9805	(85.76%)		8952
Total	154.2244 uW	60.2697 uW	3.2304e+08 pW	537.5377 uW			

Figure 4.9: Power Report before Mapping and Optimization

4.1.6 Mapping and Optimization

The mapping and Optimization process can be done by using the tool based on the design constraints. The tool consists of the algorithms that can perform well in the mapping and optimization. Basically, the tool provides automation for the mapping and optimization process but the design constraints are set by designers.

The tool performs the mapping process to produce the gate-level implementation of the design that meets design constraints by using logic gates based on the target technology libraries. During the optimization process, the tool reduces the number of the cells, area, power and number of nets without changing the functionality of the design. The tool also fixes the timing issues like setup violations and hold violations.

The status of the mapping and optimization is shown in Figure 4.10. In this case, the process is executed successfully. The total run time for the process is nine minutes and eight seconds. This process cannot be done by humans without the automation that provided by the tool. For very small design, it is possible but time consuming. For large design, it is impossible to done by humans. Thus, the design automation is important in IC design industry. It helps to save design TAT, invested money and also provides better results. The tool consists of several mapping and optimization algorithms that can be controlled by designer.

In order to start the mapping and optimization process, the designer need to instruct the tool to start the process by using command. The tool consists of several mapping and optimization algorithms that can be controlled by designer. There are several built-in options or modes for the process. The designer can instruct the tool to execute the process with preferred modes or options. The modes or options of the process that used to map and optimize the design is shown in Table 4.1. After that, the tool executes the process and generates the optimized gate-level netlist of the design.

```

-----
0:09:06 104973.2 66.58 29852.1 0.0 i_mc8051_control/i_control_mem/ie_reg[3]/D
0:09:07 104984.5 66.57 29851.5 0.0 i_mc8051_control/i_control_mem/ie_reg[3]/D
0:09:08 104984.5 66.57 29851.4 0.0
Loading db file '/home/student/Desktop/FYP_updated/synthesis/lib/saed90nm_max.db'

Optimization Complete
-----
1

```

Figure 4.10: Status of the Mapping and Optimization Process

Table 4.1: Modes or Options of the Mapping and Optimization Process

Command/Option/Mode	Description
Compile	Instruct the tool to start mapping and optimization process
-map_effort high	High mapping effort (perform additional mapping by increasing the number of iterations)
-area_effort high	Optimize the area by minimize the number of gates in the design with high effort (perform additional optimizations by increasing the number of iterations)
-power_effort high	Optimize the power by high effort (perform additional optimizations by increasing the number of iterations)
-boundary_optimization	Perform design boundary optimization

4.1.7 Analysis after Mapping and Optimization Process

After the mapping and optimization process, several analysis and verifications such as timing analysis, constraints analysis, power analysis and area analysis need to be done before proceed to next stage. If the design does not meet the requirements, the re-constraint process is required and then mapping and optimization process are executed again. Basically, these steps require to go through several iterations in order to meet the requirements.

In fact, the tool provides automation for the STA. The timing report is generated by the tool after STA is done by the tool. The report is shown in Figure 4.11. In this case, the clock frequency is around 150 MHz (1/6.67ns). The data

required time is clock period + clock network delay – clock uncertainty – library setup time. The data required time is 10.01 ns. Data arrival time is the time that the signal arrive at the input pin of the sequential elements like flip-flops. In this case, the data arrival time is 29.20 ns. The slack is the difference between data required time and the data arrival time. If the slack is positive, the design is not violated by the timing-related issues else the design is violated by timing-related issues. If the data arrival time is larger than data required time, the design is suffered by setup violations. Setup violation is defined as the late arrival of the signal that causes the signal misses the time when it should advance. In this case, the design is violated by setup violations. The tool provides automation to solve the hold violations. Unfortunately, the tool is able to optimize the design in term of timing to prevent setup violations but not able to solve setup violations. So, the designer need to re-constraint the timing constraints to solve those setup violations. For hold violations, the designer need to instruct the tool to fix the hold violations during mapping and optimization process by using the command. The command is set in design constraint script file. After re-constraint the design constraints, the mapping and optimization process need to be executed again. The tool mapped and optimized the design based on the new design constraints. Eventually, the timing issues are successfully fixed after few iterations of the process. The timing report is shown in the Figure 4.12. In this case, the slack is positive. The slack is about 0.01. So, there are no timing-related issues in the design. Therefore, the timing closure of the design is met.

In chip area analysis, the area of the chip is calculated by the tool. Compared to previous result, the chip area is reduced by 32233.7945 μm^2 which is 30.7% of the original area. The area of the design is 30.7% smaller than original design area. Based on the result, the net interconnect area is reduced by 114.190895 μm^2 which is 2.13% smaller than original net interconnect area since the number of nets is reduced by 31. The unconnected, floating, redundant nets are removed by the tool without affecting the interconnection of the design. Besides, the numbers of combinational cells and cells are reduced by 30. So, the total cell area is reduced by 32119.6036 μm^2 which is 32.24% smaller compared to the original total cell area. Figure 4.13 showed the result of the area analysis. During the mapping and optimization process, the area optimization effort is set to be high. So, the tool performs area optimization by a few iterations without changing the functionality of the chip and meets the

design constraints. Therefore, the chip area is successfully reduced by 30.7%. As a conclusion, the area of the chip is reduced without changing the functionality and the design constraints are met.

On the other hand, the tool provides automation for the power analysis. Figure 4.14 shows the power report that generated by the tool. Based on the result, the total power dissipation of the design is 341.0928 μW . The total power dissipation is reduced by 196.4449 μW which is 36.55% smaller than original total power. The number of cells is reduced after the mapping and optimization process. Since the number of cells is reduced, the cell internal power dissipation is reduced by 44.48% which is 68.7541 μW of original cell internal power. Moreover, the cell leakage power is reduced by 119.3584 μW which is 36.95% smaller than original cell leakage power. Lastly, the net switching power dissipation is also reduced because the number of nets is reduced by the tool. As mentioned earlier, the net switching power dissipation is directly proportional to switching rate and total capacitance of the switching nets. Although the switching rate is higher, the net switching power dissipation is not increased significantly. However, the net switching power dissipation has reduced significant since the number of switching nets is reduced significantly after mapping and optimization process. As the number of switching nets decreases, the total amount of the capacitance of the nets decreases also. Thus, the net switching net power dissipation decreases also. The reduction of the net switching power dissipation is 8.3311 μW . The percentage of the reduction is 13.82%. As the conclusion, the power constraint is improved after mapping and optimization process. The result of the design constraints verification will be discussed in subsequent paragraph.

The tool provides the automation to check the unachieved design constraints or also known as violated design constraints. But, the designer need to instruct the tool to perform design constraints checking by using command. The result of the verification is shown in Figure 4.15. In this case, only the violated constraints are shown in the report. From the report, only one constraint is violated. The constraint is area constraint. In the design constraints script file, the maximum area is set to 0. Means the maximum area is 0. Although this is impossible, the tool still accept this constraint during mapping and optimization process. The purpose of setting the

maximum area to 0 is to force the tool to minimize the area as small as possible. The tool performs the area optimization with maximum effort even though the tool is not able to optimize the area until 0. So, the area of the design is the optimum value. Therefore, this violated constraint can be ignored. No action needs to be taken to fix this constraint violation.

Before producing netlist, the design is checked again. The tool provides automation to check the design whether the design still consists of unconnected ports, feedthrough, shorted nets and other issues. The report is generated by tool and the report is shown in Figure 4.16. All issues are fixed by the tool. Previously, the design consists of unconnected ports, feedthrough, shorted nets and the cells that do not drive any nets. These issues can be fixed during mapping and optimization process but the design need to instruct the tool to solve these issues during the process by using commands. The tool is instructed to adding buffers to the feedthrough net in order to solve the feedthrough issue. The unconnected ports are removed by using the command *remove_unconnected_ports*. Besides, the shorted nets can be fixed by adding buffers to those shorted nets. The shorted nets are causes by the different names of the I/O ports of the different modules in the design. Since the names of those I/O ports cannot be changed, so adding buffers to those shorted nets is more practical to solve those shorted nets issues. Lastly, the cells do not drive any nets issues can be fixed by removing those cells that do not drive any nets without changing the functionality of the design.

```

clock clk (rise edge)                6.67      6.67
clock network delay (ideal)          4.00      10.67
clock uncertainty                     -0.20     10.47
i_mc8051_tmrctr_0/s_count11_reg[6]/CLK (SDFFARX1) 0.00      10.47 r
library setup time                   -0.46     10.01
data required time                    10.01
-----
data required time                    10.01
data arrival time                     -29.20
-----
slack (VIOLATED)                     -19.19

```

Figure 4.11: Timing Report with Timing Issues

clock clk (rise edge)	6.67	6.67
clock network delay (ideal)	4.00	10.67
clock uncertainty	-0.20	10.47
i_mc8051_tmrctr_0/s_counth1_reg[2]/CLK (SDFFARX1)	0.00	10.47 r
library setup time	-0.46	10.01
data required time		10.01

data required time		10.01
data arrival time		-10.00

slack (MET)		0.01

Figure 4.12: Timing Report without Timing Issues

```

Library(s) Used:

  saed90nm_max (File: /home/student/Desktop/FYP_updated/synthesis/lib/saed90nm_max.db)

Number of ports:          156
Number of nets:          336
Number of cells:         59
Number of combinational cells: 55
Number of sequential cells: 0
Number of macros/black boxes: 0
Number of buf/inv:       55
Number of references:     7

Combinational area:      45186.048157
Buf/Inv area:            5944.320122
Noncombinational area:  22309.170654
Macro/Black Box area:   0.000000
Net Interconnect area:  5255.527155

Total cell area:        67495.218811
Total area:             72750.745966

```

Figure 4.13: Area Report Generated by the Tool

```

Cell Internal Power = 85.4710 uW (62%)
Net Switching Power = 51.9385 uW (38%)
-----
Total Dynamic Power = 137.4095 uW (100%)
Cell Leakage Power  = 203.6836 uW

```

Power Group	Internal Power	Switching Power	Leakage Power	Total 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	7.7860	3.6989	6.4000e+07	75.4851	(22.13%)	
sequential	0.0000	0.0000	0.0000	0.0000	(0.00%)	
combinational	77.6853	48.2395	1.3968e+08	265.6077	(77.87%)	

Total	85.4713 uW	51.9384 uW	2.0368e+08 pW	341.0928 uW		

Figure 4.14: Power Report of the Design

Design	Required Area	Actual Area	Slack
mc8051_core	0.00	72750.74	-72750.74 (VIOLATED)

Figure 4.15: Report of the Constraint Checking

```
dc_shell> check_design
1
```

Figure 4.16: Result of the Checking Design

4.1.8 Outputting Gate-Level Netlist

After the design passes all analysis and meets all the design constraints (except maximum area constraint), the design is save as a netlist file for the physical design. The netlist file is in Verilog format and ddc format. The design constraint file is save in sdc format. The status of the outputting gate-level netlist is shown in Figure 4.17. The outputting process is success.

```
dc_shell> set verilogout_no_tri true
true
dc_shell> change_names -rule verilog -hierarchy
1
dc_shell> write -f verilog -hierarchy -out mapped/mc8051_core.v
Writing verilog file '/home/student/Desktop/FYP_updated/synthesis/mapped/mc8051_core.v'.
1
dc_shell> write -f ddc -hierarchy -output mapped/mc8051_core.ddc
Writing ddc file 'mapped/mc8051_core.ddc'.
1
dc_shell> write_sdc mapped/mc8051_core.sdc
1
```

Figure 4.17: Status of the Outputting Netlist

4.2 Physical Design

After the optimized netlist is generated, the next stage is the physical design. In this stage, the optimized gate-level netlist is implemented into the layout.

4.2.1 Library Setup

Before we start to perform physical design, library setup is required. The target library for physical design is saed90nm_max.db. The physical libraries for the physical design are saed90nm_max and saed90nm_io_max. The technology file is saed90nm_icc_1p9m.tf. The technology libraries setting and libraries paths can be done by using Synopsys start-up file named as .synopsys_dc.setup. The contents of the file is shown in appendix.

After setting up the technology library, the next step is to create the design library (milkyway library in .mw format). The design library is created by using GUI of ICC. Figure 4.18 showed the status of the design library setup. The design library is successfully created. After that, the tluplus files are set by using the GUI. Figure 4.19 showed the status of the setting. Then, the design is loaded into the library and the physical design can be started. The status of the loading design is shown in Figure 4.20.

```
Technology file ../libs/tech/saed90nm_icc_1p9m.tf has been loaded successfully.  
{mc8051_core.mw}
```

Figure 4.18: Status of Design Library Setup

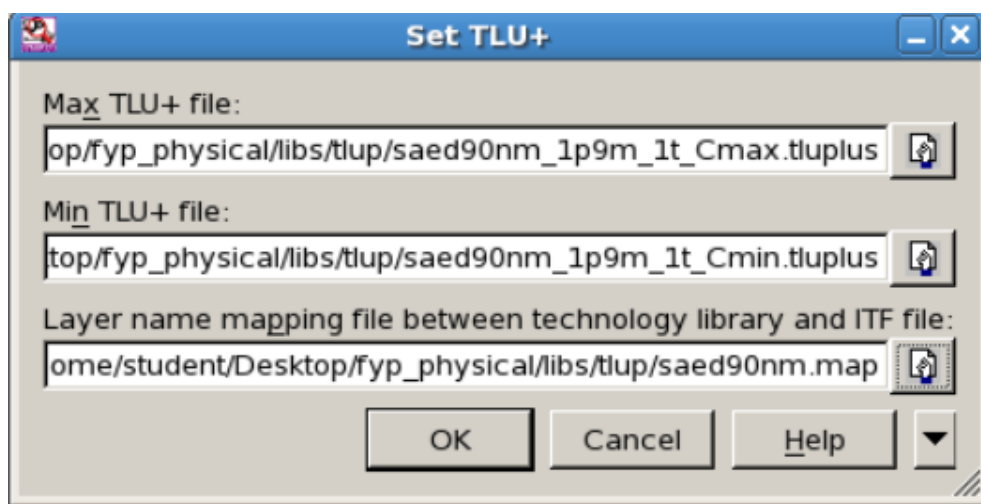


Figure 4.19: Setting Tluplus Files

```

Reading ddc file '/home/student/Desktop/fyp_physical/design_data/mc8051_core.ddc'.
Loaded 27 designs.
Current design is 'mc8051_core'.
Current design is 'mc8051_core'.

Linking design 'mc8051_core'
Using the following designs and libraries:
-----
* (27 designs)           /home/student/Desktop/fyp_physical/design_data/mc8051_core.ddc, etc
saed90nm_max (library)  /home/student/Desktop/fyp_physical/libs/db/saed90nm_max.db
saed90nm_io_max (library) /home/student/Desktop/fyp_physical/libs/db/saed90nm_io_max.db

```

Figure 4.20: Status of the Loading Design

4.2.2 Floor-planning

After libraries setup, the floor-plan of the design can be started. After the design is loaded into the IC Compiler (ICC), the initial top cells view of the design is created by the tool. The design in ddc format is translated into the standard cells view based on the physical design libraries. The result is shown in Figure 4.21. The white highlighted rectangular boxes are the standard cells of the design. The design consists of 5574 standard cells.

Floor-planning is a process of defining the area for placement and routing that meets certain constraints and without any routing violations and congestions. Floor-planning is very important because a bad floor-plan causes the waste-age of

die area, routing congestions and other issues. So, the good floor-plan will output an optimum layout.

The floor-plan of the design is created by using the GUI of ICC. The detail of the creating floor-plan is shown in Figure 4.22. The core utilization ratio is around 0.8. Core utilization ratio defines the area for standard cells, macros and blockage placement. In this case, 80% of area is available for standard cells, macros and blockage placement. The remaining 20% of area is for routing. The tool will define the optimum area for the design. The tool will assign 80% of the optimum area for placement and the remaining 20% of the optimum area for routing purpose. Besides, the gap between terminals and the core area is about 10 μm . After the floor-plan is created, the top view of the design is shown in Figure 4.23. The area inside the greenish-blue square is the core area that used for placement. The greenish-blue square is the terminals of the design and also known as I/O ports of the design. The pink colour small boxes are the standard cells.

After floor-plan initialization, the virtual placement is performed in order to verify the created floor-plan in term of routing congestions and timing violations. The result of the virtual placement is shown in Figure 4.24. The virtual placement report is shown in Figure 4.25. In this case, total number of cells violating the core area is 0. This means all cells are able to be placed into the core area (80% of the total design area). Figure 4.26 showed the chip summary after the virtual placement. The core utilization ratio is 80.75%. Previously, the decided value of the core utilization is 80%. The tool will calculate the optimum value for the core utilization ratio but the calculated value is closed to user defined value. So, the core utilization ratio that calculated by the tool is the optimum value that closed to the user defined value. Besides, the chip area is increased because the gaps between core area and I/O ports are added for four sides. The area will be optimized during the detail placement stage. Routing congestions happen when large amount of wire nets are routed in a narrow area. In this case, the routing area is 20% of the total area of the design. So, there was some possibility that the design consists of routing congestions. So, the congestions checking is important. The tool provides the automation to check the routing congestions. The tool performs global routing and then generate the congestion map and report.

During the global routing process, the tool divides the chip into global-route cells (GRCs). Each GRC consists of a finite number of routing wire. Then, the tool calculates the ratio of the amount of routed nets and the number of available nets on each GRC. If the amount of the routed nets larger than the quota of the nets on the GRC, the ratio is larger than 1 means the overflow happened in the design. So, the routing congestion exists on that particular grid-cell. The congestion report is shown in Figure 4.27. From the report, the total number of GRCs is 10100. In this case, no overflows happen on the GRCs. So, the design does not has routing congestion. Figure 4.28 showed a small portion of the GRCs in congestion map. The number that lied on the GRCs is the ratio of the amount of routed nets and the number of available nets on each GRC. The congestion map is used to visualize the congested GRCs. In this case, none of GRCs are congested. Therefore, the design passes the routing congestion and the design is ready for PNS and CTS.

Since the design does not consist of any pads like power pads and signal pads, so the virtual power pads (VDD pads and VSS pads) must be created before the execution of PNS. The design does not consists of those pads is because the instantiation of those pads are not defined in RTL codes. Since this design is not planned to fabricate because the libraries are not used for fabrication, so the pads are not important in this design. However, the power pads are required in order to perform PNS. Thus, the virtual power pads are created to fix this issue. Figure 4.29 showed the created virtual pads on the top hierarchical view of the design. The white colour highlighted areas are the virtual power pads. After that, the PNS is executed to analyse the power network of the design with certain power network constraints. The PNS can be done by the tool but the designer need to assign several parameters to the tool such as the names of the power nets (VDD and VSS in this case), value of the supply voltage, target IR drop (voltage drop) and power budget. The setting is shown in the Figure 4.30. The optimum power that required to power up the chip is around 342 mW which is obtained in logic synthesis stage. So, the power budget value is 342 mW. The supply voltage is 1.5 V and the target IR drop is the 10% of the supply voltage which is 0.15 V. After the tool generates the PNA (Power Network Analysis) power map in order to visualize the voltage drop levels on the top view of the design. The power map is shown in Figure 4.31. The area in red colour is the worst area. The wire nets in this area suffered from critical voltage drop. The worst area is lied on the

centre of the chip. As the length of the wire nets increase, the resistance of those wire nets also increase. So, the voltage drop also increases. The maximum voltage drop is 0.931 mV which is less than 10% of supply voltage (150 mV). Besides, the purple colour region is the least voltage drop region. The voltage drop in this region is 0.0093 mV. On the other hand, the PNS report for VDD nets is shown in Figure 4.32. The maximum voltage drop for VDD nets is 0.093 mV. The PNS for VDD nets is executed successfully. It is same for VSS nets. The maximum voltage drop for VSS nets is 0.087 mV. The PNS report for VSS nets is shown in Figure 4.33. Since the maximum voltage drop value does not exceed 10% of the supply voltage value, the power network of this design can be committed. The committed power network of the design is shown in the Figure 4.34. The power network is routed by several layers of the metal layers defined by the tool.

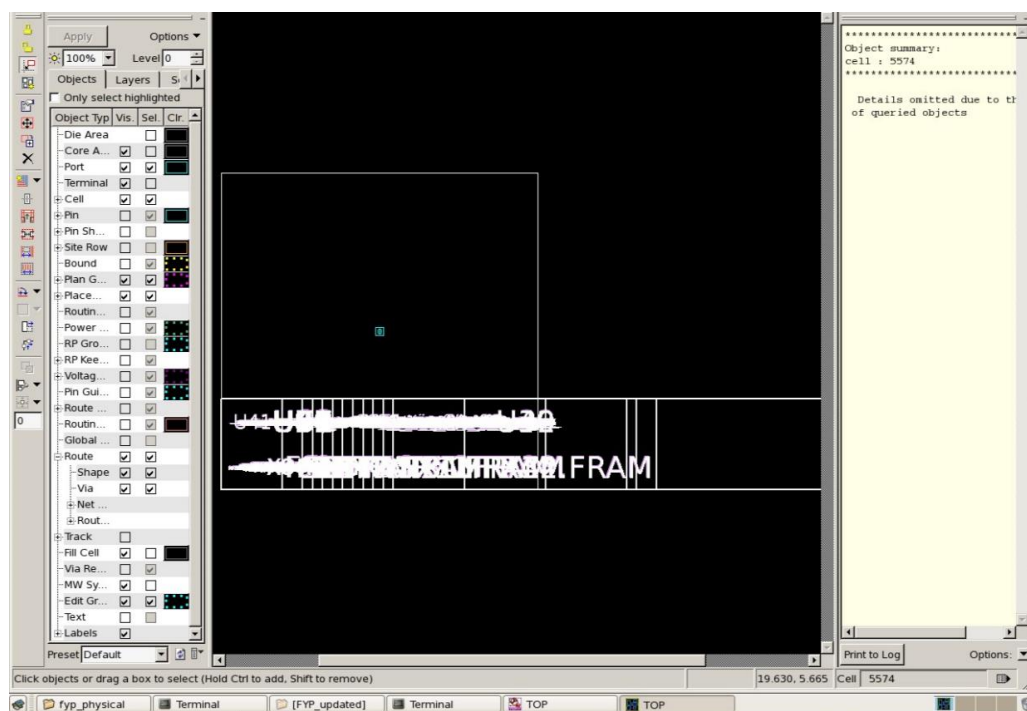


Figure 4.21: Initial Top Cell View of the Design

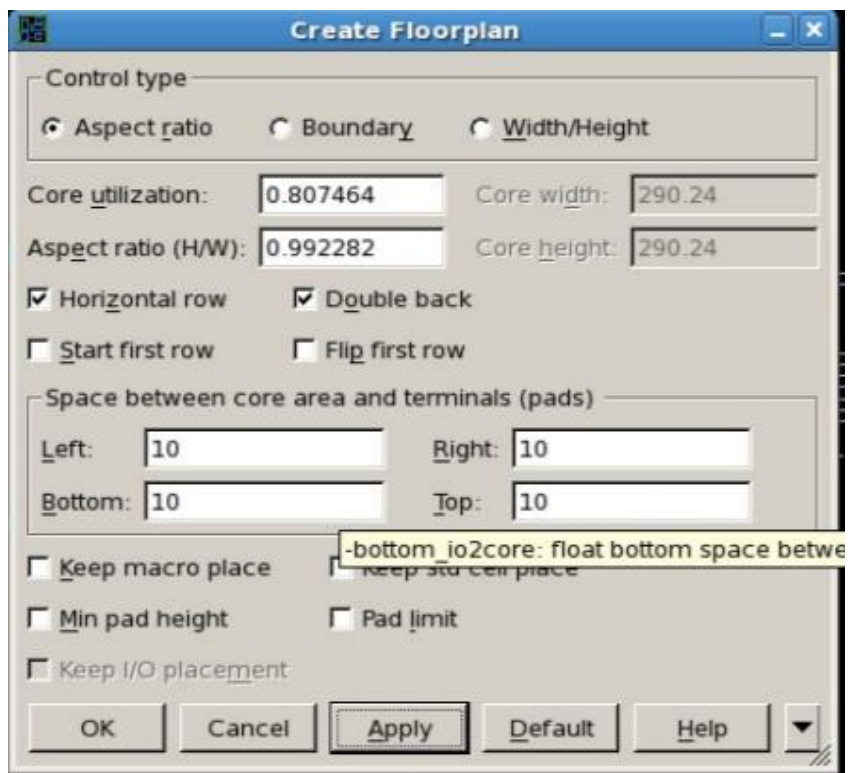


Figure 4.22: Creating Floor-plan

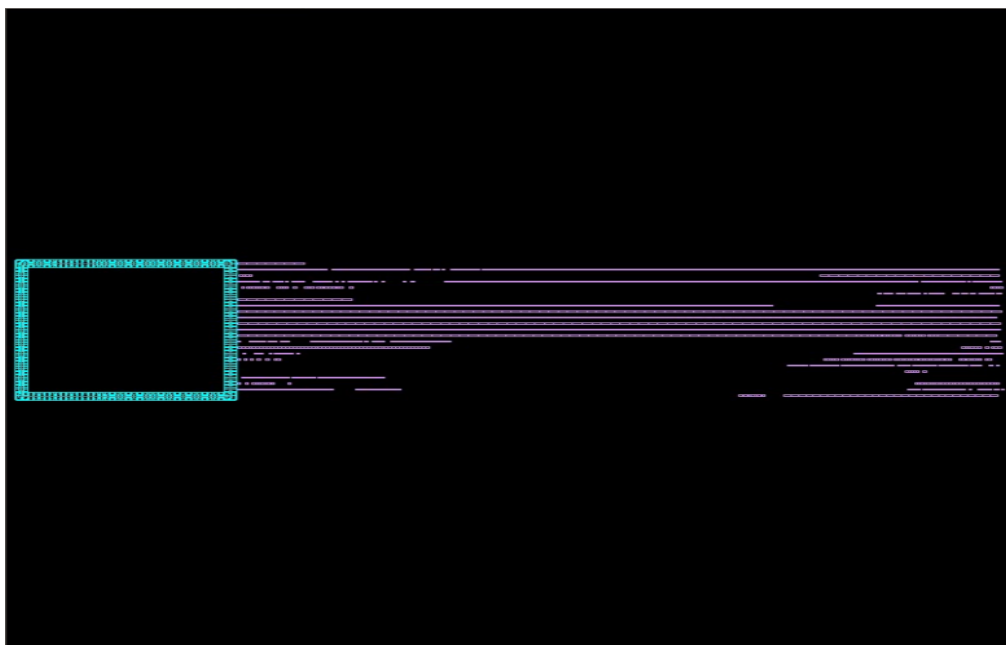


Figure 4.23: Created Floor-plan

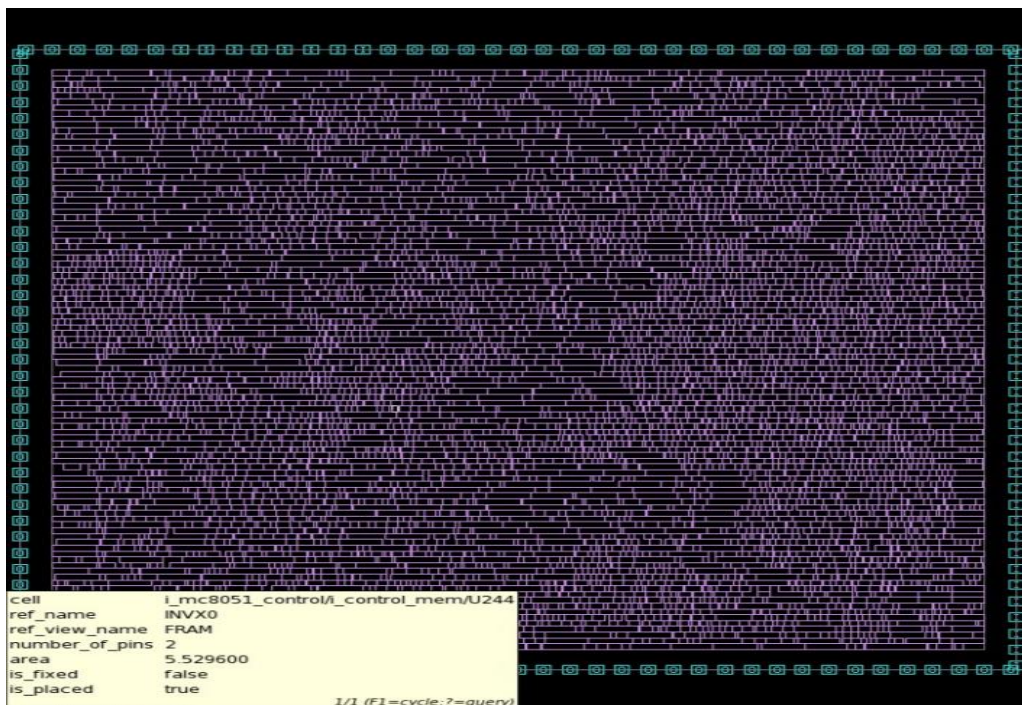


Figure 4.24: Virtual Placement

```

Number of cells violating core area: 0
Total number of cells violating plan group or core area: 0
*** global placement done.

```

Figure 4.25: Virtual Placement Report

```

Std cell utilization: 80.75% (73237/(90700-0))
(Non-fixed + Fixed)
Std cell utilization: 80.75% (73237/(90700-0))
(Non-fixed only)
Chip area:          90700 sites, bbox (0.00 0.00 290.24 288.00) um
Std cell area:      73237 sites, (non-fixed:73237 fixed:0)
                   5574 cells, (non-fixed:5574 fixed:0)
Macro cell area:    0 sites
                   0 cells
Placement blockages: 0 sites, (excluding fixed std cells)
                   0 sites, (include fixed std cells & chimney area)
                   0 sites, (complete p/g net blockages)
Routing blockages: 0 sites, (partial p/g net blockages)
                   0 sites, (routing blockages and signal pre-route)
Lib cell count:     43
Avg. std cell width: 5.06 um
Site array:         unit (width: 0.32 um, height: 2.88 um, rows: 100)
Physical DB scale: 1000 db_unit = 1 um

```

Figure 4.26: Chip Summary

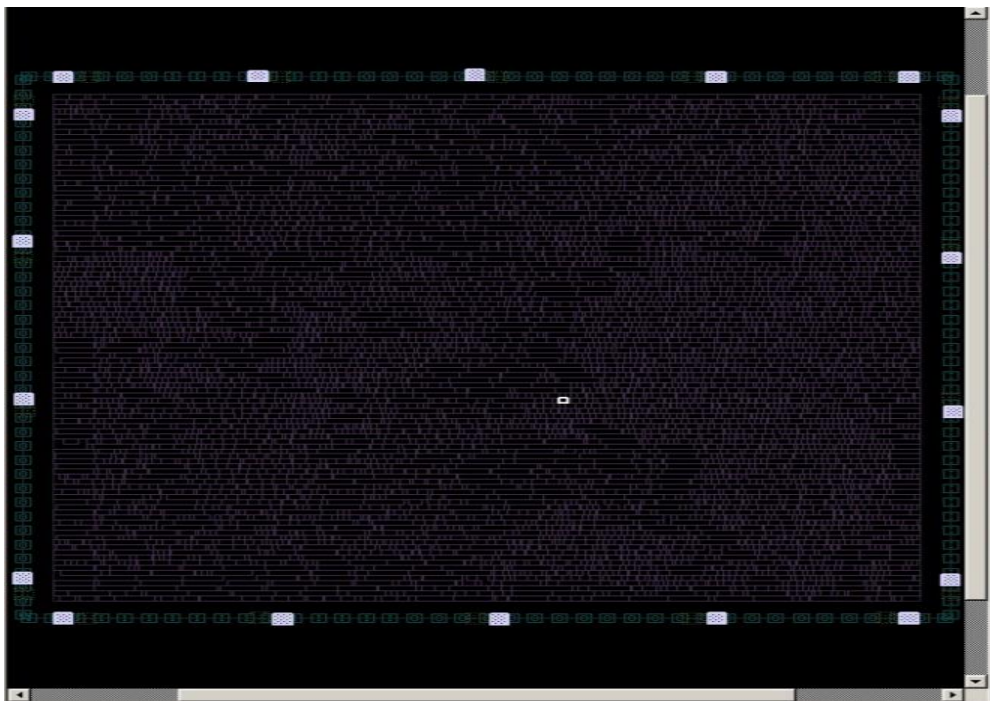


Figure 4.29: Virtual Power Pad on the Top Hierarchy of the Design

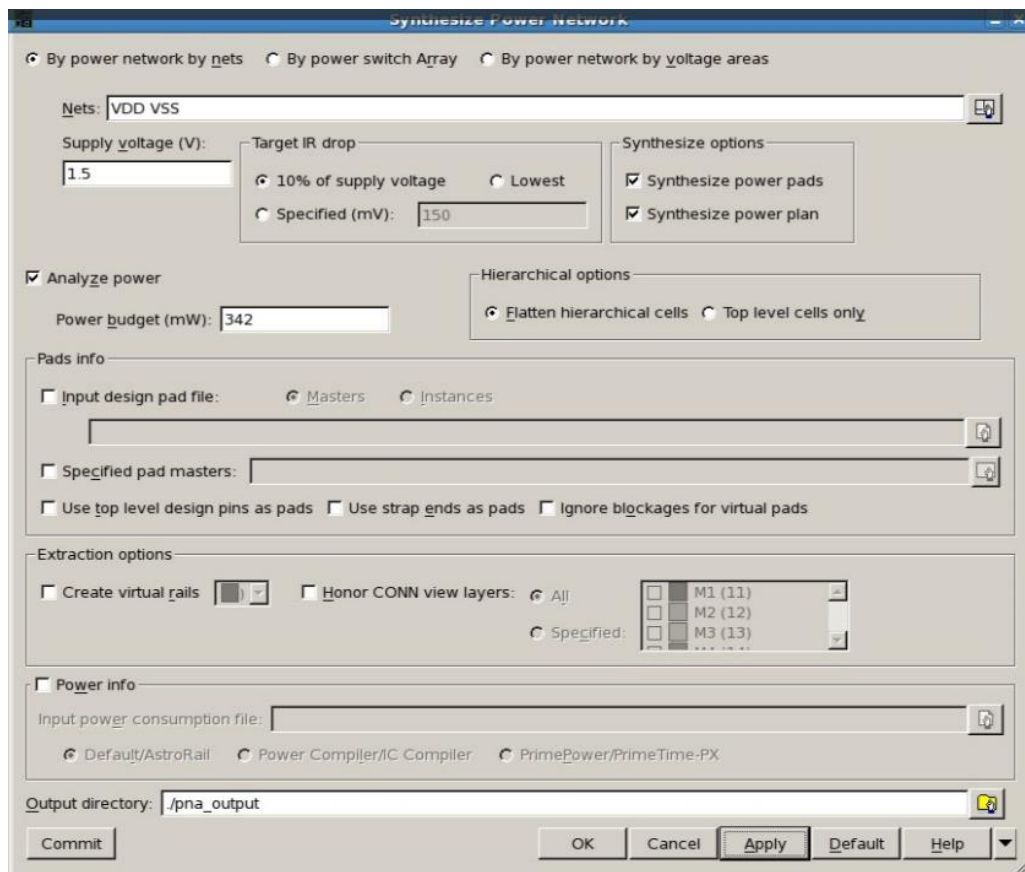


Figure 4.30: PNS setting

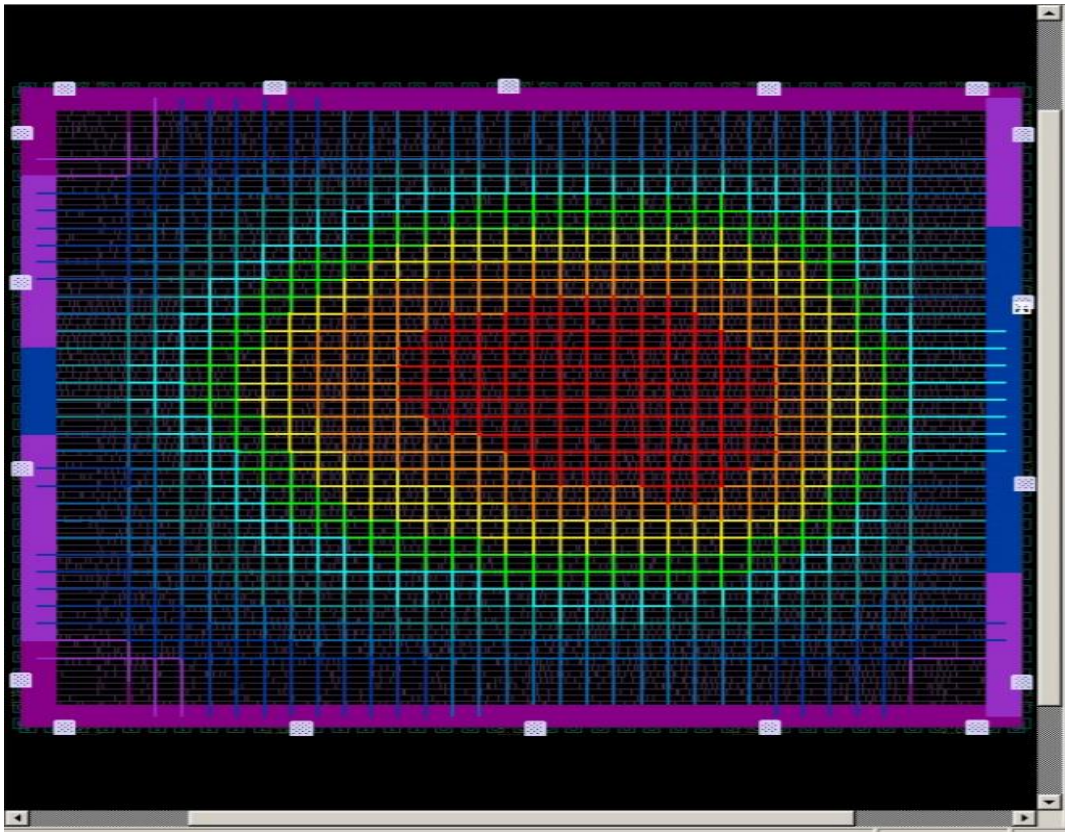


Figure 4.31: PNA Power Map

```
Total Current from Virtual Pads:    0.28 mA (100.00%)
Maximum IR drop in mc8051_core : 0.09 mV
Maximum current in mc8051_core : 0.032 mA
Maximum EM of wires in mc8051_core : 3.103923e-01 A/cm, layer M4
Maximum EM of vias in mc8051_core : 4.956384e+02 A/cm_square, layer VIA3
The PNS synthesizes the net VDD successfully
The maximum IR drop of the synthesized net VDD is    0.093 mV
```

Figure 4.32: PNS Report for VDD Nets

```
Total Current from Virtual Pads:    0.28 mA (100.00%)
Maximum IR drop in mc8051_core : 0.09 mV
Maximum current in mc8051_core : 0.034 mA
Maximum EM of wires in mc8051_core : 4.155510e-01 A/cm, layer M4
Maximum EM of vias in mc8051_core : 5.198328e+02 A/cm_square, layer VIA3
The PPS synthesizes the net VSS successfully
The maximum IR drop of the synthesized net VSS is    0.087
```

Figure 4.33: PNS Report for VSS Nets

4.2.3 Placement and Routing

After PNS, the design is ready for placement. At this stage, the detail placement is done. Before execution of the detail placement, the design must be checked for issues. The result is shown in Figure 4.34. From the result, the design consists of zero error and zero warning. Thus, the design is ready for placement.

The options for the placement is shown in Figure 4.35. The options are set to the tool so the tool performs the placement based on the options. The tool performs the placement with medium effort. The tool will not perform the congestion removal since the design does not consists of routing congestion. The tool also performs CTS during placement and optimizes the leakage power during placement. During the placement, the tool will perform area optimization, power optimization, timing optimization and CTS. Figure 4.36 showed the status of the placement. The placement is done several DRC violations. The violations will be fixed in the routing stage. Figure 4.37 showed the layout after placement with power nets routing. During placement, the CTS is done by the tool. Figure 4.38 showed the status of the CTS. The summary of the clock tree is shown in Figure 4.39. The DRC violations in the clock tree will be fixed in routing stage. After CTS, the routing of clock tree is done by the tool. In this case, the global routing is used by the tool. The result of the global routing is shown in Figure 4.40. The clock nets routing is successful. The total DRC violations is 41 violations. These violations can be fixed in detail routing stage. The total number of the clock nets is 6068.

After CTS, the next stage is routing. First initial routing is done by the tool. The tool performs global routing methodology for initial routing. The summary of the initial routing is shown in Figure 4.41. The initial route is success. The total DRC violations is 1844. The DRC violations can be fixed during detail routing stage. Then, the detail routing and optimization are performed by the tool. The effort of the routing is set to high in order to fix the DRC violations. The DRC violations are reduced to 1830. The result of the detail routing is shown in Figure 4.42. In order to solve the DRC violations, the ECO route is run. By default, the ECO routing runs detail routing to fix DRC violations. The result of the ECO routing is shown in

Figure 4.43. The number of the DRC violations is reduced. However, the DRC violations cannot be completely removed. The possible reason that causes this issue is the routing area. Most of the DRC violations are the minimum area and minimum spacing problems. The wire nets are routed too close to each other. Besides, the area of the wire nets are smaller than default area that defined in the technology libraries. In fact, the routing area of this design is 20% of the total area of the design. In this case, the routing area is not enough for the routing. Thus, the routing area should be larger than 20% of the total area. So, the floor-planning stage must be run again in order to solve those DRC violations.

```
Total messages: 0 errors, 0 warnings
dump check_physical_design result to file ./cpd_pre_place_opt_2016Aug18143221_3523/index.html
1
```

Figure 4.34: Checking Design Report

```
The options for place_opt:
-----
POPT: place_opt effort level           : medium
POPT: Congestion removal               : No
POPT: Area recovery                    : Yes
POPT: Optimize dft                     : No
POPT: Clock Tree Synthesis             : Yes
POPT: Optimize power                   : Yes
POPT: Optimize power mode              : Leakage
-----
```

Figure 4.35: Options for Placement

```
Placement Optimization Complete
-----

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

Nets with DRC Violations: 65
Total moveable cell area: 68460.1
Total fixed cell area: 32.3
Total physical cell area: 68492.4
Core area: (10000 10000 300240 298000)
```

Figure 4.36: Status of Placement

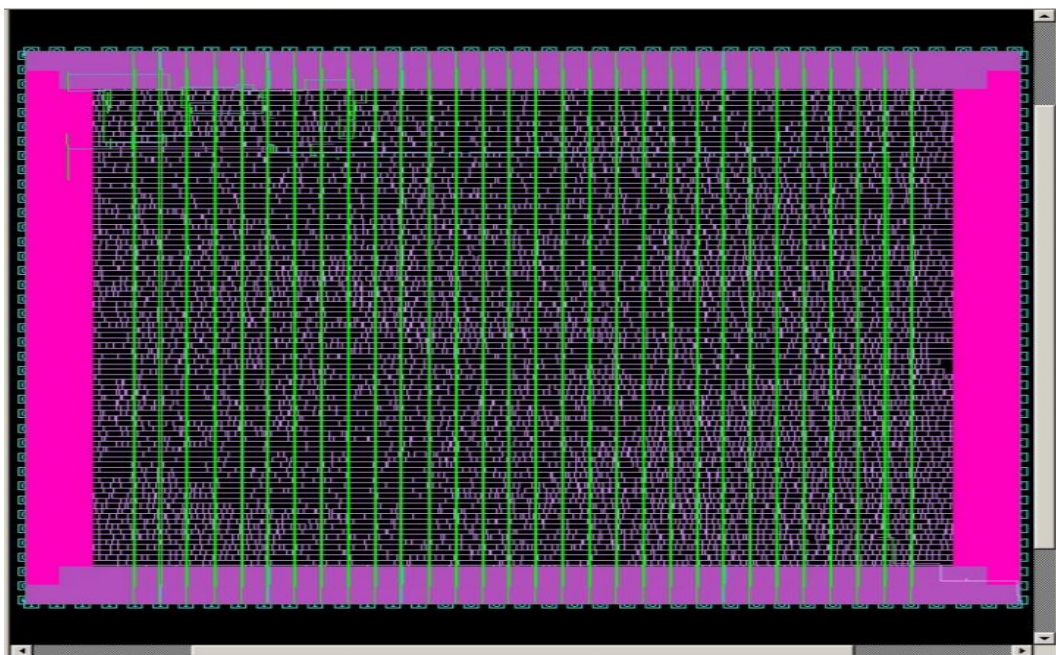


Figure 4.37: Layout after Placement and Power Nets Routing

```
Optimize clock tree Successful...
Report clock tree summary results after clock tree optimization
```

Figure 4.38: CTS Status

```
===== Clock Tree Summary =====
```

Clock	Sinks	CTBuffers	ClkCells	Skew	LongestPath	TotalDRC	BufferArea
clk	48	2	2	0.0041	0.6459	3	32.2560

Figure 4.39: Clock Tree Summary

```
Total number of nets = 6068
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 = 41
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)
Routing of clock nets Successful.
RC Extraction...
```

Figure 4.40: Result of Clock Tree Routing (Global Routing)

```

Total number of nets = 5953
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 = 1844
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)

```

Figure 4.41: Summary of the Initial Routing

```

Total number of nets = 5972
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 = 1830
Total number of antenna violations = antenna checking not active
Total number of voltage-area violations = no voltage-areas defined
Total Wire Length =                233974 micron
Total Number of Contacts =          52404
Total Number of Wires =              50917
Total Number of PtConns =            6212
Total Number of Routable Wires =     50917
Total Routable Wire Length =         232492 micron

```

Figure 4.42: Summary of Detail Routing and Optimization

```

Total number of nets = 5972
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 = 1783
Total number of antenna violations = antenna checking not active
Total number of voltage-area violations = no voltage-areas defined
Total Wire Length =                234390 micron
Total Number of Contacts =          52548
Total Number of Wires =              50927
Total Number of PtConns =            6222
Total Number of Routable Wires =     50927
Total Routable Wire Length =         232908 micron

```

Figure 4.43: Summary of ECO Routing

4.2.4 Design Rule Checking Fixing and Final Verifications

In order to solve the Design Rule Checking (DRC) violations, floor-planning needs to be executed again. The methodology is same as previous. All steps for previous physical design stage remain the same except for floor-plan initialization. In this time, the core utilization ratio is set to 0.5. 50 % of the design area is for placement and remaining 50% of the design area is for routing. Although the size of the chip might be slightly larger but it is efficient in DRC fixing. After the floor-plan is initialized, the congestion analysis is executed. The congestion heat map is shown in Figure 4.44. From the heat map, the design does not consist of routing congestions. The several GRCs on the design is shown in Figure 4.45. From the result, there are no overflows exist. The ratios on the GRCs are not exceed 1. Therefore, the design does not consist of routing congestions. The power map of the design is shown in Figure 4.46. The maximum voltage drop value is about 0.0673 mV. The voltage drop value is slightly higher than previous value. This is because the length of the power nets becomes longer since the design become slightly larger than previous one. Fortunately, the maximum voltage drop value is still less than 10% of the voltage supply which is 150 mV. So, the design passes the PNS. The critical area of the voltage drop is still in the centre of the design. After that, the power network of the design is route by the tool with automation. In this time, the top metal layer for power network is changed to metal layer 9 in order to fix the density violations.

The following steps are also the same as those in placement and routing stage. All standard cells are placed into core area by using design automation. Further optimization is done by the tool. There are no critical issues in placement, CTS and routing stages.

Some DRC violations still exist the design. So, the ECO routing is run again to fix DRC violations. The result of the ECO routing is shown in Figure 4.47. Based on the result, the DRC violations are fixed. No DRC errors are existed in the design. On the other hand, the Layout versus Schematic (LVS) checking is done by using the tool. The result is shown in Figure 4.48. The electrical interconnections in the layout are same as the interconnections in the gate-level netlist that generated in logic

synthesis. The total must joint nets is zero means all wire nets defined in netlist are routed correctly. There are some minor LVS violations exist. The minor violations are floating port violations, shorted nets violations and open-nets violation. Some of the ports in the design are not connected to any wire nets. So, the ports are defined as floating ports. These violations can be ignore since it will not affect the layout in term of functionality if the design. Besides, the shorted nets are caused by the improper naming of the ports between modules. Technically, the connections are correct based on the schematic but the tool recognizes the nets are shorted since the two ports have different name. To solve the problems, buffers can be added to the nets but adding buffers causes the size of the chip become larger again. So, these violations are not to be fixed since it will not affect the design in term of functionality. Lastly, the open-nets are caused by the absent of the I/O pads. In fact, the top metal layers should connected with the I/O pads for external interfacing. Since this design does not consists of the I/O pads, so the violations are present. The violations can be ignored.

The final layout is shown in Figure 4.49. This top hierarchical view of the layout. The design is routed by nine metal layers. The bottom level is the standard cell. The metal fillers and filler cells are inserted into the design in order to solve the density violations. The centre of the layout is shown in Figure 4.50.

Before stream out, several verifications are done. The final timing report is shown in Figure 4.51. Based on the report, the timing closure is improved compared to the previous report. The CTS helps to improve the timing closure. The design will not suffer from any timing issues. Besides, final area analysis is done by the tool. The report is shown in Figure 4.52. The chip area is slightly larger compared to the previous result. This is because the buffers are added into the design in order to solve the timing issues and shorted nets issues. Besides, the routing area is increased in order to fix the DRC violations like minimum spacing and minimum area of the nets. On the other hand, the power report is shown in Figure 4.53. The tool has successfully optimized the cell leakage power. The cell leakage power is reduced. Unfortunately, the cell internal power and net switching power are increased. The timing closure is improved therefore the sufficient switching power is required for high speed switching. The amount of the clock tree nets also causes the switching

power to increase since the number of switching nets is increased. As the switching nets are increased, the total capacitance of the switching nets increases also and eventually causes the net switching power dissipation to increase. The power constraint is skewed down in order to improved performance (timing) and area of the design. There are trade-off relationship between power, timing and area. In order to improve timing and area of the design, the power constraint must be skewed down.

Lastly, the GDSII file is streamed out by using the tool. The status of the stream out process is shown is Figure 4.54. The GDSII file is used for fabrication. In this case, this design is not for fabrication because the technology libraries are not for fabrication.

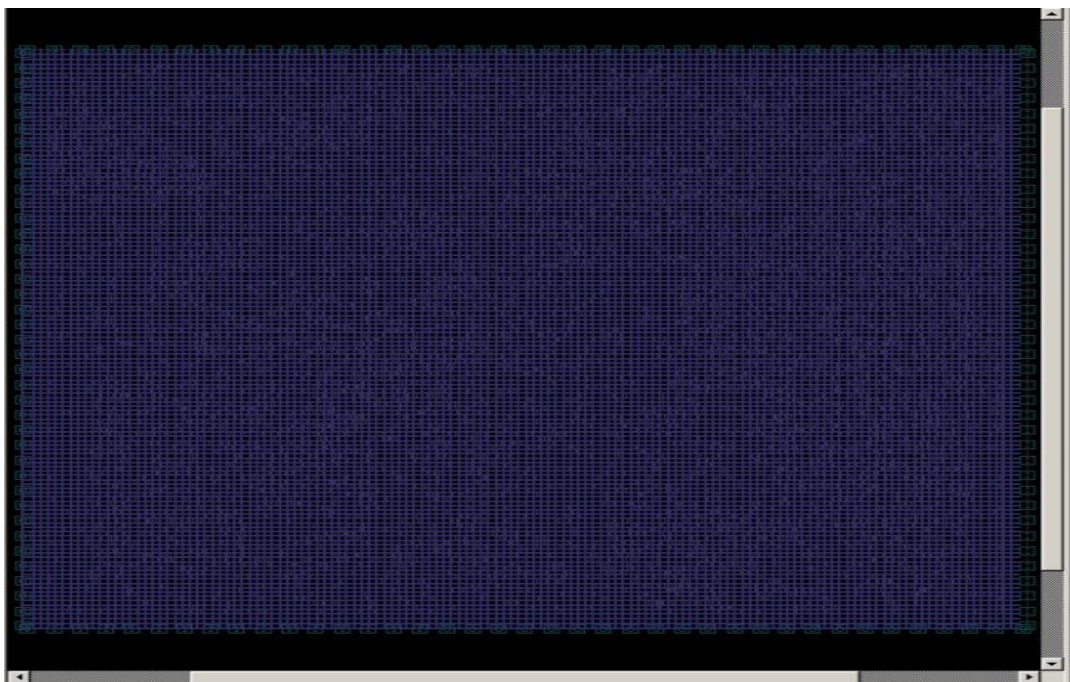


Figure 4.44: Congestion Heat Map

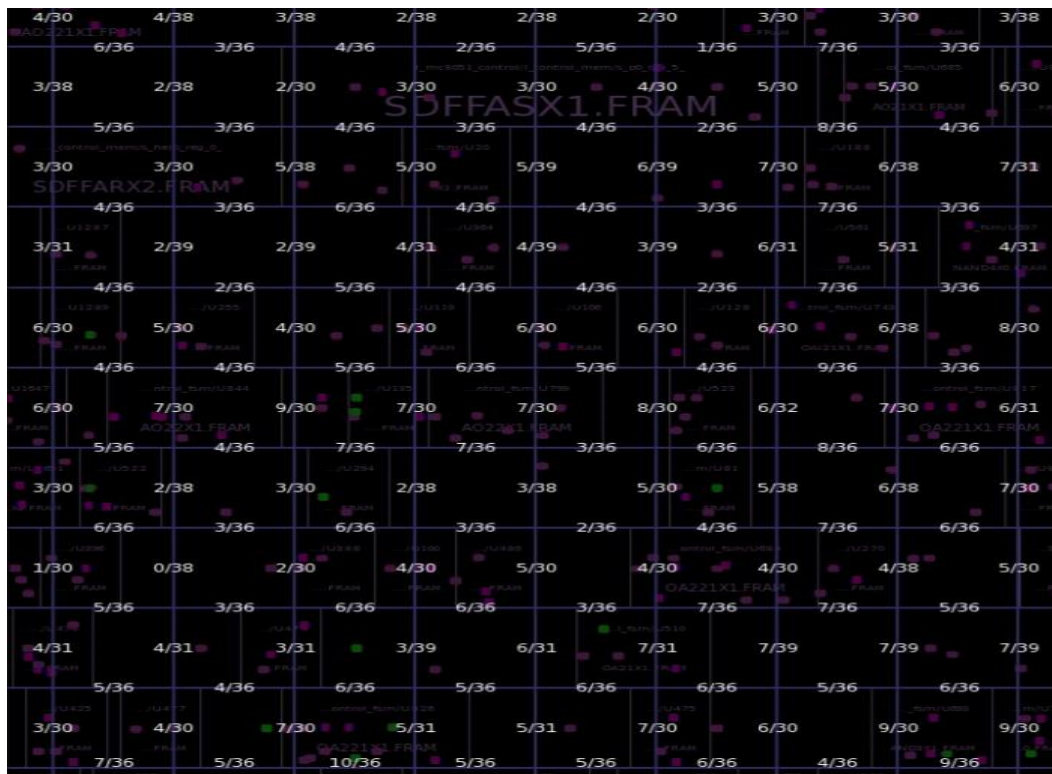


Figure 4.45: Congestion Heat Map with Ratios

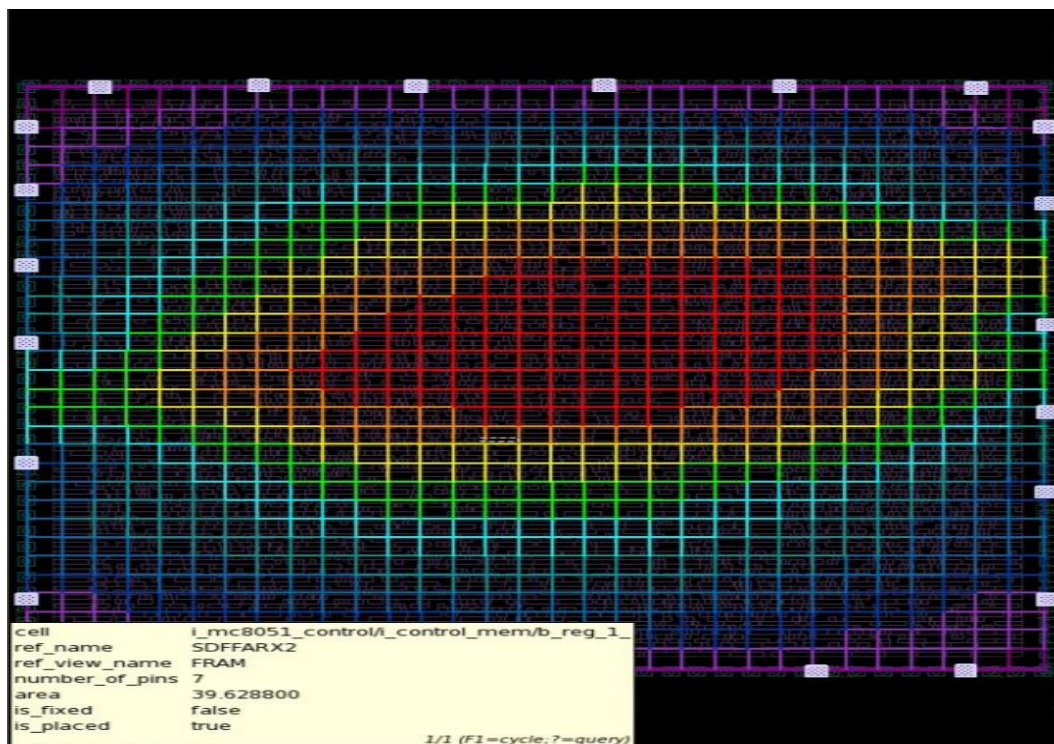


Figure 4.46: Power Map of the Design

```

Total number of DRCs = 0
Total number of antenna violations = antenna checking not active
Total number of voltage-area violations = no voltage-areas defined
Total Wire Length =                236721 micron
Total Number of Contacts =          51955
Total Number of Wires =              47181
Total Number of PtConns =           379
Total Number of Routable Wires =    47181
Total Routable Wire Length =        236646 micron

```

Figure 4.47: Result of the ECO Routing

```

** Total Electrical Equivalent Error are 0.
** Total Must Joint Error are 0.

```

Figure 4.48: Result of LVS Verification

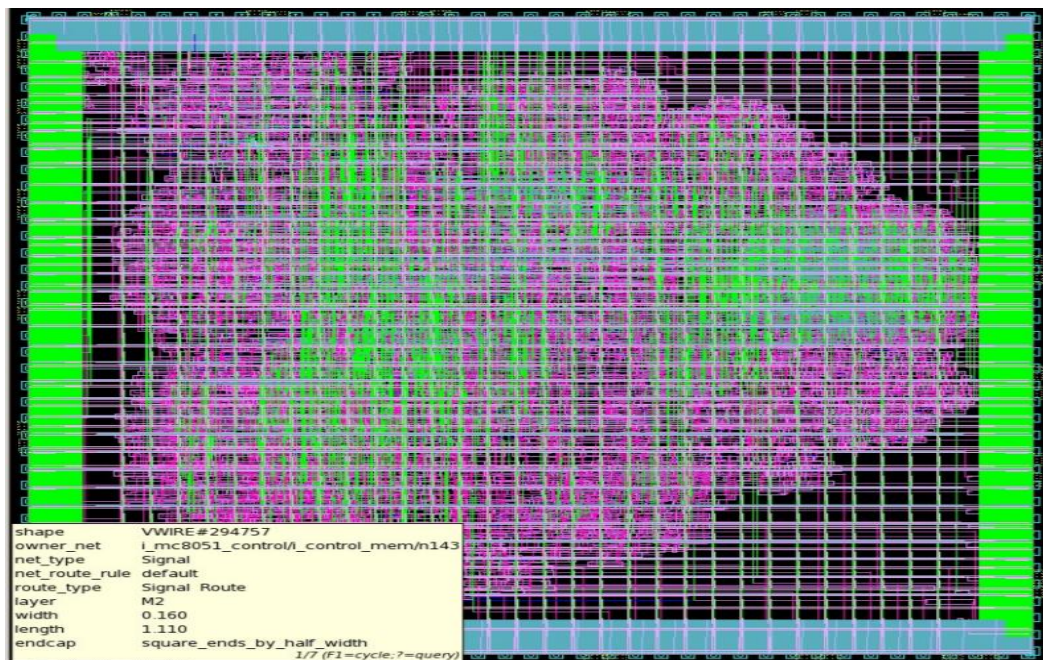


Figure 4.49: Final Layout

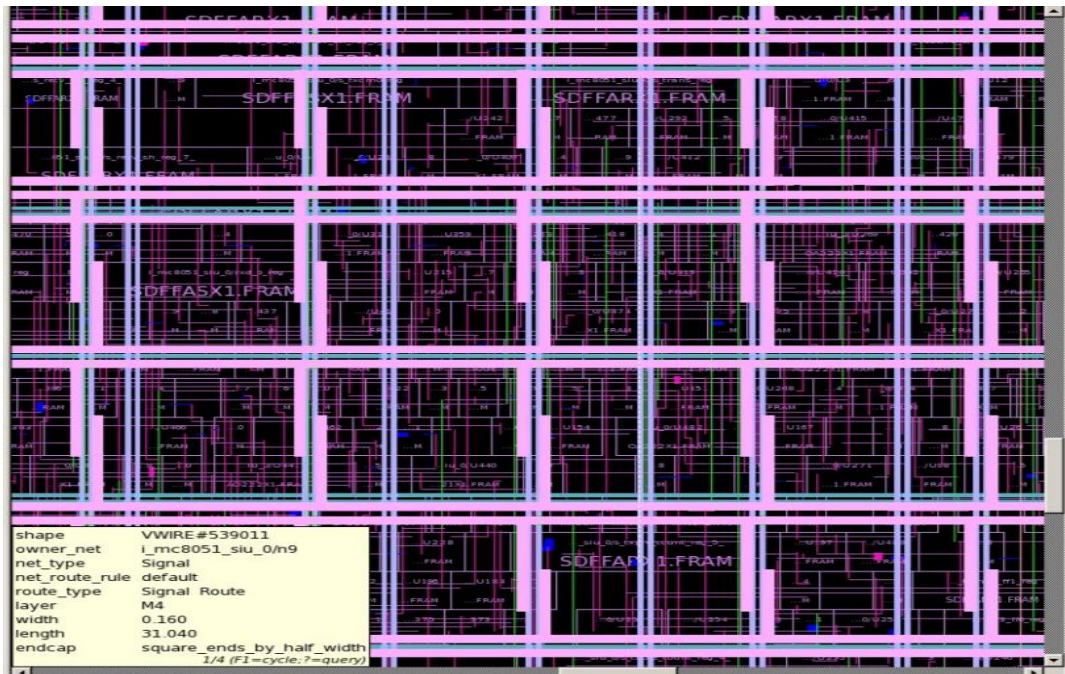


Figure 4.50: Centre of the Layout

clock clk (rise edge)	6.67	6.67
clock network delay (propagated)	1.64	8.30
clock uncertainty	-0.20	8.10
i_mc8051_tmrctr_0/s_count10_reg_7 /CLK (SDFFARX1)	0.00	8.10 r
library setup time	-0.47	7.63
data required time		7.63

data required time		7.63
data arrival time		-5.58

slack (MET)		2.05

Figure 4.51: Final Timing Report

```

Number of ports:                156
Number of nets:                 437
Number of cells:               166
Number of combinational cells: 162
Number of sequential cells:    0
Number of macros/black boxes:  0
Number of buf/inv:            162
Number of references:          11

Combinational area:            47629.209712
Buf/Inv area:                  8173.670498
Noncombinational area:        21770.956055
Macro/Black Box area:         0.000000
Net Interconnect area:        7849.649084

Total cell area:               69400.165767
Total area:                    77249.814850

```

Figure 4.52: Final Area Report

```

Cell Internal Power = 143.1559 uW (36%)
Net Switching Power = 256.6308 uW (64%)
-----
Total Dynamic Power = 399.7867 uW (100%)
Cell Leakage Power  = 194.2029 uW

Power Group      Internal      Switching      Leakage      Total
                 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   38.9558        127.7010       1.2949e+06     167.9517 ( 28.28%)
register        11.8236         8.3318         5.0552e+07     70.7075 ( 11.90%)
sequential      0.0000         0.0000         0.0000         0.0000 ( 0.00%)
combinational   92.3768        120.5980       1.4236e+08     355.3307 ( 59.82%)
-----
Total           143.1562 uW    256.6307 uW    1.9420e+08 pW  593.9899 uW

```

Figure 4.53: Final Power Report

```

Outputting Contact $$VIA45C_300_300_1_33
Outputting Contact $$VIA45C_300_300_10_1
Outputting Contact $$VIA45C_300_300_33_1
Outputting Contact $$VIA45C_300_300_11_1
write_gds completed successfully!
1

```

Figure 4.54: Status of Stream Out Process

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

The VLSI design flow from RTL to GDSII is successfully implemented. The design flow is done by using the Synopsys Design Compiler and Synopsys IC Compiler. The tools provide design automations which can automate the design tasks which include mapping and optimize the process, STA, area analysis, auto place and route, CTS, floor-planning, PNS and others. The automations provided by the EDA tools are able to help to reduce the design TAT. The total period that is used to implement this microcontroller core is about six months. Without design automations, the design cannot be implemented within six months. Therefore, design automations have reduced the design TAT efficiently. Besides, the EDA tools are able to handle complex designs like SoC. Human are not able to design complex SoC manually without any helps from EDA tools. Thus, EDA tools are important in IC design industry.

The logic synthesis stage is done without any violations like timing violations and constraints violations. The clock frequency is about 150 MHz. The clock frequency is improved compared to the original 8051 microcontroller which is 12 MHz (Intel Corp., n.d.). The clock frequency of this design is improved up to 12 times of original clock frequency. Thus, the performance of the design is improved.

Besides, the physical design stage is done. The area of the design is optimized. The area of the design is smaller than original design. Besides, the timing closure is

improved. The slack is larger than previous case. Lastly, the power constraint of the design is skew down. The required power is slightly larger than original design. The power constraint is sacrificed due to the improvement of the performance and area constraints. Overall, the design has been improved in terms of performance and area constraints. The summary of the design improvement is shown in Table 5.1. The GDSII file is generated.

Table 5.1: Summary of the Design Improvement

Original Result	Final Result
Clock Frequency: 12 MHz (Intel Corp., n.d.)	Clock Frequency: 150 MHz
Area: 104984.540474 μm^2 (Based on saed90nm_max library)	Area: 77249.814850 μm^2

5.2 Recommendations

In future, the performance of the design in term of the clock frequency can still be improved further. The MIPS (Million Instructions per Second) rate is important for new model of microcontroller. As the performance of the microcontroller is improved, the MIPS rate is increased also. The MIPS rate is depends on the clock frequency. The clock frequency is also known as clock cycle per second. The microcontroller requires clock cycles to execute the instructions. Some of the instructions requires two clock cycles to be executed. Besides, several instructions requires three clock cycles to be executed. So, the clock frequency still can be improved in order to improve the MIPS rate. The performance of the design depends on the MIPS rate. Thus, the MIPS rate should be larger in order to achieve the higher performance of the design. Therefore, the clock frequency should be increased again in order to increase the MIPS rate.

I/O pads are not taken into consideration in current design. Hence, the design cannot interface with any external devices. In order to interface with external devices, the I/O pads can be implemented into the layout of the design. The I/O pads can be

instantiated in the RTL code. Then, the I/O pads are synthesized into gate-level netlist. Lastly, the I/O pads can be implemented into layout. This can be done by using ICC.

The chip area can be reduced by using other industrial based technology libraries with smaller process. In this project, the technology process is 90nm. This library is educational based library. The smaller technology process like 14 nm can be used. As the technology process becomes smaller, the chip area definitely becomes smaller. The industrial based libraries consist of industrial based standard cells and macros which are not in educational based library. The size of the standard cells and macros in the industrial based library definitely is smaller than the educational based library. Thus, the design area can still be reduced by using industrial based library with smaller process.

As mentioned earlier, this design cannot be fabricated because of the technology library. In order to fabricate, the design can be implemented by using the industrial based technology library that provided by foundry. By using different library, the design flow needs to be executed again in order to generate the final layout and GDSII file for fabrication.

REFERENCES

- Barr, K.E. 2007. ASIC Design in the Silicon Sandbox. New York: McGraw-Hill.
- Bernard, V. and Dominique, B. 1989. Logic Synthesis for VLSI. In: *Walter, E.P. and Hans, R. ed. VLSI and Computer Peripherals: 3rd Annual European Computer Conference, 8/9/10/11/12 May 1989, Hamburg*. New Jersey: IEEE, pp.5/10-5/14.
- Chang, F.C., Kwok, M. and Kenneth, R. 1999. In: *Silicon-Level Physical Verification of SubWavelength Designs. In: Twelfth International Conference on VLSI Design, 7/8/9/10 January 1999, Goa*. New Jersey: IEEE, pp.603-605.
- Chang, S.H. and Kim, S.D. 2006. Reused-based Methodology in Developing System-on-Chip (SoC). In: *Fourth International Conference on Software Engineering Research, Management and Applications, 9/11 August 2006, Seattle*. New Jersey: IEEE, pp.125-131.
- Chen, W.K. ed. 2009. Computer Aided Design and Design Automation. New York: CRC Press.
- Chirag, P. 2012. Logic Synthesis. [Online] Available at: <http://asicpd.blogspot.my/2012/08/logic-synthesis.html>
- Das, D. 2010. VLSI Design. New Delhi: Oxford University Press.
- Derek, L. 2016. Place and Route using Synopsys IC Compiler. [Online] Available at: <http://www.csl.cornell.edu/courses/ece5745/handouts/ece5745-tut3-icc.pdf>

- Ethan, M. 2006. Establishing Moore's Law. *IEEE Annals of the History of Computing*. 28(3), pp. 62-75.
- Eugenio, V. and Pablo, S. 1995. CAD Tools for Synthesis. In: *ISIE '95 Proceedings of the IEEE International Symposium on Industrial Electronics, 10/11/12/13/14 July 1995, Athens*. New Jersey: IEEE, pp.27-32.
- Intel Corp. (no date). 8051 Datasheet. [Online] Available at: <http://html.alldatasheet.com/html-pdf/107780/INTEL/8051/300/2/8051.html>
- Intel Corp. 2015. Moore's Law and Intel Innovation. [Online] Available at: <http://www.intel.com/content/www/us/en/history/museum-gordon-moore-law.html>
- Jeffery, B., Arun, S. and Naveed, S. 1993. Physical Tradeoffs for ASIC Technologies. In: *AISC Conference and Exhibit, 1993: Proceedings., Sixth Annual IEEE International, 27 September – 1 October 1993, New York*. New Jersey: IEEE, pp.70-78.
- Jiang, W.Y., Quan, X. and Zhou, S. 2008. Historical, Entrepreneurial and Supply Chain Management Perspectives on the Semiconductor Industry. In: *Portland International Center For Management Of Engineering And Technology, 27/31 July 2008, Cape Town*. Portland: IEEE, pp.2552-2559.
- Kang, S.M., Leblebici, Y. and Kim, C. 2015. CMOS Digital Integrated Circuits: Analysis & Design. 4th ed. Singapore: McGraw-Hill
- Oregano System. 2013. 8051 IP Core. [Online] Available at: http://www.oreganosystems.at/?page_id=96
- Rajesh, M.A., Soujanya, R., Kalpashree, M.A. and Soumya, S. 2014. Automated Physical Verification of I/O Pads in Full-Custom Environment. In: *2014 Fifth International Symposium on Electronic System Design, 15/16/17 December 2014, Surathkal*. New Jersey: IEEE, pp.203-205.

- Ryota, K., Kennosuke, F., Kazuo, T., Hitoshi, K. and Shoji, H. 1985. An Integrated Modular and Standard Cell VLSI Design Approach. *IEEE Transactions on Electron Devices*. 32(2), pp.487-492.
- Samar, K.S. 2013. Emerging Business Trends in the Semiconductor Industry. In: *2013 Proceedings of PICMET '13: Technology Management in the IT-Driven Services, 28 July-1 August 2013, San Jose*. New Jersey: IEEE, pp.2744-2748.
- Steven, L. and Zhao, B. 2010. Tutorial 5: Synthesis with Synopsys and Encounter. [Online] Available at: http://www.engrclasses.pitt.edu/electrical/faculty-staff/levitan/1192/2008/Tutorials/Tutorial5N/Tutorial_Synthesis.html
- Steven, M.R. 1994. Computer Aids for VLSI Design. [Online] Available at: <http://www.rulabinsky.com/cavd/text/chapc.html>
- Synopsys, Inc. 2007. Design Compiler. [Online] Available at: <http://training.synopsys.com>
- Synopsys, Inc. 2007. IC Compiler. [Online] Available at: <http://training.synopsys.com>
- Thomas, F. and William, G. 2010. Synopsys Tutorial: Using the Design Compiler. [Online] Available at: http://lyle.smu.edu/~manikas/CAD_Tools/SDC/lab2/lab2_synopsys_dc.pdf
- Yun, J. and Ha, D.S. 2009. Design Vision – Verilog Logic Synthesis Tool. [Online] Available at: http://www.vtvt.ece.vt.edu/vlsidesign/tutorialSynopsys_VerilogSynth.php

APPENDICES

APPENDIX A: Project Timeline

	Feb	Mar	Apr	May	Jun	Jul	Aug
Collecting Information							
RTL Stage and Debugging							
Logic Synthesis with sc_max Library							
Logic Synthesis with Saed90_max Library							
Physical Design							
Physical Design (2 nd iteration)							
Final Verification							

APPENDIX B: Synopsys Start-up File for Design Compiler

```
# .synopsys_dc.setup file

#search path
set search_path "$search_path unmapped unconstrained mapped rtl lib cons"

#library setting
set target_library saed90nm_max.db
set link_library "* $target_library"
set symbol_library saed90nm.sdb;

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

#store used commands
history keep 1000

#alias
alias rt "report_timing -nets"
alias rc "report_constraint -all_violators"
alias rd "report_design"
alias ct "check_timing"
alias q "quit"
```

```
alias h "history"
alias ptl "printvar target_library"
alias pll "printvar link_library"
alias psl "printvar symbol_library"
alias ra "report_area"
alias save "write -f ddc -hier -out mapped/mc8051_core.ddc"
alias con "source scripts/mc8051_core.con"
```

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

# set alib_library_analysis_path [get_unix_variable HOME]
set alib_library_analysis_path ..

read_vhd mc8051_core.vhd

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

APPENDIX C: Synopsys Start-up File for IC Compiler

```
# -----  
# Suppress known and/or annoying messages  
# -----  
suppress_message {PSYN-040 PSYN-088 PSYN-058 PSYN-039 PSYN-024 RCEX-  
060 PSYN-087 PSYN-850 TFCHK-055}  
  
# Suppress warnings about metal layer pitch that occurs during create_mw_lib:  
suppress_message {TFCHK-049 TFCHK-050}  
  
# Suppress warning that "43 logical cells do not have P/G pins" from  
check_mv_design -power_nets:  
suppress_message {MV-597}  
  
# Suppress warning about ignored DEF syntax and "Information" about "preferred  
wire track direction  
# not being set" during read_def:  
suppress_message {DDEFR-054 MWDEFR-159}  
  
# Suppress warning about "skipping AHFS on don't touch high-fanout nets" during  
place_opt:  
suppress_message {PSYN-1002}  
  
# Suppress warnings about "Ignore pin on layer 0", "Ignore top cell pins with no  
ports",  
# and "METAL pitch too small" during route_zrt_global -congestion_map_only true:  
suppress_message {ZRT-026 ZRT-027 ZRT-030}
```

```

# Suppress warning about "P/G ports being on non-routing layer "UNKNOWN" "
and warning about
# using Elmore instead of "clock_arnoldi" delay calculation model during clock_opt:
suppress_message {MWLIBP-311 CTS-352}

# Suppress warning about "not enough nets being routed" during route_opt:
suppress_message {RCEX-047}

# Suppress warning: Power connection/checking is skipped for 2666 power pins
because the required power pin information cannot be found in logical libraries.
suppress_message {MV-510}

# -----
# Load useful "functions" or procedures, like "view"
# -----
source ../ref/tools/procs.tcl

# -----
# General useful settings
# -----
# Disable more-like page mode
set_app_var enable_page_mode false
# Don't want to see CMD-041 when creating new variables
set_app_var sh_new_variable_message false
# Increase history buffer from 20 commands to 100
history keep 100

# -----
# Enable logging of commands and everything by date/shell
# -----
set timestamp [clock format [clock scan now] -format "%Y-%m-%d_%H-%M"]
set sh_output_log_file "${synopsys_program_name}.log.[pid].$timestamp"
set sh_command_log_file "${synopsys_program_name}.cmd.[pid].$timestamp"

```

```
# -----  
# Aliases  
# -----  
alias v view  
alias rt "report_timing -nosplit"  
alias rtm "report_timing -nosplit -delay min"  
alias rc "report_constraint -all_violators -nosplit"  
alias rq report_qor  
alias h history  
alias _ measure_time  
  
# -----  
# Logic Library settings  
# -----  
lappend search_path ./libs/db ./libs/tlup  
set_app_var target_library "saed90nm_max.db"  
set_app_var link_library "* saed90nm_max.db saed90nm_io_max.db"  
set_min_library saed90nm_max.db -min_version saed90nm_min.db  
set_min_library saed90nm_io_max.db -min_version saed90nm_io_min.db  
  
# -----  
# mc8051_core setup variables  
# -----  
set my_mw_lib mc8051_core.mw  
set mw_path "./libs/mw_lib"  
set tech_file " ./libs/tech/saed90nm_icc_1p9m.tf"  
set tlup_map "saed90nm.map"  
set tlup_max "saed90nm_1p9m_1t_Cmax.tluplus"  
set tlup_min "saed90nm_1p9m_1t_Cmin.tluplus"
```

```
set top_design "mc8051_core"  
set verilog_file "./design_data/mc8051_core.v"  
set sdc_file    "./design_data/mc8051_core.sdc"  
set ddc_file    "./design_data/mc8051_core.ddc"
```

```
set mw_logic0_net "VSS"  
set mw_logic1_net "VDD"
```