ARCHITECTURE EVALUATION FOR PROGRAMMABLE LOGIC IN 65NM

Major Project Report (SEM. III-IV)

Submitted in partial fulfillment of the requirements for the degree of

Master of Technology
IN

ELECTRONICS & COMMUNICATION ENGG.
(VLSI DESIGN)

By

Chandresh Hirani
(06MEC006)

Under the Guidance of

Mrs. Namerita Khanna,
Project Manager,
eCL Group,
STMicroelectronics,
Greater Noida.

Prof. N.P.Gajjar,
E.C. Department,
Institute of Technology,
Nirma University,
Ahmedabad.

Department of Electronics & Communication Engineering
INSTITUTE OF TECHNOLOGY
NIRMA UNIVERSITY OF SCIENCE & TECHNOLOGY,
AHMEDABAD-382481
MAY-2008

ACKNOWLEDGEMENTS

I am indebted to **Mr. Vivek sharma**, Director, STMicroelectronics India, for giving me an opportunity to work in such a reputed organization.

I am grateful to **Mr. R. Varambally, DU Manager, TECH.R&D** and **Mrs. Namerita Khanna, Section Manager, eCL Group** for extending their support and providing us with requisite infrastructure for training at ST Microelectronics.

I pay my special regards to **Mr. Ajay kumar Gautam** for giving me an opportunity to work on this project and for their esteemed guidance, support and encouragement to achieve the aim, throughout the project.

I am thankful to **Mr. kanwar Rajan singh, Mr. Jagmohan Singh**, **Mr.Himanshu Srivastva, Mrs. Jyoti Kumar** and others in the group for their able guidance and timely help, their critical observations and encouragement that made successful completion of this work possible.

I am grateful to **Mr. Rajeshware Arora**, document control administrator for his Co-operation.

I would like to specially thank **Dr. N.M.Devashrayee**, PG Co-ordinator, M.Tech. VLSI Design, Nirma University, Ahmeadabad for providing me with an opportunity to take up this training and for their constant support and encouragement. I would also like to thank **Prof.N.P.Gajjar**, Nirma University, Ahmedabad to provide guidance for this project. After undergoing this project training, I can confidently say that this experience has not only enriched me with technical knowledge, but also with the deep insight into quality concepts that are required to be a successful professional.

Hirani Chandresh
Roll No: 06MEC006
M.Tech. VLSI Design,
Nirma University,
Ahemedabad.

CONTENTS

1. ABSTRACT 1

2. COMPANY PROFILE 2
   2.1 ABOUT THE COMPANY 3
   2.2 PRODUCTS 6
   2.3 APPLICATIONS 6

3. INTRODUCTION 7
   3.1 PROGRAMMABLE LOGIC DEVICES 7
   3.2 ADVANTAGES OF PROGRAMMABLE LOGIC OVER THE FIXED LOGIC 8

4. CADENCE VIRTUOSO 9
   4.1 VIRTUOSO LAYOUT EDITOR 9
   4.2 BENEFITS 9

5. IMPLEMENTATION OF 65 nm LIBRARY 11
   5.1 MODIFICATION IN DIFFERENT LAYOUTS 11
   5.2 NEW MADE LAYOUTS BASED ON PREVIOUS LAYOUTS 12
   5.3 TOTALLY INDEPENDENT MADE LAYOUTS 13

6. MODIFICATION IN LAYOUT 18
   6.1 PARALLEL RUN PROBLEM 18
   6.2 ACCESSIBILITY PROBLEM 19
   6.3 PIN ALIGNMENT PROBLEM 20
   6.4 METAL-2 PROBLEM 21

7. EXTRACTION OF THE LAYOUT 22

8. SYNTHESIS 24
   8.1 WHAT IS LOGIC SYNTHESIS? 24
   8.2 LEVELS OF ABSTRACTION 25
   8.3 WHY SYNTHESIS TOOL IS REQUIRED? 26
   8.4 SYNTHESIS IS CONSTRAINT-DRIVEN 27

9. PLACE AND ROUTE 28
   9.1 LOAD LIBRARY 29
   9.2 IMPORT GATE NETLIST 32
   9.3 SPECIFY DESIGN CONSTRAINT 35
   9.4 FLOOR PLANNING 36
   9.5 POWER PLANNING 40
   9.6 PHYSICAL SYNTHESIS 41
   9.7 CLOCK TREE SYNTHESIS 43
   9.8 ROUTING 45
   9.9 PHYSICAL VERIFICATION 47
   9.10 POST-LAYOUT VERIFICATION 48

10. AES (ADVANCED ENCRYPTION STANDARD) 50
10.1 NOTATION AND CONVENTIONS 50
10.2 ALGORITHM SPECIFICATION 54
10.3 CIPHER 55
10.4 KEY EXPANSION 60
10.5 IMPLEMENTATION ISSUES 61

11. SYNTHESIS RESULTS OF AES 62
11.1 SYNTHESIS INPUTS 62
11.2 SYNTHESIS OUTPUT 63

12. PNR RESULTS OF AES 69
12.1 INPUTS TO PNR TOOL 69
12.2 OUTPUT OF PNR TOOL 70
12.3 OUTPUT DESIGN IMAGES: 71

13. CONCLUSION 76

14. REFERENCES 77

LIST OF FIGURES

Fig. 4.1 Virtuoso View....10
Fig. 4.2 Layout of NOR Gate....10
Fig. 5.1 Configuration of EX-OR and EX-NOR gates using tri-state topology....15
Fig. 5.2 Configuration Multiplexer using tri-state topology....16
Fig. 6.1 Parallel Run Problem....18
Fig. 6.2 Pin Alignment Problem....19
Fig. 6.3 Pin Alignment Problem....20
Fig. 6.4 Metal-2 Problem....21
Fig. 7.1 Extraction flow for any layout....23
Fig. 9.1 An example of a standard cell abstract.....31
Fig. 9.2 Cell and instance....33
Fig. 9.3 Uniquify the netlist.....34
Fig 9.4 Definition of the core area.....37
Fig. 9.5 Three types of cell row configuration.....38
Fig. 9.6 A floor plan with one macro and I/O pads....39
Fig. 9.7 Steps in building a power plan.....41
Fig. 9.8 Clock tree schematics example....44
Fig. 9.9 Insertion of the filler cells....45
Fig. 9.10 Physical verification flow.....48
Fig. 9.11 Post-layout verification flow....49
Fig. 10.1 State array input and output.....53
Fig. 10.2 Key-Block-Round Combinations....54
Fig. 10.3 Flow of AES Algorithm....55
Fig 10.4 SubBytes() applies theS-Box to each byte of the state.....56
Fig.10.5 S-box: substitution values for the byte XY (in hexadecimal format)....57
Fig. 10.6 ShiftRows() cyclically shifts the last three rows in the State.....58
Fig 10.7 MixColumns() operates on the State column-by-column....59
Fig. 10.8 AddRoundKey() XORs each column of the State with a word from the key schedule....60
Fig 11.1 Comparison of Individual Cell Area....66
Fig 11.2 Comparison of Individual Cell Count Used....67
Fig 11.3 Comparison of Total occupied by Individual Cell in Design....68
Fig 11.4 Comparison of Normalized Cell Count and Corresponding occupied area .68
Fig 12.1 Placed Design Floorplan View....72
Fig 12.2 Placed Design With Routing Nets....73
Fig 12.3 Zoomed Placed Design View....73
Fig 12.4 Netlist View after adding Filler....74
Fig 12.5 Output Optimized Net....75

LIST OF TABLES

Table 2.1 Company Profile 2
Table 5.1 Modification in different layouts 11
Table 5.2 Metal Requirement For New Made Layouts Based On Previous Layouts. 12
Table 5.3 No. of transistor pairs requires to drive given load in Inverter. 13
Table 5.4 No. of transistor pair required to drive given load in buffer 14
Table 5.5 No. of Metal-2 and Metal-3 used in new independent made layouts 17
Table 9.1 Evolution of placement methodology with process technology. 42
Table 10.1 Hexadecimal representation of bit patterns. 51
Table 10.2 Indices for Bytes and Bits. 52
Table 11.1 Library Information 62
Table 11.2 Timing Constraints 63
Table 11.3 Timing Report 64
Table 11.4 Total Area Occupied by design 64
Table 11.5 Individual Cell Report 65
Table 11.6 Attribute Meaning used in Synthesis Report 66

ABSTRACT

A plurality of Electronically Reconfigurable Gate Array (ERCGA) logic circuits are interconnected via a reconfigurable interconnect, and electronic representations of large digital networks are converted to take temporary actual operating hardware form on the interconnected circuits. The reconfigurable interconnect permits the digital network realized on the interconnected circuits to be changed at will, making the system well suited for a variety of purposes including simulation, prototyping, execution and computing. The reconfigurable interconnect may comprise a partial crossbar that is formed of ERCGA circuits dedicated to interconnection functions, wherein each such interconnect ERCGA is connected to at least one, but not all of the pins of a plurality of the logic circuits. In STMicroelectronics we divide the programmable logic in two part software and hardware. Software part includes development of software for the use of PiCoGA architecture and hardware part includes development of different libraries in 90nm and 65 nm for the realization of that architecture. Work carried out in this group was to prepare complete library set which includes all basic functionality in 65nm technology using Structured ASIC.

2. COMPANY PROFILE

COMPANY	ST MICROELECTRONICS Pvt Ltd
Head Office website	http:/www.st.com
Contact No.	91 120-2352999
Incorporated in	1987
CEO	Carlo Bozotti
Address	STMicroelectronics Private Ltd. GREATER NOIDA Plot No.1, Knowledge Park III 201308, GREATER NOIDA, India
Branches in India	Bangalore, Greater Noida
No. of Employees	Approx 50,000.

Table 2.1 Company Profile

2.1 ABOUT THE COMPANY

STMicroelectronics is one of the world's largest semiconductor companies with net revenues of US$9.85 billion in 2006 and US$4.69 billion for the first half of 2007.

The Company's sales are well balanced between the semiconductor industry's five major high-growth sectors (percentage of ST's sales in 2007): Communications (35%), Consumer (17%), Computer (16%), Automotive (16%) and Industrial (16%).

According to the latest industry data, ST is the world's fifth largest semiconductor company with market leadership in many fields. For example, ST is the leading producer of application-specific analog chips and power conversion devices. It is also the #1 supplier of semiconductors for the Industrial market and for set-top box applications, and occupies leading positions in fields as varied as discrete devices, camera modules for mobile phones and automotive integrated circuits.

Product Portfolio

ST aims to be the leader in multimedia convergence applications and power solutions, offering one of the world's broadest product portfolios, including application-specific products containing a large proprietary IP content and multi-segment products that range from discrete devices to high-performance microcontrollers, secure smart card chips and MEMS (Micro-Electro-Mechanical Systems) devices.

For complex ICs in demanding applications such as mobile multimedia, set-top boxes and computer peripherals. The balanced portfolio approach allows ST to address the needs of all microelectronics users, from global strategic customers for whom ST is the partner of choice for major System-on-Chip (SoC) projects to local enterprises that need fully-supported general-purpose devices and solutions.

ST has also announced its intention, together with Intel and Francisco Partners, to form a new, independent semiconductor company, Numonyx, which will focus on supplying non-volatile memory solutions for a variety of consumer and industrial devices.

Research & Development and Manufacturing

Since its creation, ST has exhibited an unwavering commitment to R&D and is one of the industry's most innovative companies. ST's process technology portfolio includes advanced CMOS logic (including embedded memory variants), mixed-signal, analog and power processes. In advanced CMOS, ST is to partner with the IBM consortium for the development of next-generation process technologies, including 32nm and 22nm CMOS process development, design enablement and advanced research adapted to the manufacturing of 300mm silicon wafers. ST and IBM will also cooperate at ST's Crolles 300mm facility in the development of value-added CMOS derivative SoC technologies.

ST has a worldwide network of front-end (wafer fabrication) and back-end (assembly, packaging and test) plants. The Company is moving towards a less capital-intensive manufacturing strategy and has recently announced plans to phase out some of its older facilities. ST's principal wafer fabs are presently located in Agrate Brianza and Catania (Italy), Crolles, Rousset and Tours (France), Phoenix and Carrollton (USA) and Singapore. The wafer fabs are complemented by highly efficient assembly and test facilities located in China, Malaysia, Malta, Morocco and Singapore.

Alliances

ST has developed a worldwide network of strategic alliances, including product development with key customers, technology development with customers and other semiconductor manufacturers, and equipment- and CAD-development alliances with major suppliers. These industrial partnerships are complemented by a wide range of research programs conducted with leading universities and research institutes around the world, in addition to playing a key role in Europe's advanced technology research programs such as MEDEA+ and industry initiatives such as ENIAC (European Nano-electronics Initiative Advisory Council).

Sustainable Excellence

STMicroelectronics was one of the first global industrial companies to recognize the importance of environmental responsibility, its initial efforts beginning in the early 1990s. Since then ST has made outstanding progress; for example, energy consumption per product unit was reduced by 47% between 1994 and 2006 and CO2 emissions have been reduced by 61% over the same timescale. In addition, ST has gone far beyond existing legal requirements in almost completely eliminating the use of hazardous substances such as lead, cadmium, and mercury. Since 1991, the Company's sites have received more than 100 awards for excellence in all areas of Corporate Responsibility, from quality to corporate governance, social issues and environmental protection.

Facts and Figures

STMicroelectronics was created in 1987 by the merger of SGS Microelettronica of Italy and Thomson Semiconducteurs of France. Since its formation, ST has grown faster than the semiconductor industry as a whole and it has been one of the world's Top Ten semiconductor suppliers since 1999.

The group totals approximately 50,000 employees, 16 advanced research and development units, 39 design and application centers, 15 main manufacturing sites and 78 sales offices in 36 countries.

Corporate Headquarters, as well as the headquarters for Europe and for Emerging Markets, are in Geneva. The Company's U.S. Headquarters are in Carrollton (Texas); those for Asia-Pacific are based in Singapore and Japanese operations are headquartered in Tokyo. The "Greater China" region, which includes Hong Kong, China and Taiwan, is headquartered in Shanghai.

Since December 8, 1994, when ST completed its initial public offering, the Company's shares have been traded on the New York Stock Exchange (NYSE: STM) and on Euronext Paris; since June 1998, ST has also been listed in Milan on Borsa Italiana. The Company now has around 900 million outstanding shares, 71.1% of which are publicly traded on the various stock exchanges. The balance of the shares is

held by STMicroelectronics Holding II B.V. (27.5%), a company whose shareholders are Cassa Depositi e Prestiti and Finmeccanica of Italy, and Areva of France, and treasury shares (1.4%) held by STMicroelectronics NV.

2.2 PRODUCTS

- Analog & Mixed Signal ICs
- Memories
- Power Management:
- Transistors
- Application Specific for Audio power
- Product Technologies

2.3 APPLICATIONS

- Automotive
- Industrial
- Computer
- Security & Smartcard
- Industrial

3. INTRODUCTION

In the world of digital electronic systems, there are three basic kinds of devices: memory, microprocessors, and logic. Memory devices store random information such as the contents of a spreadsheet or database. Microprocessors execute software instructions to perform a wide variety of tasks such as running a word processing program or video game. Logic devices proide specific functions, including device-to-device interfacing, data communication, signal processing, data display, timing and control operations, and almost every other function a system must perform.

3.1 PROGRAMMABLE LOGIC DEVICES

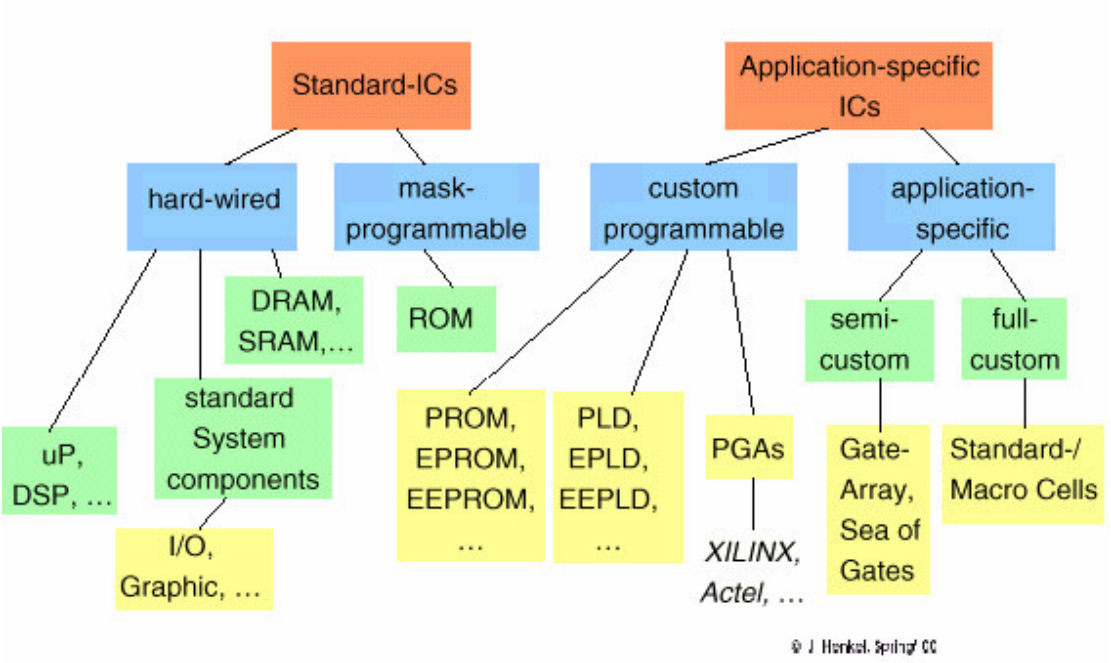


Fig. 3.1 Logic Device Family

The main differences in programmable devices are between:

- Mask-programmable and field-programmable
- Erasable and non-erasable

The mask-programmable types are programmed when they are manufactured whereas the user sets up the field-programmable device with some form of programmer. Mask-programmable devices are expensive in low production runs but are relatively cheap for large production runs, which is opposite for the field-programmable devices. An erasable device allows the stored set-up to be changed whereas the non-erasable type is permanent.

3.2 ADVANTAGES OF PROGRAMMABLE LOGIC OVER THE FIXED LOGIC

1. PLDs offer customers much more flexibility during the design cycle because design iterations are simply a matter of changing the programming file, and the results of design changes can be seen immediately in working parts.
2. PLDs do not require long lead times for prototypes or production parts - the PLDs are already on a distributor's shelf and ready for shipment.
3. PLDs do not require customers to pay for large NRE costs and purchase expensive mask sets - PLD suppliers incur those costs when they design their programmable devices and are able to amortize those costs over the multi-year lifespan of a given line of PLDs.
4. PLDs allow customers to order just the number of parts they need, when they need them, allowing them to control inventory. Customers who use fixed logic devices often end up with excess inventory which must be scrapped, or if demand for their product surges, they may be caught short of parts and face production delays.
5. PLDs can be reprogrammed even after a piece of equipment is shipped to a customer. In fact, thanks to programmable logic devices, a number of equipment manufacturers now tout the ability to add new features or upgrade products that already are in the field. To do this, they simply upload a new programming file to the PLD, via the Internet, creating new hardware logic in the system.

4. CADENCE VIRTUOSO

Virtuoso® Layout Editor is the industry-standard base-level custom physical layout tool of the Virtuoso custom design platform. It supports the physical implementation of custom digital, mixed-signal, and analog designs at the device, cell, and block levels.

The Virtuoso custom design platform is a comprehensive system for fast, silicon-accurate design and is optimized to support "meet-in-the-middle" design methodologies such as advanced custom design. Virtuoso includes the industry's only specification-driven environment, multi-mode simulation with common models and equations, vastly accelerated layout, advanced silicon analysis for 0.13 microns and below, and a full-chip, mixed-signal integration environment. The Virtuoso platform is available on the Cadence® CDBA database and the industry standard Open Access database. With the Virtuoso platform, design teams can quickly design silicon that is right and on time at process geometries from one micron to 90 nanometers and beyond.

4.1 VIRTUOSO LAYOUT EDITOR

With Virtuoso Layout Editor, custom layout is accelerated with a comprehensive set of user-configurable and easy-to-use pure polygon layout features within a hierarchical multi-window environment. Additional acceleration is provided through optional parameterized cells (Pcells) and a powerful scripting language called SKILL that provides direct database access, tool configuration, and interoperability with other tools.

4.2 BENEFITS

- Easy creation and navigation of complex designs with unlimited hierarchy support coupled with a multi-window editing environment (see Figure 4.1)
- Accelerated layout entry using easy-to use and easily accessed editing functions
- Increased productivity and design optimization using Pcells
- Efficient, high-performance handling of large designs using the Open Access database

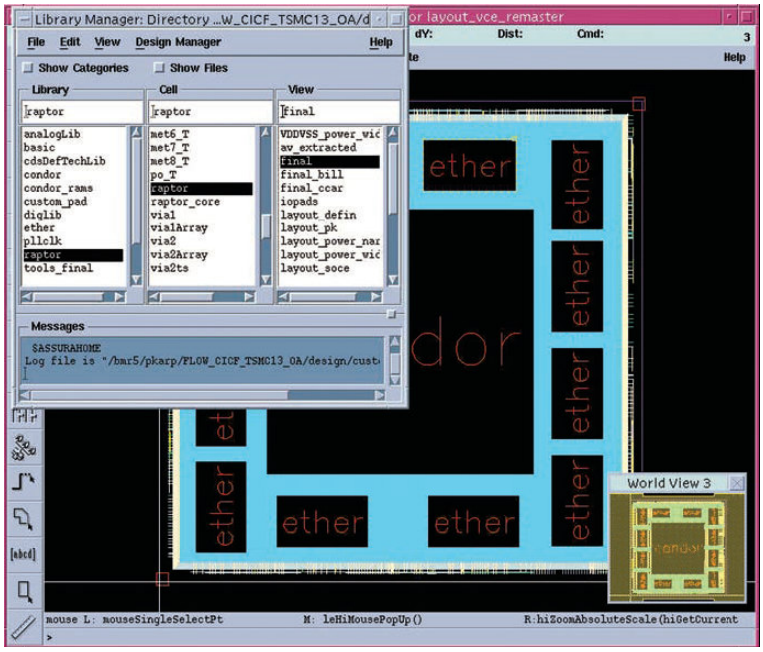


Fig. 4.1 Virtuoso View

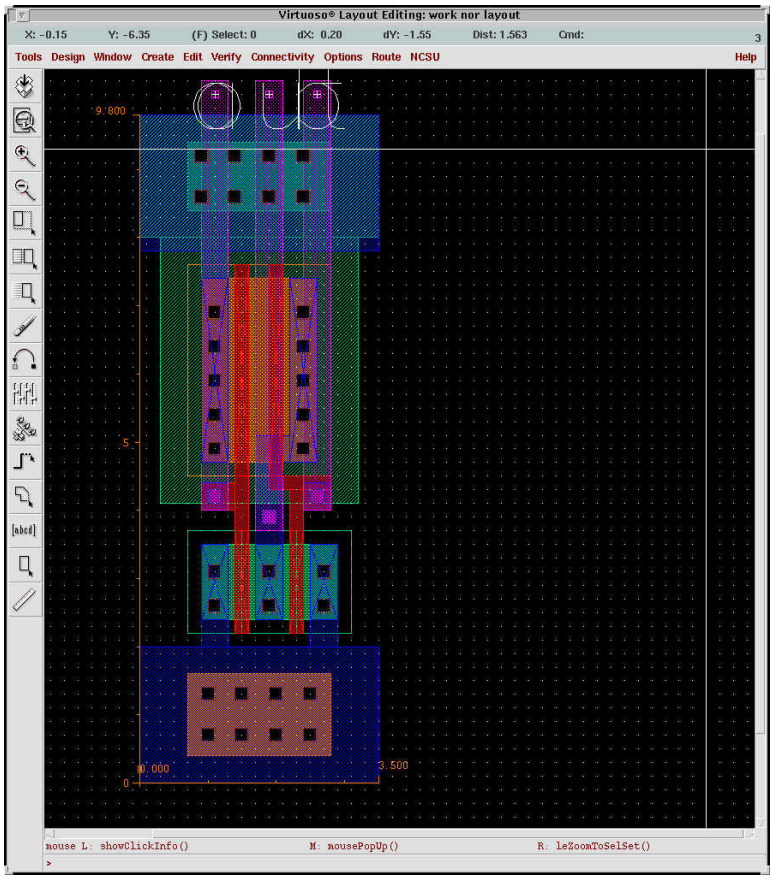


Fig. 4.2 Layout of NOR Gate

5. IMPLEMENTATION OF 65 nm LIBRARY

In cadence layout maker I am working with 65nm technology. Here I have allotted the library of INVERTER, NAND gate, NOR gate, XOR gate, MUX and some small function like A+BC with given load driven capability.

This library has one conman base cell on which we implement our logic. This base cell have some constrain like,

1. The metal-3 must pass on the gird line and horizontally.
2. The spacing between the different metal and poly.
3. The PRBoundary setting must be on (0, 0) position etc.

5.1 MODIFICATION IN DIFFERENT LAYOUTS

Then I started to design a new library of the same component and try to optimize routing by reducing number of metal-2 and metal-3 required for connection in the cell. By this routing number of metal-2 and number of metal-3 saved is given in table 5.1.

Layout name		Original				After Editing			
		No. of M2 track	No. of M2 track on path	No. of M3 track	No. of M3 track on path	No. of M2 track	No. of M2 track on path	No. of M3 track	No. of M3 track on path
AO6 X18		0	-	0	-	0	-	0	-
AO7 X18		1	0	0	-	1	0	0	-
AO12X27		1	0	0	-	0	-	0	-
AO12X35		1	0	0	-	0	-	0	-
MUX X18		3	2	1	1	3/2	3	1	1
NAND	X18	0	-	0	-	0	-	0	-
	X36	3	3	0	-	0	-	0	-
	X54	3	3	0	-	0	-	0	-
NOR	X18	0	-	0	-	0	-	0	-
	X36	3	3	0	-	0	-	0	-
	X54	3	3	0	-	0	-	0	-
XOR X18		3	2	1	0	3/2	3	1	1

Table 5.1 Modification in different layouts

5.2 NEW MADE LAYOUTS BASED ON PREVIOUS LAYOUTS

There is another task to made new layouts based on previously modified layouts. The available layouts are of lower load capacity. Now my task is make new cells for other load capacity using these available cells. The new made layouts and required metal-2 and metal-3 is given in table 5.2.

Layout name		No. of M2 track	No. of M2 track on preferred path	No. of M3 track	No. of M3 track on preferred path
AOI12	X9	0	0	0	0
	X36	4	4	0	0
	X54	4	4	0	0
OAI12	X36	4	3	0	0
	X54	4	3	0	0
MUX	X9	1	1	0	0
	X36	4	4	2	2
	X54	4	4	3	3
NAND	X9	0	0	0	0
	X36	0	0	0	0
	X54	0	0	0	0
NOR	X9	0	0	0	0
	X36	0	0	0	0
	X54	0	0	0	0
XOR	X9	1	1	0	0
	X36	3	3	2	2
	X54	3	3	3	3

Table 5.2 Metal Requirement For New Made Layouts Based On Previous Layouts

5.3 TOTALLY INDEPENDENT MADE LAYOUTS

There are some requirements where this layouts are not giving satisfactory performance as well as some functionality are not design for this library. So I have to design it independently. Some such design is listed below which is further described next.

- Inverter
- Buffer
- AO-functionality
- OA-functionality
- NAND
- NOR
- XOR using tri-state topology
- XOR using Boolean Equation
- XNOR using tri-state topology
- XNOR using Boolean Equation
- MUX using tri-state topology

5.3.1 Design Of Inverter:

Inverter consists of transistor pairs in parallel connection. No. of transistor pair is depends on output load required to drive. Table 5.3 indicates the designed inverters and no. of transistor pair required to drive given load.

Inverter Load Capacity	No. of Transistor pair in driver
X 9	1
X 18	2
X 27	3
X 36	4
X 45	5
. .	. .
X 216	24

Table 5.3 No. of transistor pairs requires to drive given load in Inverter.

5.3.2 Design of Buffer:

Buffer consists of two connected inverters as described below. It can be design in various ways depends on output load condition. No. of transistor pair in stage B is depend on output load. For example if no. of transistor pair in stage B is 4 than it can drive double load than the buffer which includes 2 transistor pair in stage B. Here no. of transistor pair in stage A is depends on no. of transistor pair in stage B. This no. should be sufficient that can drive no. of transistor pair in stage B.

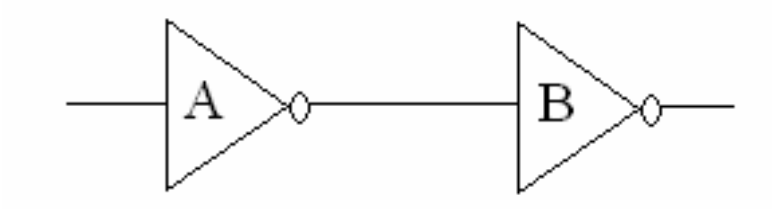


Table 5.4 indicates no. of transistor pair in stage A and B for different load capacity.

Buffer Load Capacity	No. of Transistor pair in A stage	No. of Transistor pair in B stage
X 9	1	1
X 18	1	2
X 27	1	3
X 35	2	4
X 45	2	5
X 54	2	6
. . .	. . .	. . .
X 216	8	24

Table 5.4 No. of transistor pair required to drive given load in buffer

5.3.3 Design of EX-OR and EX-NOR using tri-state topology

Functionality of **EX-OR** is given by below equation **Z = A'B + AB'** whereas functionality of EX**-NOR** is given by equation Z **= AB + A'B'**. These functionalities can be implement by two connected tri-state as shown in figure 5.1.

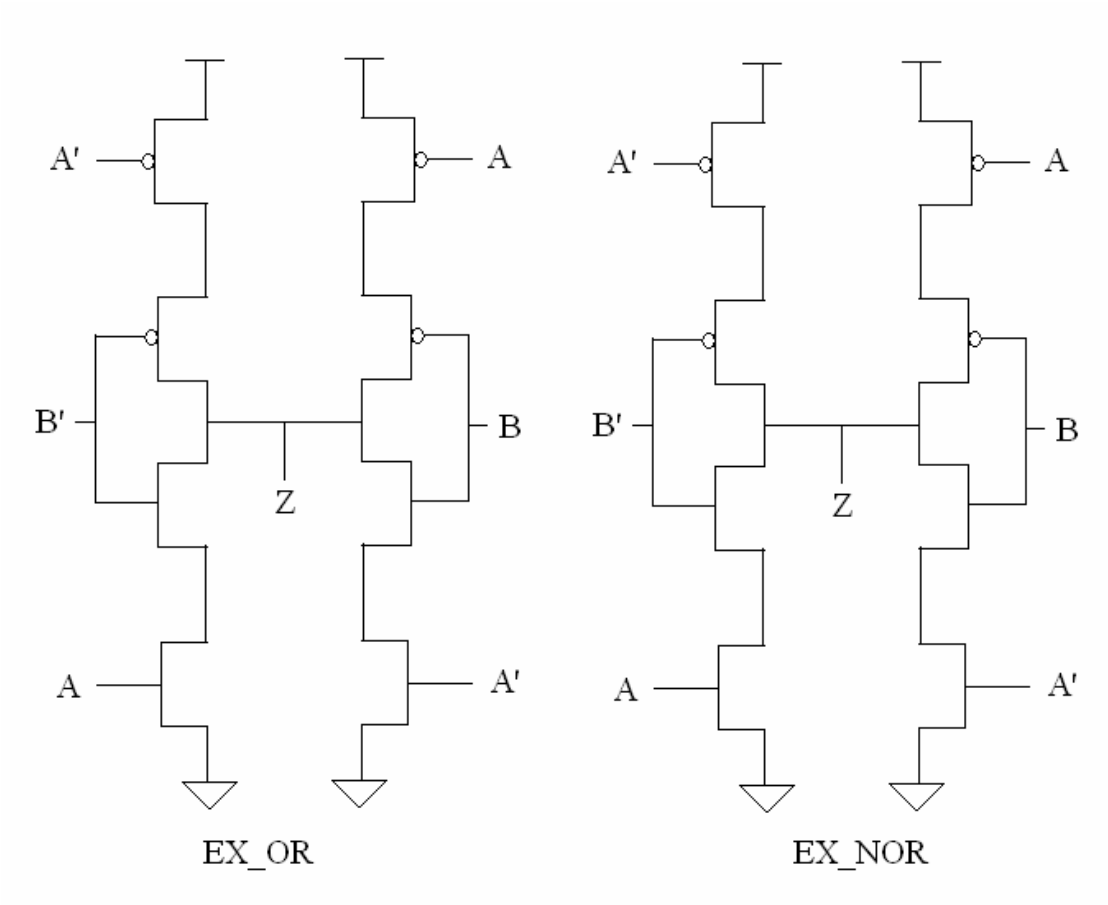


Fig. 5.1 Configuration of EX-OR and EX-NOR gates using tri-state topology

In EX-OR signal A drives your another signal B or B', and output depends on A and B. Here this configuration provides low load capacity. So for improving load capacity we can insert buffer stage as per our requirement. In EX-NOR also we can implement in the same way as XOR functionality.

5.3.4 MULTIPLEXER using tri-state topology

Functionality of Multiplexer is given by below equation

Z = D0 S0' + D1 S0

This can be also designed by tri-state topology as given figure 5.2.

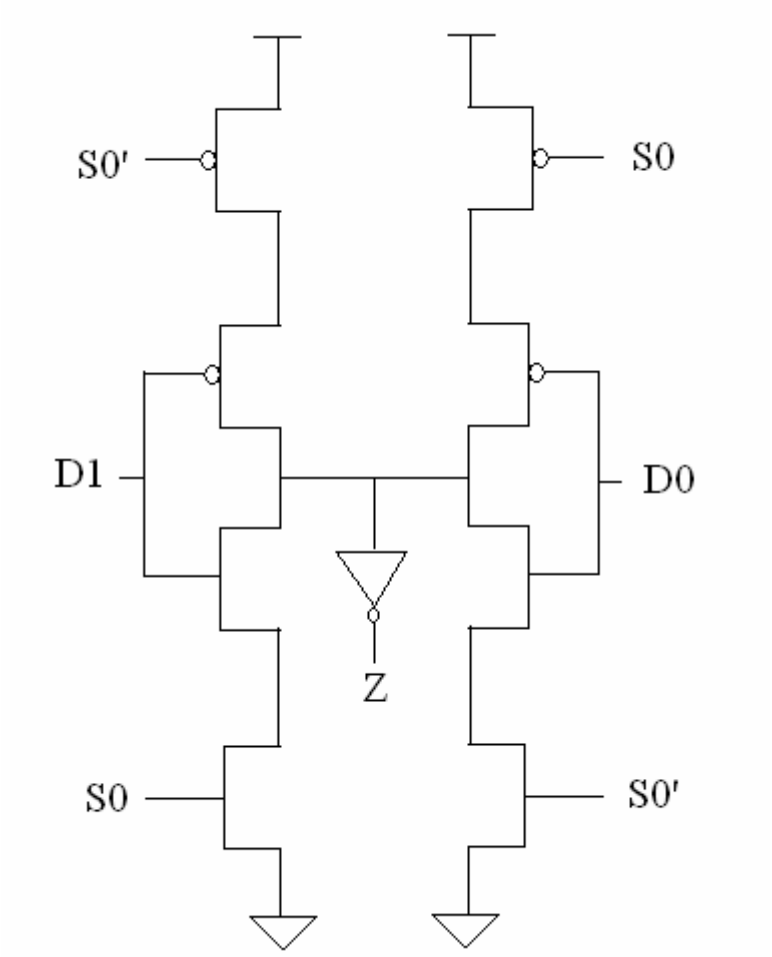


Fig. 5.2 Configuration Multiplexer using tri-state topology

This functionality can be also explained in the same way as exclusive gates. Here load capacity of multiplexer is depends on design of inverter.

5.3.5 Requirement of Metal-2 and Metal-3 of independent made layouts

Requirement of Metal-2 and Metal-3 in independent layouts are given in table 5.5.

Layout name	No. of M2 track	No. of M2 track on preferred path	No. of M3 track	No. of M3 track on preferred path
Buffer	0	-	0	-
Inverter	0	-	0	-
XOR X9	1	1	0	-
XNOR X9	1	1	0	-
MUX X18, X36, X54	0	-	0	-
AOI22	1	1	0	-
AO112	1	1	0	-
NAND X9	0	-	0	-
NOR X9	0	-	0	-

Table 5.5 No. of Metal-2 and Metal-3 used in new independent made layouts

6. MODIFICATION IN LAYOUT

After making complete library set, it will goes to used in synthesis and place& Route. At this time it will gives some problem which is not issue in case of individual layout. So it is required to modify these layouts to solve problems which might be generated after Place and Route. Such type of some problems and their solutions are discussed in below sections.

6.1 PARALLEL RUN PROBLEM

As shown in figure 6.1 when any metal line with its width less than 'Y' is passing near boundary, with distance less than 'X/2' from PRBoundary, this type of problem occurs. Here individual layout doesn't give any DRC Errors. But when this type of cells placed near each other in PNR, it will gives DRC errors, because it requires distance between two metals larger than 'X' when their individual width is less than 'Y'.

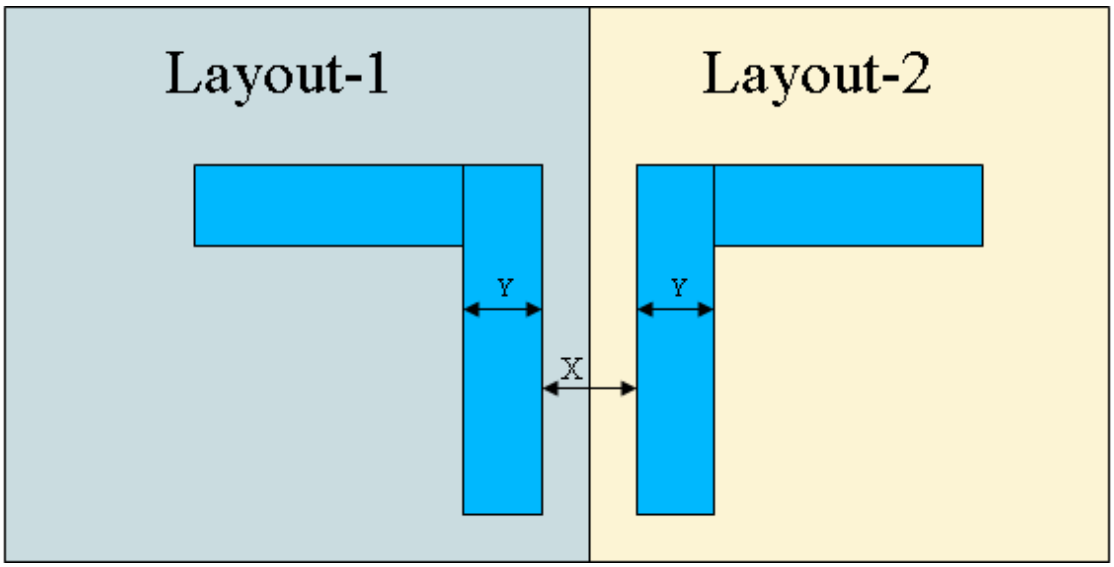


Fig. 6.1 Parallel Run Problem

This problem can be solved by two ways.

- Modify these layouts in such a way that each metal running parallel to the PRBoundary than its width is larger or equal to ‘Y’.
- Modify these layouts in such a way that distance between this type of metals is always larger or equal to ‘X’.

  In our library we solve this problem using first solution.

6.2 ACCESSIBILITY PROBLEM

At the time of PNR it will connect two cells using metal layers using higher than metal-1. So if your pin is of metal-1 than first it will place via and than connect metal-1 to metal-2. And this metal-2 is used in other connection.

So if you are already using metal-2 in your cells and it is passing above metal-1 pin as shown in fig 6.2 than it is not possible to connect that pin to any other cell at the time of PNR.

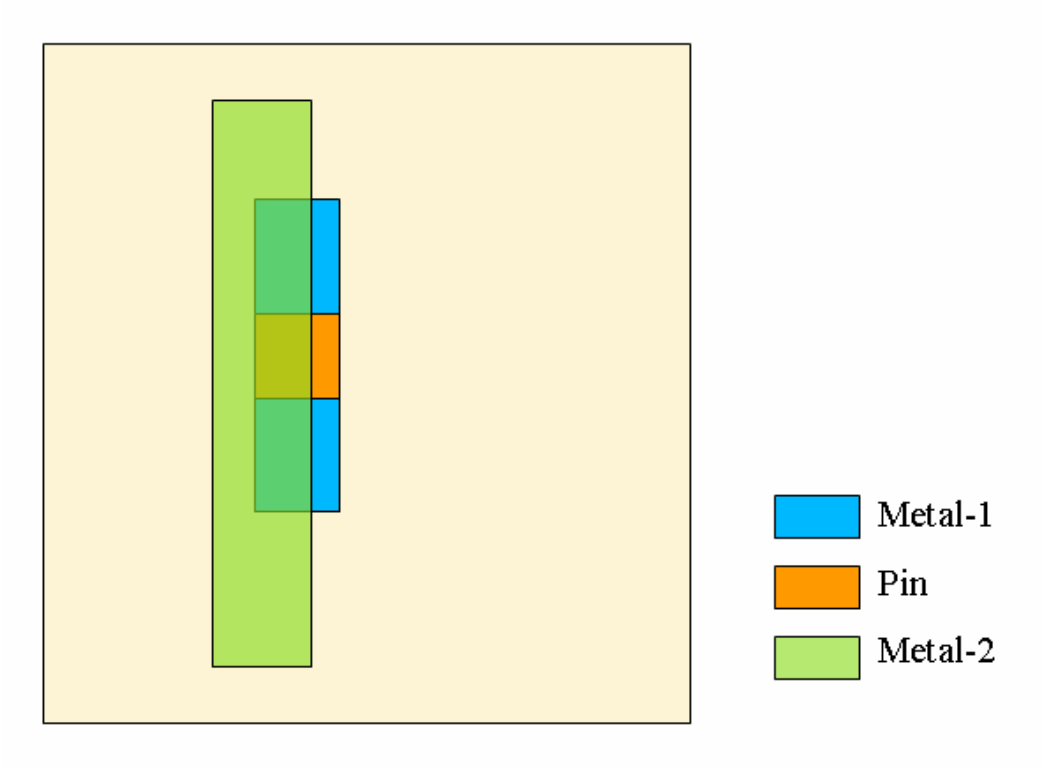


Fig. 6.2 Pin Alignment Problem

This problem can be solved using two methods.

- Shift this metal two in such a way that it is not passing above any metal-1 contains pin.
- Change in layout in such a way that metal-1 pin shift its position where it is not lying below metal-2.

6.3 PIN ALIGNMENT PROBLEM

At the time of PNR, tool will search pins in grid position and connects it with metal-2. So if your pin is not on grid than tool is not able to find pin position and not able to connect it with other cells. This situation is described in fig 6.3.

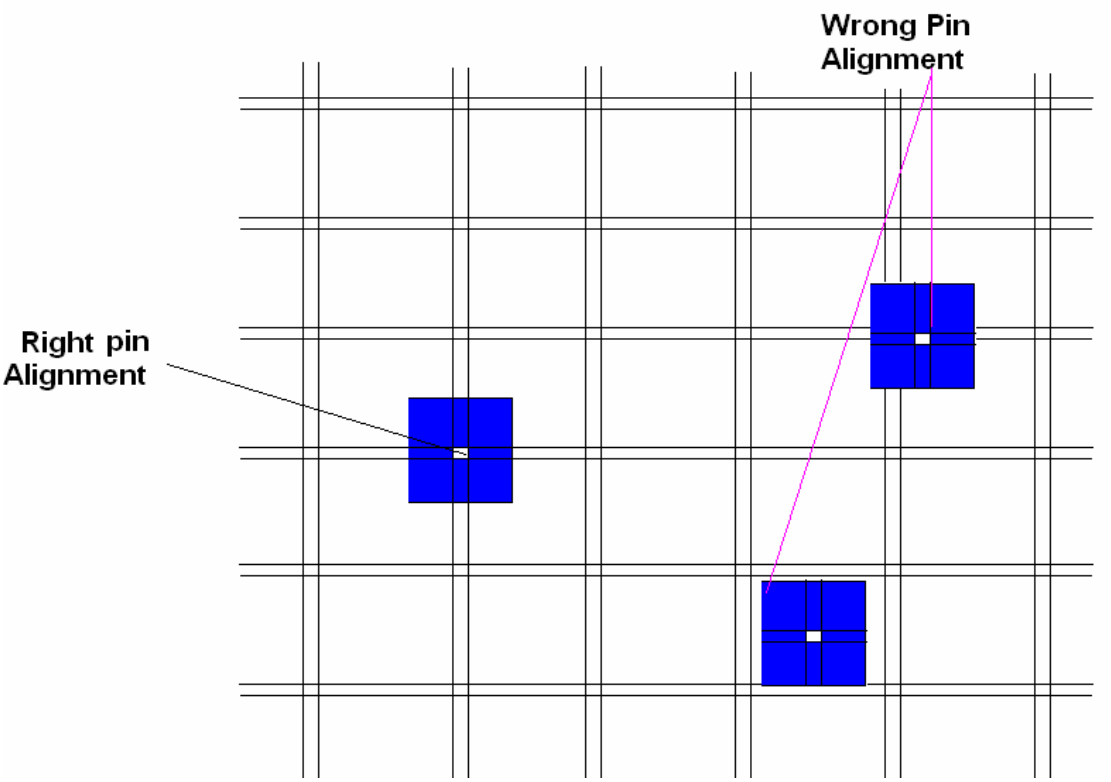


Fig. 6.3 Pin Alignment Problem

To solve this problem pin should be shifted in the centre of grid along with metal-1 so that at the time of PNR this pins are identified and can be connected with other cells.

6.4 METAL-2 PROBLEM

In many cases metal-2 is used for power connection. So if your layout contains Metal-2 and passing near power line than it might give DRC problems if proper distance is not maintained. This condition is described in fig 6.4.

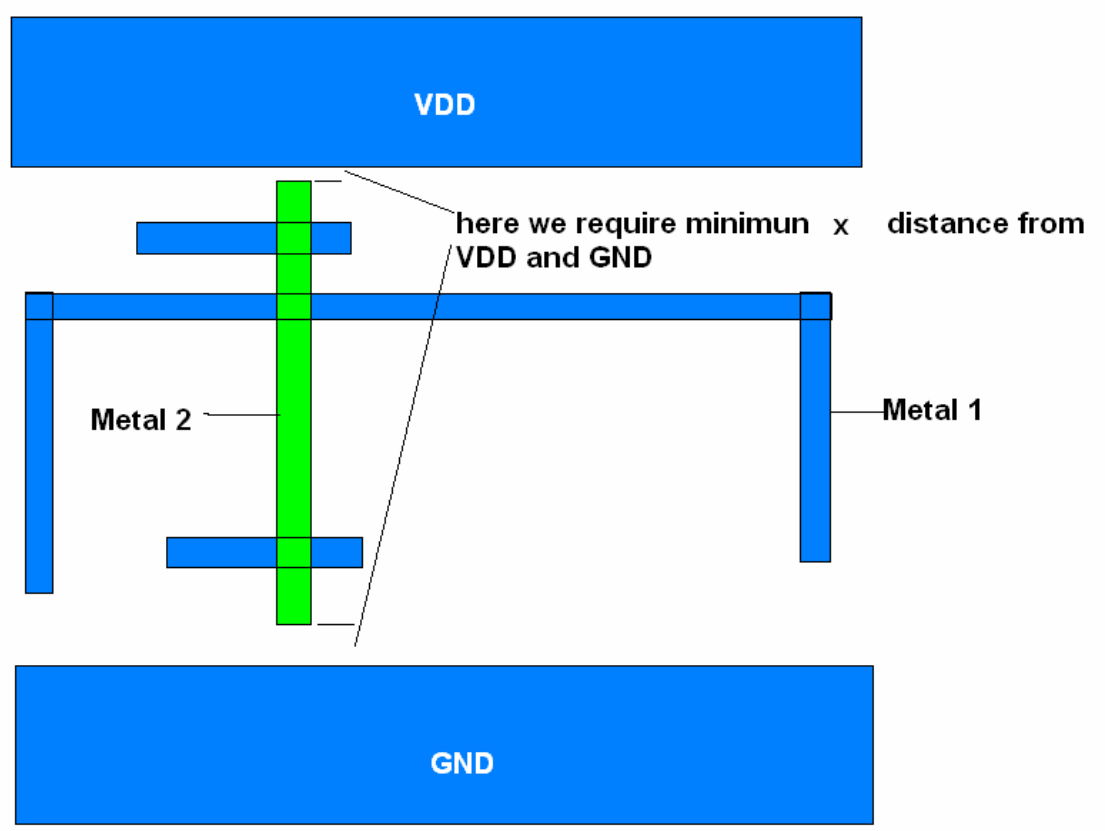


Fig. 6.4 Metal-2 Problem

So to solve this problem, you have to modify these layouts in such a way that metal-2 remains at larger or equal distance from power line.

7. EXTRACTION OF THE LAYOUT

After completion of layouts its extraction is required for generation of .spi file. This file contains required information for post-layout simulation. It contains netlist of layout including resistance of each nets as well as capacitance between these nets. So by using this information we can find out maximum operating frequency, its power consumption and many more things. So extraction of each layout is necessary.

The flow of the extraction is as shown in fig 7.1.

As shown in the figure 7.1 first we generate the .GDS (Graphical Design System) file from the layout which contains information of layout. It includes information in terms of dimension of rectangles used in layout and its relative position.

Then we generate the **.**CDL (Circuit Description Language) file from the schematic which contains the circuit connection details. It is netlist but contains information only related to connection. It not includes any information of resistance and capacitance used in it.

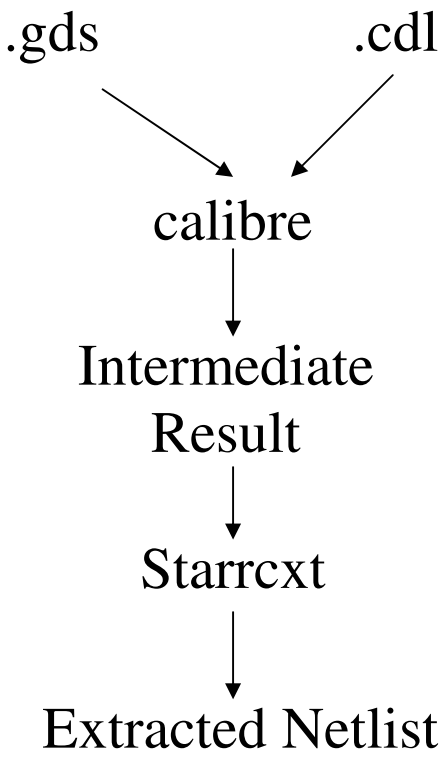


Fig. 7.1 Extraction flow for any layout

This two files, .GDS and .CDL can not accept directly by Starrcxt tool. So it is required to convert this files into some another format. So these two files are given to the tool caliber and it generates the intermediate result that can accept by Starrcxt tool.

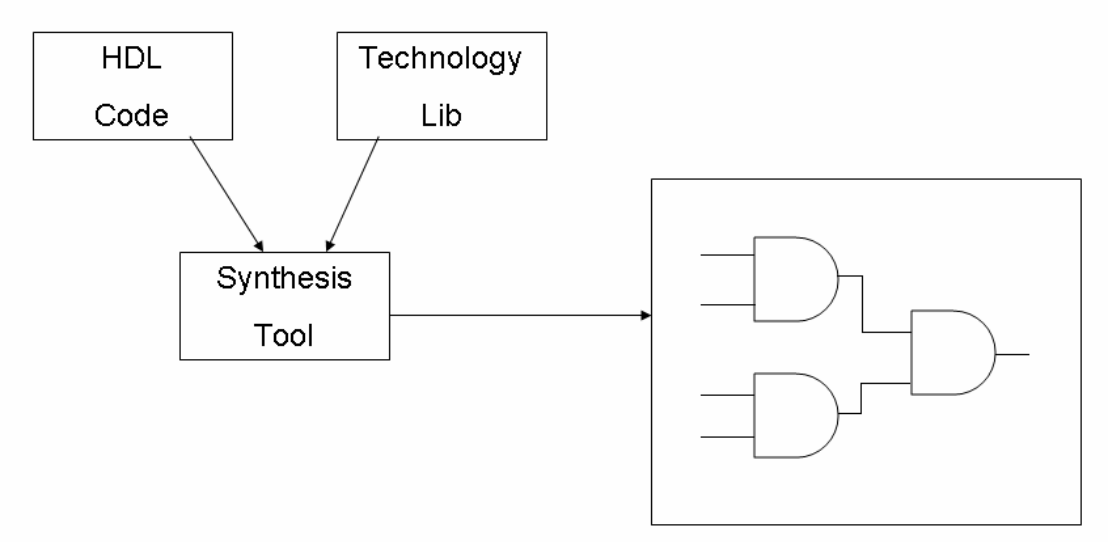
This tool compares both files and then generates LVS report as well as netlist including resistance and capacitance between different nets. This netlist is our required .spi files.

8. SYNTHESIS

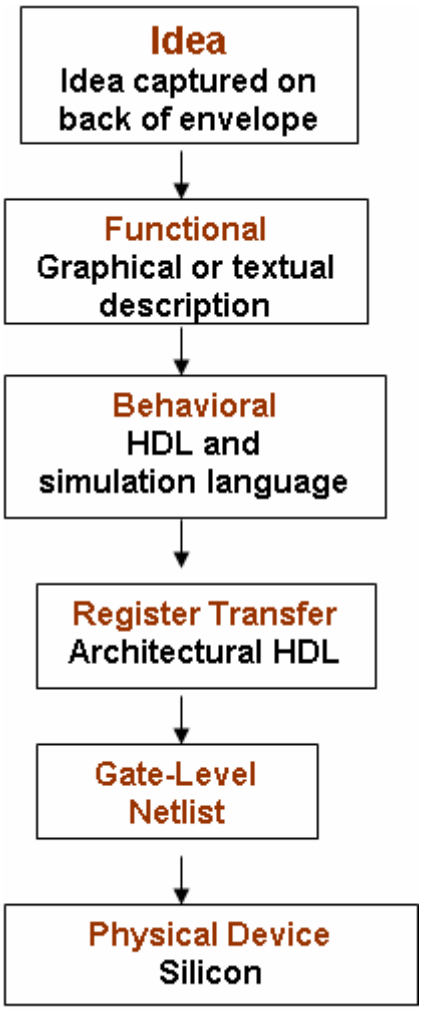
8.1 WHAT IS LOGIC SYNTHESIS?

Logic synthesis is the process of converting a high-level description of design into an optimized gate-level representation. Logic synthesis uses a standard cell library which have simple cells, such as basic logic gates like and, or, and nor, or macro cells, such as adder, muxes, memory, and flip-flops. Standard cells put together are called technology library. Normally the technology library is known by the transistor size (0.18u, 90nm).

A circuit description is written in Hardware Description Language (HDL) such as Verilog. The designer should first understand the architectural description. Then he should consider design constraints such as timing, area, testability, and power.



8.2 LEVELS OF ABSTRACTION



So synthesis may be divide into two three parts.

- Translation
- Optimization
- Mapping

First it will translate your HDL code into Boolean function. So that it can be implement as hardware. Than this Boolean function will minimize using different algorithm, which is said Optimization of design. As discuss this optimization is depends on your input constraints. This constraint may be area or timing or both.

Than this optimized design is mapped into available libraries. This mapping depends on technology of library as well as functions available into library. It may be possible that some required functionality is not available in library. So this type of functions is converted into functions which are available into library. And after mapping of this design gate-level netlist is prepared, which is further used in Place and Route process.

8.3 WHY SYNTHESIS TOOL IS REQUIRED?

Synthesis process is possible either manually or using tool. For very small design manual synthesis is possible but as design becomes larger, this type of manual synthesis is not feasible. There are also some other reasons which leads to use synthesis tool.

Productivity:

Nowadays design becomes larger and larger. As design becomes larger no. of gates to implement also increase. So manually synthesis of that number of gates is not possible. So we require synthesis tool to synthesize any big design.

Design Tricks:

For implement any design much iteration are required. Manually it is not possible to implement all type of design and check which one is best for our application constraints. So here tool have many design tricks for synthesis. So it can try them depends on loads, fan-outs, library limitation etc.

Abstraction:

There are many high level issues like fan-outs, load capacitances etc. which are not possible to take into consideration. So this type of calculation is only possible by using Synthesis tool.

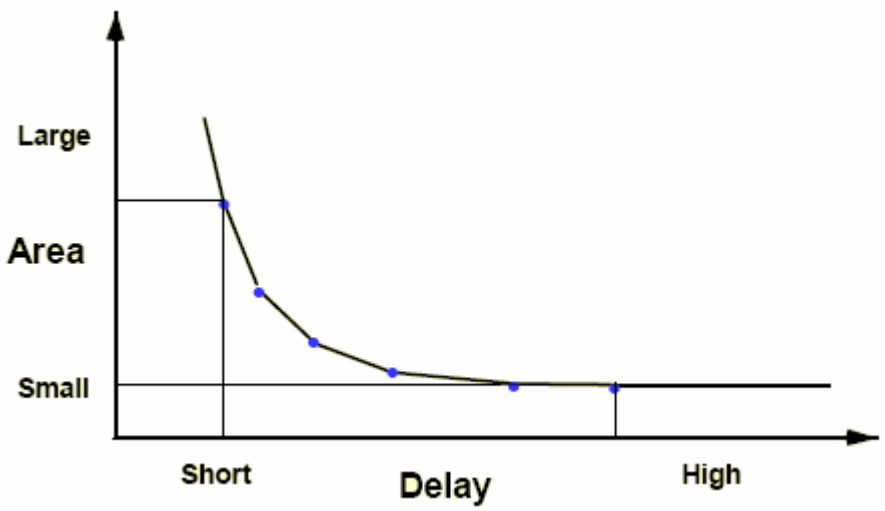
Reusability:

Your HDL code is not technology dependent. So it can be implementing using any of your designed library. So if you want to try your design using different libraries than it is not possible manually. So for this type of comparison automated tool is necessary.

And there are many issues like verification if synthesized design, number of parametric standards is also leads us to use Synthesis tool.

8.4 SYNTHESIS IS CONSTRAINT-DRIVEN

Synthesis process is constraints driven process. It depends on your given inputs. There are many type of trade-off like area and maximum clock frequency. Means as you want to increase clock frequency than you have to increase required area of design as shown in figure.



So we can say that for lower delay constraints design we require larger area to implement. So in Synthesis tool we give constraints to tool and it will give you final design.

9. PLACE AND ROUTE

This chapter covers the essential steps of a Gate to Layout flow. The basic steps of a Gate to Layout flow include the following:

1. Load library
2. Import gate netlist
3. Specify design constraints
4. Floor planning
5. Power planning
6. Physical synthesis
7. Clock tree synthesis
8. Routing
9. Physical verification
10. Post-layout verification

Most P&R projects have to move through these steps. However, the steps are not necessarily executed in the order listed. Below are examples of the variation that may occur in the execution of P&R:

- Physical verification (step 9) of the pad ring during floor planning (step 4).
- Physical synthesis (step 6) to assess the feasibility of a floor plan (step 4) without power planning (step 5).
- Some P&R flow perform physical synthesis (step 6) and clock tree synthesis (step 7) concurrently.

Not all the steps are necessary for a P&R project. Depending on the logic design requirements and fabrication process technology, some steps can be omitted, or some steps can be added. For example,

- Clock tree synthesis might not be necessary at 0.6um process technology.
- Crosstalk noise violation analysis and fixing is a must for layout using 0.18um or smaller process technology.

9.1 LOAD LIBRARY

A P&R library contains two types of information

- technology library
- cell library

9.1.1 Technology Library

Wires that satisfy all layout design rules must be put in place by the router. The timer engine requires accurate parasitic capacitances and resistances for static timing analysis, crosstalk analysis and power analysis. Information regarding the layout design rules and capacitance look-up table are nested in the technology library. The following table lists some of the P&R tasks and their corresponding roles from the technology library that are necessary to perform the tasks.

P&R Task	**Technology library**
Congestion driven P&R	Process design rules.
Timing driven P&R	Routing Parasitic.
Cross-talk aware P&R	Routing parasitic with coupling capacitance.
Electro-migration (EM) sign-off	Electro-migration limit for each metal layer.
Metal and via density filling	Metal and via density requirement

9.1.2 Cell Library

The cell library holds both logical and physical information of the logic cell. The logical information is similar to the contents of a synthesis library. Several different types of information exist and cannot all be listed here. The following are some of the common contents of the cell
library.

- Cell type (e.g. combinational, sequential, pad, timing model, etc.)
- Pin-to-Pin delay
- Slew for output pin
- Capacitive loading on input pin
- Leakage and dynamic power consumption
- Design rule (e.g. maximum input slew for input pins and maximum load for output pins)
- Maximum allowable noise for input pins, and holding resistance for output pins

The full physical layouts of the cells are too complicated to be used in a P&R environment. Hence, the physical information in the cell library contains a simplified version of the layout commonly known as an "abstract". An abstract contains the following information:

- Cell name
- Size of the cell
- Allowable orientation
- Pin names and their layout geometries
- Routing blockages
- Process antenna areas for the pins

Power and ground pins are typically excluded from the logical library but they must be included in the abstract library. The following graphic (Fig. 1.1) depicts an example of a standard cell abstract.

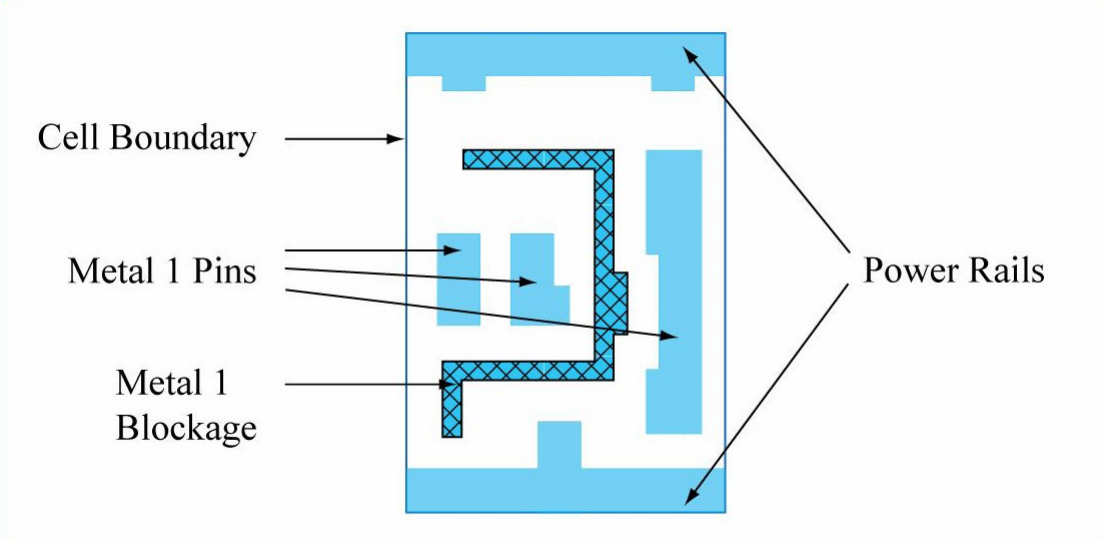


Fig. 9.1 An example of a standard cell abstract.

A popular format for abstract library is Layout Exchange Format (LEF). LEF format supports both the technology library and the cell library. The technology defines the name of the layers (e.g. metal layers and via layers) that are used in the cell libraries, therefore it is mandatory to load the technology library before the cell library.

9.1.3 Three Types of Cell Libraries

There are three types of cell libraries:

- Standard cell library
- Pad cell library
- Macro library

Standard cells must be placed in the "core" and on the "cell row". "Core" and "Cell row" are described in the latter part of this chapter. Pads and macros do not have this restriction and can be placed anywhere on the layout. However, pads are typically placed on the peripherals of the layout. All P&R tools have specific functions to automate the placement of the pads.

9.2 IMPORT GATE NETLIST

Verilog is the most popular gate-level netlist format. It is also the preferred netlist format for most P&R tools. An alternative is the use of the VHDL gate-level netlist.

After loading the netlist into the P&R tool, the logic gate in the design bind to their cell master in the cell libraries. This process seems trivial, but it is necessary to cater to all situations where cells from different libraries have the same cell name.

9.2.1 Cell and Instance

A cell library is a collection of cells. Standard cell refers to a logic gate. I/O cells are usually called I/O pads. Hard macros refer to the layout of the IP. An IP without a layout implementation is called a soft macro.

An instance refers to a cell in the design. Rather than saying “adding the cell AND2D1 to the design”, the common term used is “instantiating the cell AND2D1 to the design”. Every instance in the same design hierarchy must have a unique instance name. The following example is an instantiation of a 2-input AND gate:

```
AND2D1 inst0 ( .A1(net1), .A2(net2), .Z(net3) );
```

The instance name of the 2-input AND gate is “inst0”, and the master name (or cell name) is AND2D1. Figure 9.2 summaries the definition of cell and instance.

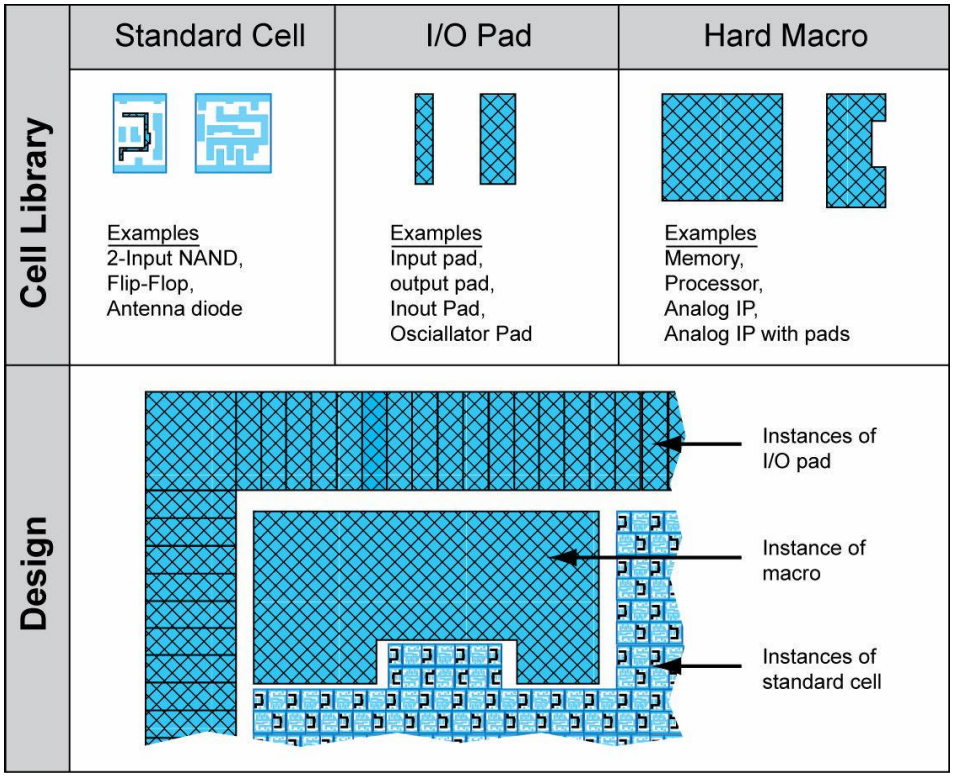


Fig. 9.2 Cell and instance

9.2.2 Define power and ground connections

Refer to the instantiation of a 2-input AND gate.

AN2D1 inst0 (.A1(net1), .A2(net2), .Z(net3));

Gate level netlist does not typically include the connection to the power and the ground supplies. However, these connections have to be defined before layout implementation. All P&R tools use global nets and wildcards to define these connections. For example, the following Design Exchange Format (DEF) statements create two new nets named VDD and VSS. It also connects all the pins with name "vdd" to net VDD, and all the pins with name "vss" to net VSS. The "*" in the commands is the wildcard that matches to any instance name.

-VDD (* vdd)

-VSS (* vss)

If VDD and VSS are the only two power nets in the design, and all the power pins of the cell instances are named "vdd" and "vss", then these two commands are sufficient to define all the power supply connections.

Tie-high and tie-low refer to the connection of the input pins to the power supply. Tie-high and tie-low are represented as 1'b1 and 1'b0 in the Verilog netlist. There are two ways to physically connect tie-high and tie-low nets. These nets can either connect directly to the power and the ground nets, or connect using tie-high and tie-low cells. Whichever the case, it is t he design of the pad library and the standard cell library that determines the appropriate type of connection.

9.2.3 Instance name and hierarchical instance name

A design can contain many hierarchical levels. The design name is the same as the top-level design hierarchy name. An instantiation can be a cell or a sub-hierarchy. A netlist can instantiate the same sub-hierarchy several times. However, in order to allow the layout of the sub-hierarchy to be implemented differently for each instance, the sub-hierarchy must be "uniquified" by duplicating the sub-hierarchy with different hierarchy cell names. Figure 1.3 illustrates the "uniquification" process. Note that instance names are preserved during uniquification.

A full hierarchical name is required to refer to a particular instance. Refer again to Figure 1.3. Assume there is a flip-flop with instance name "U1" in "Adder32". As "Adder32" is instantiated four times before uniquification, we can refer to that flip-flop in the four instances of "Adder32" as Inst0/U1, Inst1/U1, Inst2/U1 and Inst3/U1. As instance names are preserved during uniquification, the instance names of the four flip-flops remain unchanged after uniquification.

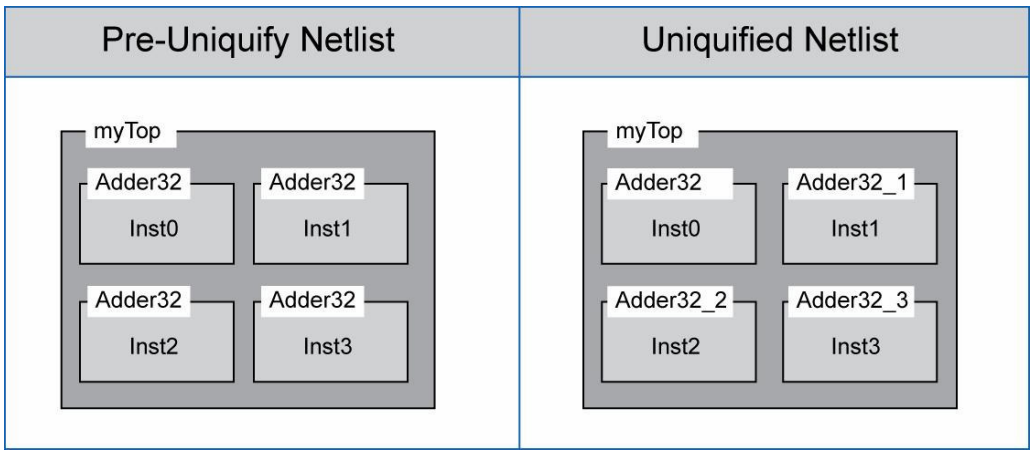


Fig. 9.3 Uniquify the netlist.

9.3 SPECIFY DESIGN CONSTRAINT

P&R is a constraint-driven process. In the absence of constraints, P&R optimization is purely congestion-driven. A timing constraint is an important part of the design constraint. Timing constraints specify the timing goals of the design. In order to perform timing-driven placement or physical synthesis, timing constraints must be available. Timing constraints are most likely to be specified in SDC format.

The timing constraints should be specified at the top-level of the design. It is necessary to specify a complete top-level timing constraint for a timing-driven flow. Any unconstrained timing paths will not be optimized for timing performance. Hence, unconstrained paths might be implemented with logics that are very slow or with extraordinarily large slew. On the other hand, over-constraining the timing requirement is undesirable. An over-constrained design can result in an implementation that is unnecessarily large in area and the P&R will possibly consume a much longer run-time.

In addition to timing constraints, there are constraints not related to timing performance. There is no standardized name for this type of constraint, so it will be termed a “non-timing constraints” in this book.
There are many different types of non-timing constraints. The following list describes some of them.

- Design rules which include maximum fan-out, maximum slew and maximum capacitance loading.
- Scan-chain re-ordering and re-partitioning.
- Selective hierarchy flattening.
- Buffering of inputs and outputs with user-specified cells.
- Identification of cells that the tool cannot modify or can only resize.
- Identification of nets that must be preserved during logic optimization.
- Disallow the use of certain cells.
- Assign higher priority to certain nets so as to achieve shorter wiring length.
- Restriction in the area that certain cells can be placed.
- Among others.

Non-timing constraints can be employed to ensure the physical implementation meets the design requirements, improvement of layout quality and turn-around time, as well as to work-around the limitations
of the P&R tools.

9.4 FLOOR PLANNING

Floor planning is the first step of physical layout implementation. A floor plan should include the following decisions:

- Size of the layout
- Core area
- Placement of i/o pads and i/o pins
- Placement of the hard macros

A floor plan should include the placement of all the pads (or pins) and the hard macros. However, the standard cells are not placed yet and no routing is performed at this stage.

9.4.1 Size of the layout

The first step in floor planning is to define the outline of the layout. If the layout is rectangular, only the length and the width of the layout are required. More co-ordinates are needed to define the outline of a rectilinear layout, such as an L-shape layout. Most of the P&R tools do not alter the size of the layout specified by the user.

9.4.2 Core Area

The core area is usually defined by specifying the distance between the edge of the layout and the core, as shown in Figure 9.4.

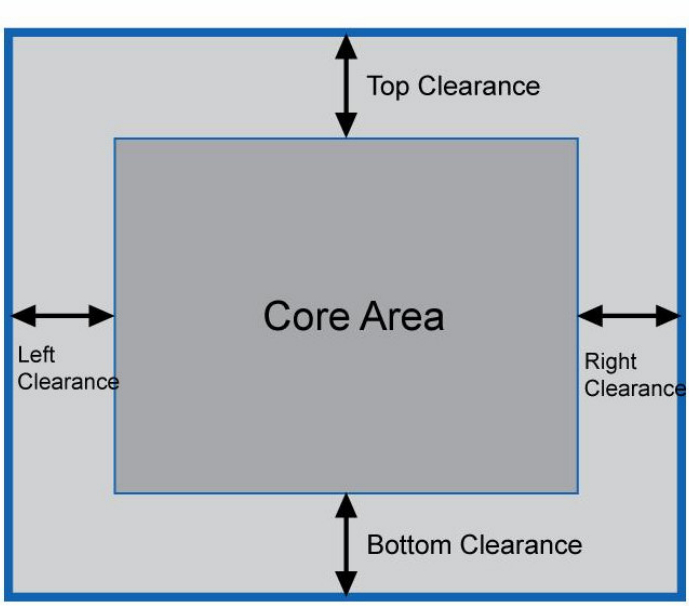


Fig 9.4 Definition of the core area.

All standard cells must be placed in the core area. I/O pads and macros do not have this restriction although it is common to place macros in the core area. The area outside the core can be used to place the I/O pads, the I/O pins, and the core power rings.

Standard cells are placed in rows, similar to placing books on a book shelf. The rows are called “cell rows” and are drawn inside the core area. All cell rows have the same height. There are three common ways to arrange the cell rows (Figure 9.5).

The most common approach for layout with more than three metal layers is to flip every other cell row which does not leave a gap between the cell rows.
The second configuration is to flip every other cell row, but leave a gap between every two cell rows. The purpose of the gaps is to allocate more resources for the inter-connect routings.

The last configuration is to leave a gap between every cell row, and not flip the cell rows. This configuration is useful when only two or three metal layers are available for routing.

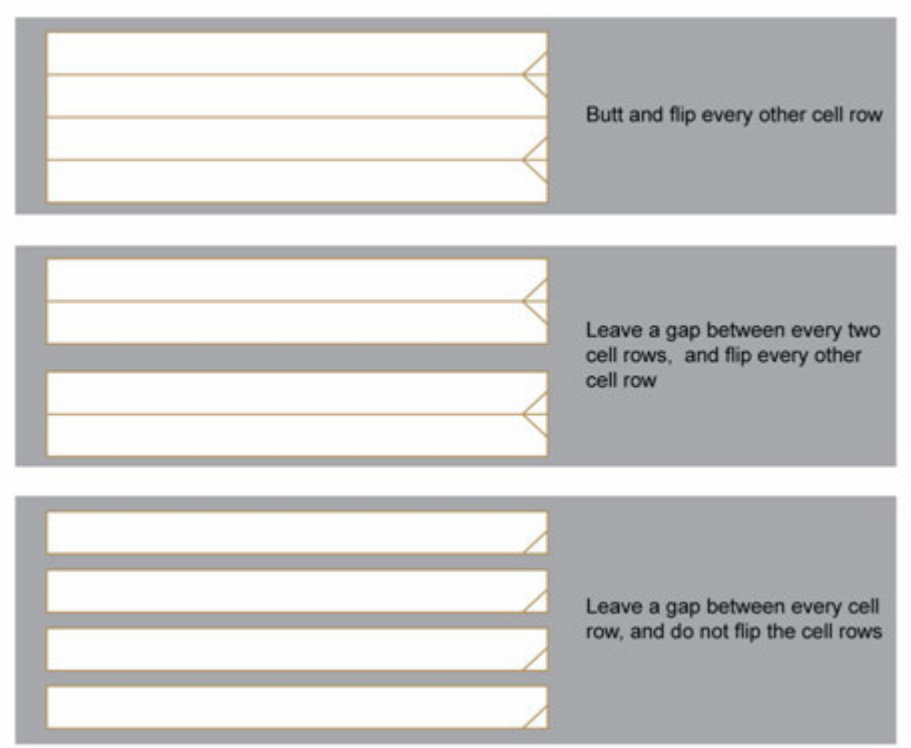


Fig. 9.5 Three types of cell row configuration.

The "slanting lines" on the right of the cell rows in Figure 1.5 denote the orientation of the cell rows. Modern P&R tools will fill the core area with cell rows automatically. Some P&R tools require the user to specify the areas in the core where the cell-row should be created.

If all the standard cells are in the same power domain, then only one core area is required. In a multiple core power P&R flow, more than one core area must be defined, and every core area must associate itself with a power domain.

9.4.3 Placements of IO Pads and IO Pins Geometries

For a chip-level layout, the next step is to place the IO pads. The P&R tool can fill the gaps between the pads with pad filler cells and corner cells. For a block-level layout, the user needs to define the location and geometries (size and metal layer) of every IO pin.

9.4.4 Placements of the Hard Macros

The floor plan is complete if the design does not contain any hard macro[1]. Otherwise, the next step is to place the hard macros. Placing the hard macros may not be a simple task. A good macro placement has the following qualities:

- Provides a compact layout.
- Does not cause routing congestion.
- Does not make timing closure difficult.
- Allows robust power routing between the power pads and the macros.

The biggest challenge in placing the macros is in assessing the quality of the floor plan, which cannot be achieved without executing the rest of the flow. Thus, floor planning is an iterative and time consuming process. The trick in performing floor planning is to shorten the turn- around time of the iterations, and to reduce the number of iterations. The following figure 9.6 depicts a simple floor plan of a chip-level layout with only one macro.

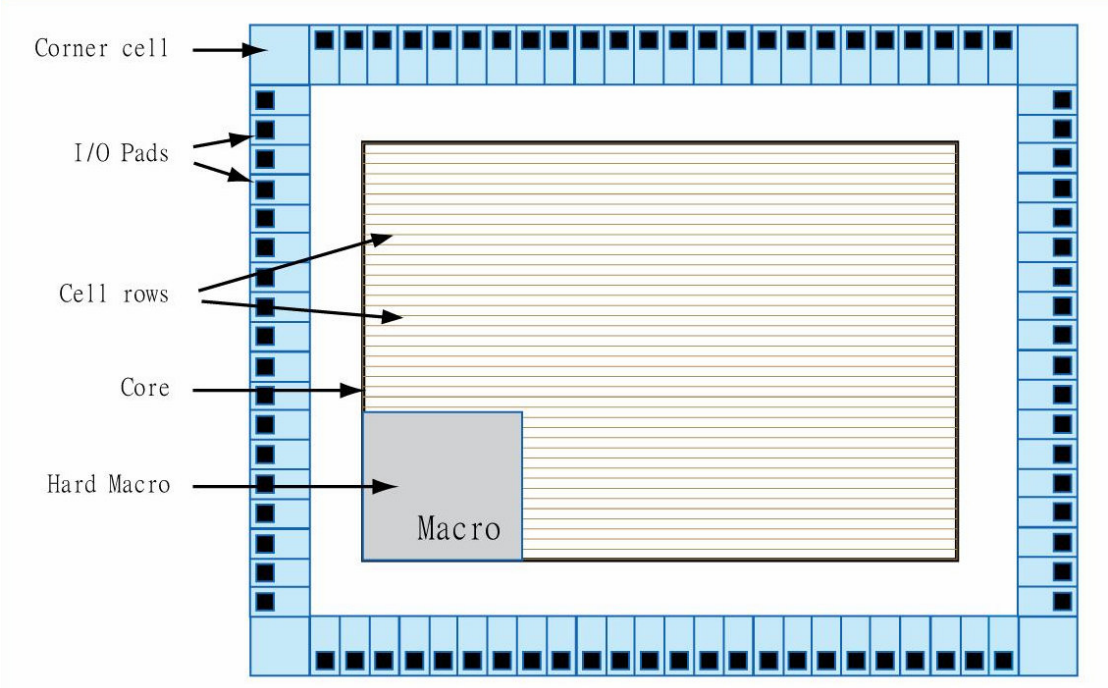


Fig. 9.6 A floor plan with one macro and I/O pads

9.5 POWER PLANNING

All connections to the power and ground nets are routed during power planning. The only exception is the tie-high and the tie-low nets. Most P&R tools use a dedicated router to route the power nets. All power routings created by the power router are considered pre-routes, and are not modified by the detailed router when the signal nets are routed.

The key consideration for power planning is:

- An acceptable IR-drop from the power pads to all power pins
- Meeting electro-migration requirements
- Does not result in routing congestion
- Compact layout

A power plan consists of several types of power structure. Figure 9.7 illustrates a typical sequence to construct the power structures.

1. Core power rings are routed first
2. Core power pads are connected to the core power rings
3. The power rings are added around the macros where necessary
4. Vertical stripes and horizontal stripes are added to reduce the IR- drop at the power rails of the standard cells and the macros
5. The power pins of the hard macros are tapped to the core rings or the power stripes
6. If tie-high and tie-low cells are not used, the tie-high and tie-low inputs to the hard macros and IO pads are tapped to the power structures
7. The power rails for the standard cell are added to the power plan

The power rails can tap the power from the core power rings, the power stripes and the macro power rings

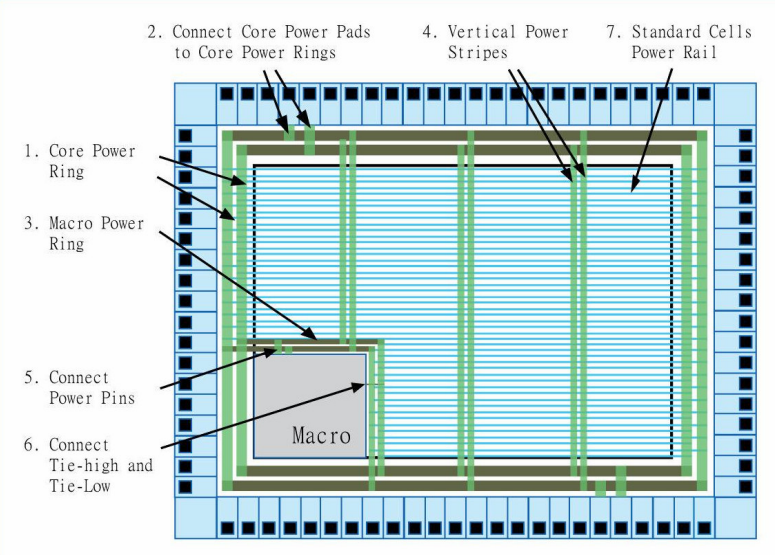


Fig. 9.7 Steps in building a power plan.

9.6 PHYSICAL SYNTHESIS

Physical synthesis refers to the placement of the standard cells and the optimization of the layout base on the design constraints.

Physical synthesis consists of a few typical phases:

- Global placement
- Global routing
- Physical optimization
- detailed placement
- further physical optimization

After physical synthesis, all standard cells are placed on the cell rows. The placement should be legalized, which means that the standard cells are on the cell row, on the placement grid, non-overlapping, and the power pins of the standard cells are properly connected. The placement should be routable, meeting the timing goal, and satisfy the placement constraints specified by the user.

In order to meet the timing goal, the tool might need to optimize the netlist. Different tools have different capabilities given the type of optimization it can perform. The user can also configure the type of optimization the tool can utilize. Some of optimization techniques a P&R tool can employ are listed below. The optimization techniques are listed in an increasing order of structural change relative to the original netlist.

- Gate sizing
- Buffer insertion and removal
- Pin swapping
- Cloning
- Logic restructuring
- Architecture retargeting

Physical synthesis becomes essential when the IC industry started to adopt process technologies that are 0.25um and smaller. The following table summaries the evolution of the placement methodology.

Process Technology	**Physical Optimization Techniques**
0.6um and larger	Placement is congestion driven. Manual insertion of buffers to long nets after routing.
0.35um	Placement is timing driven. Physical optimization is restricted to gate sizing andbuffer insertion.
0.25um and smaller	Physical synthesis is fully adopted

Table 9.1 Evolution of placement methodology with process technology.

Most P&R flows will not attempt to restructure the logic in the clock network. Some P&R tools have the ability to size the cells in the clock network during physical synthesis. To achieve a good estimation of the inter-connect parasitics, global routing is performed during physical synthesis. It is assumed that detailed routing will match global routing closely so that physical synthesis is optimizing on the real critical paths.

The routing congestion map can be derived from global routing. Any routing

congestion at this stage should be resolved as much as possible by reiterating the placement with additional controls, or by improving the floor plan and power plan. Before proceeding further to the layout design, the layout is now ready to perform IR-drop analysis. Two main objectives of IR-drop analysis are to ensure

- All power pins of the hard macros and the standard cells are connected to the power structures
- The voltage drops in the power structures are within acceptable limit.

9.7 CLOCK TREE SYNTHESIS

After all the standard cells are placed, the clock nets are buffered. The following list provides the additional requirements for synthesizing clock trees when compared to the buffering of the high fan-out non-clock nets:

- clock latency
- clock skew
- restriction on the type of buffer and inverter the clock tree can use
- stricter signal slew requirements on the clock nets

Clock latency is the delay of the clock signal from the clock source to the clock pin. Clock skew is the difference between the clock latencies and the two clock pins.

It is straight forward to specify the clock tree requirements to the P&R tool. If the clock tree starts from one source and fans-out only to the clock pins of the flip-flops and the macros, the clock tree meets the requirements of the P&R tool. Unfortunately, this is not always the case. For example, the clock tree schematic shown below has the following additional requirements:

- The clock latencies of flip-flops div_reg* do not have to be balanced with the clock latencies of the other flip-flops
- A small clock skews between the flip-flops div_reg*
- A small clock skews for the rest of the flip-flops in both functional mode and test mode
- Optimize for shorter clock latencies during functional mode

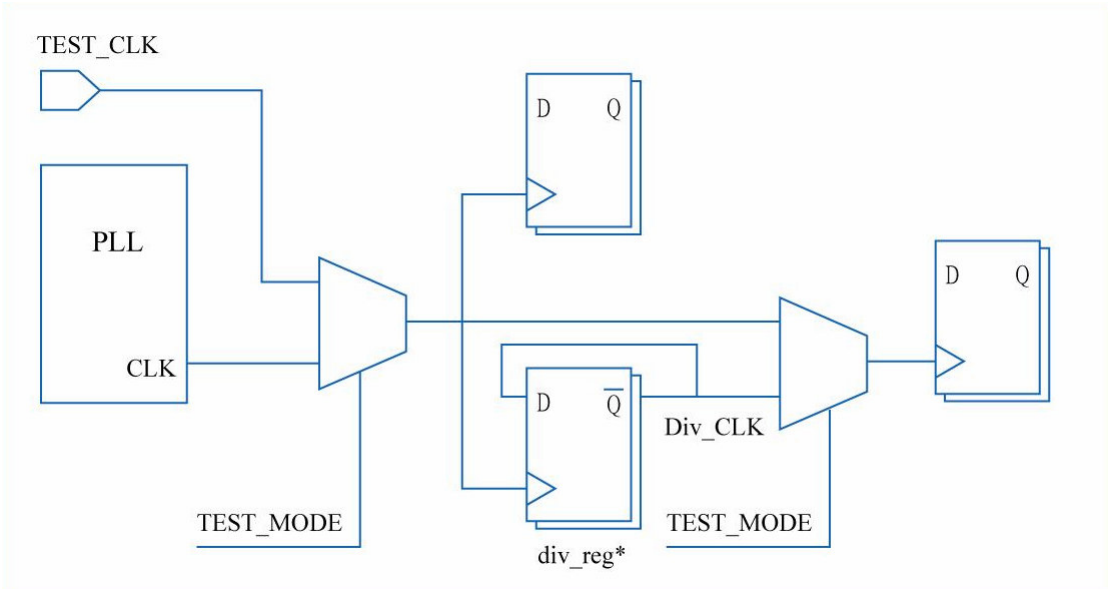


Fig. 9.8 Clock tree schematics example

It is not uncommon for the clock tree synthesis algorithm to generate a poor clock tree when complicated blockages exist in the layout. The following is an example of a poor clock tree: Additional routing requirements are often applied on the clock nets. In order to reduce noise coupling, the route spacing to the clock net can be doubled. Shielding the clock net is another option to reduce noise coupling. For a clock with very high clock frequency, it may be necessary to use multiple-via on the clock routing to meet the electro-migration rules.

Before clock trees are inserted, the tool uses ideal clock latencies and ideal clock skews for timing analysis. After the clock trees are synthesized, the tool should use the actual clock latencies and clock skews for the timing analysis. It is now possible to analyze and fix hold violations using computed clock latencies associated with every clock end-point after clock tree synthesis. Hold violations should be fixed first in "best corner" operating condition, and then in the "worst corner" operating condition. Current P&R tools fix hold violations by adding delay to the data path. The tool will not attempt to make changes to the clock path. It is advisable to analyze the buffers added by the tool for hold fixing. If the result is not satisfactory, clock trees might need to be re-synthesized using different approaches.

9.8 ROUTING

The process of routing the non-power signals is called “detailed routing”. After clock tree synthesis, empty space still exists in the cell rows. These empty spaces in between the standard cells will eventually be filled by filler cells. Fillers can be inserted either before or after detailed routing. If the fillers contain metal routing other than the power rail, then the fillers should be inserted before routing. Otherwise, it is best to add the fillers after routing. Figure 9.9 shows the results before and after filler cell insertion.

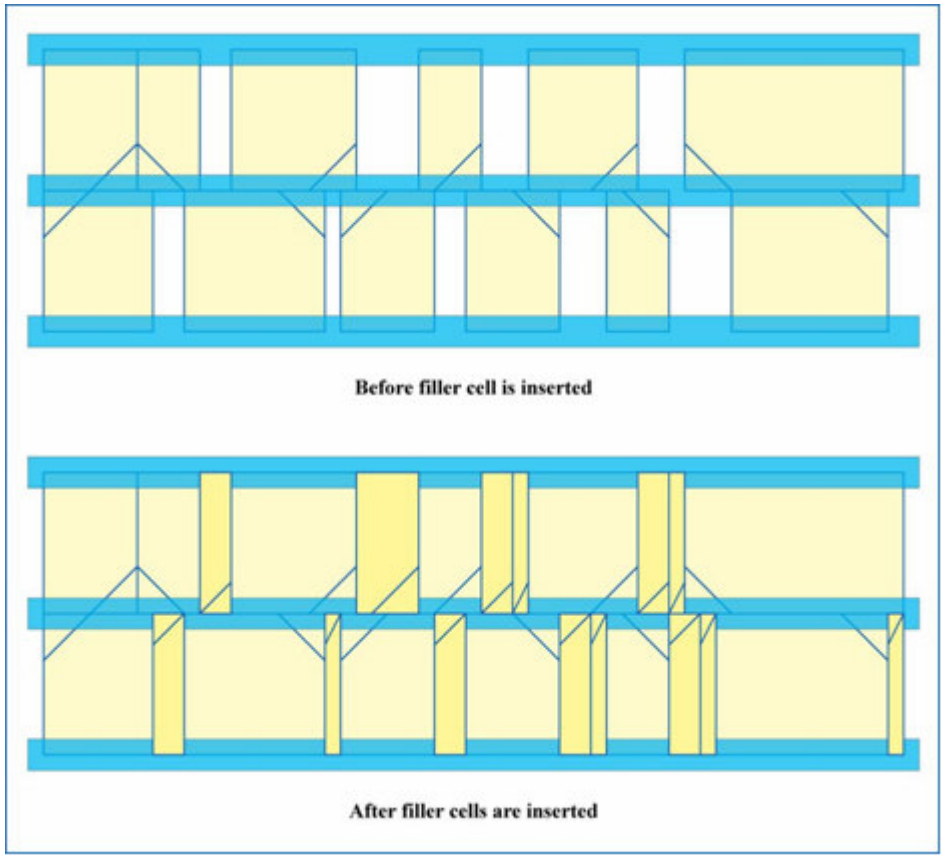


Fig. 9.9 Insertion of the filler cells

Routing is performed throughout P&R flow. Powers are routed during power planning by a dedicated router. During physical synthesis, the signal nets are “globally routed” using the global router. The routings are called global routes. Global routes allow the P&R tool to resolve routing congestion and estimate the routing

parasitic. After the clock trees are synthesized, the clock trunks are routed with actual metal geometries. This occurs before the clock trees are re-optimized. This is then followed by detailed routing.

Detailed routing is carried out in stages. The detailed routing stages are different in various P&R tools, but the methodology used for detailed routing is similar. The stages are outlined below.

1. Track routing: Global routing uses a very coarse routing grid. Track routing assigns the global routes to the actual routing tracks.
2. Detailed routing with only metal one: Connections between cells that are placed side-by-side are possible candidates.
3. Connecting the rest of the signal nets by following the result of track routing: The aim is to connect all the routings so that there are none "Open". The routings can be full of routing violations (e.g. short) and design rule violations (e.g. metal to metal spacing violations).
4. Resolve routing violations iteratively: The detailed router divides the layout into regions and works within each region to clean up the routing violations. This process iterates with a region of different sizes and aspect ratios. The iterative process continues until there are no more routing violations or the limit on the number of iterations has been reached.
5. Iterating between fixing antenna violations and cleaning up routing violations: New routing violations can be introduced during the process of fixing the antenna violations.
6. Optimizing the detailed routes: The types of optimization to be performed will depend on the user's specifications and the tool's capability. The optimization can include minimizing wire jog and switching of routing layers, or even out the spacing between the routes, and replacing single via with redundant via.

All P&R tools have the functionality to perform cell-level DRC and LVS on a routed design. However, it is mandatory to perform a full-layer (or sometimes called full-GDS) physical verification.

9.9 PHYSICAL VERIFICATION

After the layout is routed, a GDS containing the design layout can be generated from the P&R tool. Ideally, the GDS is ready to be sent for mask making and fabrication. However, any mistakes made during the P&R flow will be left undetected. Physical verification is an independent process of verifying the integrity of the layout.

There are three types of physical verification:

- Design Rule Check (DRC)
- Layout Versus Schematic (LVS)
- Electrical Rule Check (ERC)

DRC checks the layout geometries for the manufacturing process. A full DRC deck contains hundreds of DRC rules. The following are some common DRC rules:

- Spacing between geometries
- Minimum and maximum width of the geometries
- Density of the metals, the poly and the diffusion
- Antenna violation
- Via reliability

LVS checks the layout for correct connectivity between the devices in the circuit. A circuit device can be a transistor, a resistor, a capacitor, a diode, among others. During LVS verification, circuit devices and the inter-connections are extracted from the layout and saved as a layout netlist. This typically exists in a format similar to that of the spice format. The layout netlist is compared with the post-layout design netlist (usually i n Verilog format).

ERC identifies errors in layout that are related to electrical connections. Examples are latch-up protection, floating substrate, floating well, and bad device

connections. Currently, the ERC rules are typically embedded in the DRC and LVS rules. At 0.35um or larger technology, it is possible for the layout implementation to meet timing performance without physical synthesis and clock tree synthesis. In this case, the pre-layout netlist is the same as the post-layout netlist. However, P&R with 0.35um or smaller technology definitely requires physical synthesis and clock tree synthesis for timing closure. Hence, the P&R tool has to make modifications to the netlist and output a post-layout netlist. As LVS performs the comparison between the post-layout netlist and the layout, it does not verify that the post-layout netlist is functionally equivalent to the pre-layout netlist. Equivalent checker tools are needed to guarantee the post-layout netlist has the same functionality as the pre-layout netlist. The complete physical verification flow is shown in Figure 9.10.

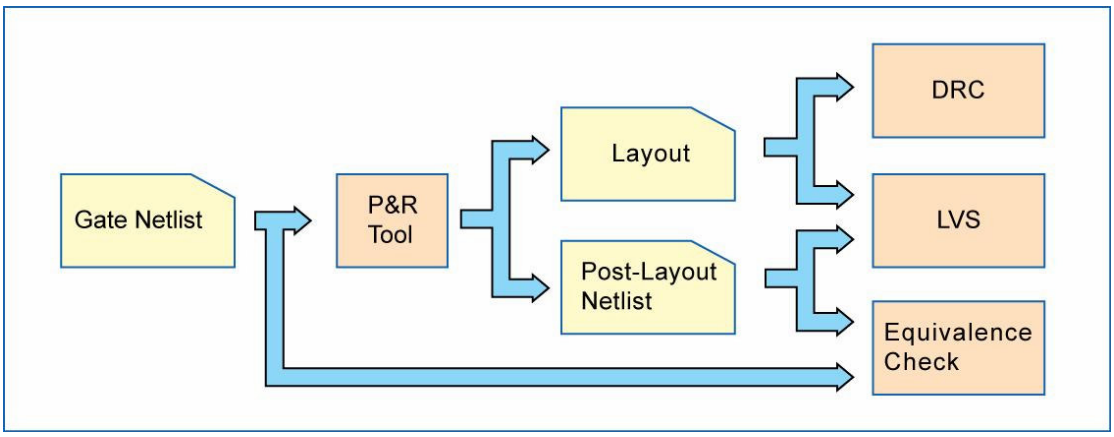


Fig. 9.10 Physical verification flow.

9.10 POST-LAYOUT VERIFICATION

Sign-off tools are tools that are used to perform the final check on the design. Due to run-time and memory usage considerations, P&R tools have to use simplified models during parasitics extraction, static timing analysis and other types of analyses. Hence, there is a need for sign-off tools.

On average, the parasitics extraction result from a P&R tool should be within a few percentage points from a sign-off parasitics extraction tool. However, some nets can exhibit a large difference, in excess of 100%. Similarly, the static timing engine in

the P&R tool and the sign-off static timing tool might show differences with respect to design constraints, delay calculation or tracing of the timing path. The sign-off tools may find violations the P&R tool misses. It may be necessary to correlate the interpretation of the design constraints by the P&R tool and the sign-off tools early in the P&R project. It is common practice to over-constrain the design by a small margin so that differences among the tools do not pose any concerns.

The result of parasitics extraction is usually stored in SPEF or SPDF format. The extracted parasitics can be back-annotated into the sign-off static timing analyzer (STA) tool. STA performs delay calculations and verifies the timing performance. The STA tool can generate a standard delay format (SDF) file that is used by a logic simulator for post-layout sign-off simulation. The post-layout verification flow is shown in Figure 9.11.

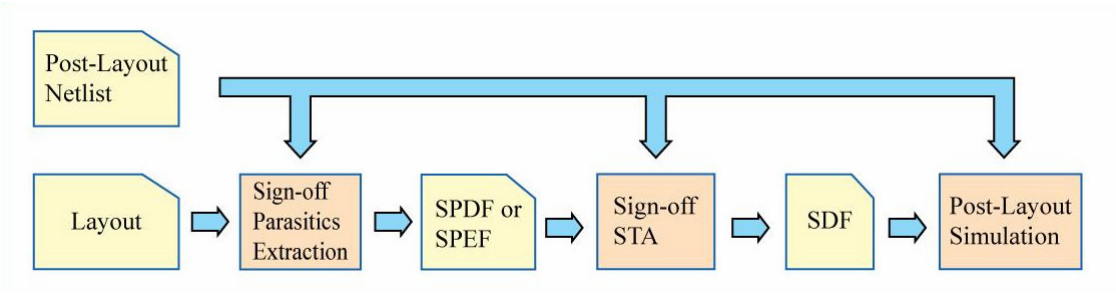


Fig. 9.11 Post-layout verification flow

If the timing goal is not achieved, then a possible attempt to close the timing is initiated. A correction to the layout by manual gate sizing and buffer insertion is standard practice to achieve the timing goal. Subsequently, the post-layout verification is repeated. Manual modification on the layout is too tedious with many timing violations. Therefore, it is also possible to back-annotate SPEF or SPF into the P&R tool, and let the tool optimize the layout to meet the timing requirement. Care should be taken to ensure that the P&R tool does not create new timing violations while fixing existing timing violations.

10. AES (ADVANCED ENCRYPTION STANDARD)

For explain flow of Synthesis and PNR here I am taking one example. Here I am taking Advanced Encryption standard (AES) as an example and in further chapters its synthesis and PNR will be explained.

The Advanced Encryption Standard (AES) specifies a FIPS-approved cryptographic algorithm that can be used to protect electronic data. The AES algorithm is a symmetric block cipher that can encrypt (encipher) and decrypt (decipher) information. Encryption converts data to an unintelligible form called cipher text; decrypting the cipher text converts the data back into its original form, called plaintext. The AES algorithm is capable of using cryptographic keys of 128, 192, and 256 bits to encrypt and decrypt data in blocks of 128 bits.

.

10.1 NOTATION AND CONVENTIONS

10.1.1 Inputs and Outputs

The input and output for the AES algorithm each consist of sequences of 128 bits (digits with values of 0 or 1). These sequences will sometimes be referred to as blocks and the number of bits they contain will be referred to as their length. The Cipher Key for the AES algorithm is a sequence of 128, 192 or 256 bits. Other input, output and Cipher Key lengths are not permitted by this standard.

The bits within such sequences will be numbered starting at zero and ending at one less than the sequence length (block length or key length). The number i attached to a bit is known as its index and will be in one of the ranges $0 \leq i < 128$, $0 \leq i < 192$ or $0 \leq i < 256$ depending on the block length and key length (specified above).

10.1.2 Bytes

The basic unit for processing in the AES algorithm is a **byte,** a sequence of eight bits treated as a single entity. The input, output and Cipher Key bit sequences are processed as arrays of bytes that are formed by dividing these sequences into groups of eight contiguous bits to form arrays of bytes. For an input, output or Cipher

Key denoted by *a*, the bytes in the resulting array will be referenced using one of the two forms, a_n or $a[n]$, where n will be in one of the following ranges:

Key length = 128 bits, $0 \leq n < 16$; Block length = 128 bits, $0 \leq n < 16$;
Key length = 192 bits, $0 \leq n < 24$;
Key length = 256 bits, $0 \leq n < 32$.

All byte values in the AES algorithm will be presented as the concatenation of its individual bit values (0 or 1) between braces in the order $\{b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0\}$. These bytes are interpreted as finite field elements using a polynomial representation:

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0x^0 = \sum_{i=0}^{7} b_i x^i \quad \ldots (10.1)$$

For example, {01100011} identifies the specific finite field element $x^6 + x^5 + x + 1$.

It is also convenient to denote byte values using hexadecimal notation with each of two groups of four bits being denoted by a single character as in table. 10.1.

Bit pattern	Character Bit	Bit pattern	Character Bit	Bit pattern	Character Bit	Bit pattern	Character Bit
0000	0	0100	4	1000	8	1100	c
0001	1	0101	5	1001	9	1101	d
0010	2	0110	6	1010	a	1110	e
0011	3	0111	7	1011	b	1111	f

Table 10.1 Hexadecimal representation of bit patterns.

Hence the element {01100011} can be represented as {63}, where the character denoting the four-bit group containing the higher numbered bits is again to the left. Some finite field operations involve one additional bit (*b*8) to the left of an 8-bit byte. Where this extra bit is present, it will appear as '{01}' immediately preceding the 8-bit byte; for example, a 9-bit sequence will be presented as {01}{1b}.

10.1.3 Arrays of Bytes

Arrays of bytes will be represented in the following form:

a0, a1, a2, a3…… an-1

The bytes and the bit ordering within bytes are derived from the 128-bit input sequence

$input_0, input_1, input_2 \ldots input_{126}, input_{127}$

as follows:

$a_0 = \{input_0, input_1, \ldots, input_7\}$;

$a_1 = \{input_8, input_9, \ldots, input_{15}\}$;

:

:

:

$a_{15} = \{input_{120}, input_{121}, \ldots, input_{127}\}$. ----------

-10.2

Following table shows how bits within each byte are numbered.

Input Bit Sequence	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	…
Byte Number	0								1								2								
Bit Number in byte	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	…

Table 10.2 Indices for Bytes and Bits.

10.1.4 The State

Internally, the AES algorithm's operations are performed on a two-dimensional array of bytes called the **State**. The State consists of four rows of bytes, each containing ***Nb*** bytes, where ***Nb*** is the block length divided by 32. In the State array denoted by the symbol *s*, each individual byte has two indices, with its row number *r* in the range $0 \leq r < 4$ and its column number *c* in the range $0 \leq c < \boldsymbol{Nb}$. This

allows an individual byte of the State to be referred to as either $s_{r,c}$ or $s[r,c]$. For this standard, ***Nb*** = 4, i.e., $0 \leq c < 4$.

Here input – the array of bytes *in*0, *in*1, in2, … *in*15 – is copied into the State array as illustrated in Fig. 10.1. The Cipher or Inverse Cipher operations are then conducted on this State array, after which its final value is copied to the output – the array of bytes $out_0, out_1, \ldots out_{15}$.

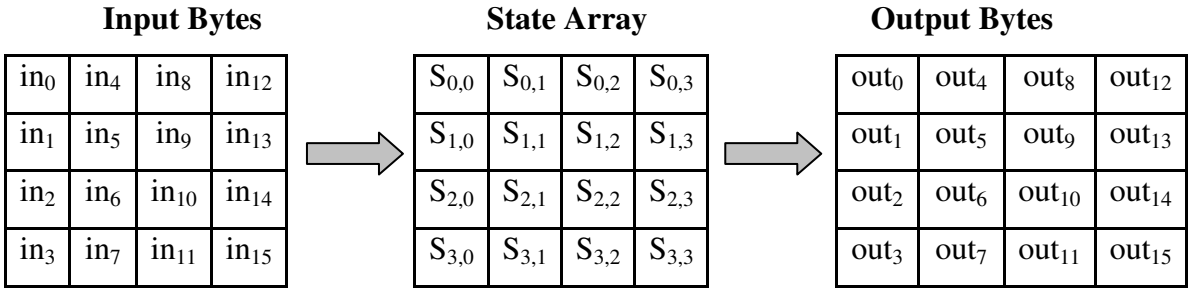


Fig. 10.1 State array input and output.

Hence, at the beginning of the Cipher or Inverse Cipher, the input array, *in*, is copied to the State array according to the scheme:

$$s[r, c] = in[r + 4c] \quad \text{for } 0 \leq r < 4 \text{ and } 0 \leq c < Nb. \text{ ---------- (10.3)}$$

and at the end of the Cipher and Inverse Cipher, the State is copied to the output array *out* as follows:

$$out[r + 4c] = s[r, c] \quad \text{for } 0 \leq r < 4 \text{ and } 0 \leq c < Nb. \text{---------- (10.4)}$$

10.1.5 The State as an Array of Columns

The four bytes in each column of the State array form 32-bit **words**, where the row number *r* provides an index for the four bytes within each word. The state can hence be interpreted as a one-dimensional array of 32 bit words (columns), *w*0...*w*3, where the column number *c* provides an index into this array. Hence, for the example in Fig. 10.1, the State can be considered as an array of four words, as follows:

$$w0 = s0,0\ s1,0\ s2,0\ s3,0 \qquad w2 = s0,2\ s1,2\ s2,2\ s3,2$$
$$w1 = s0,1\ s1,1\ s2,1\ s3,1 \qquad w3 = s0,3\ s1,3\ s2,3\ s3,3 \text{ .----------(10.5)}$$

10.2 ALGORITHM SPECIFICATION

For the AES algorithm, the length of the input block, the output block and the State is 128 bits. This is represented by *Nb* = 4, which reflects the number of 32-bit words (number of columns) in the State. For the AES algorithm, the length of the Cipher Key, *K*, is 128, 192, or 256 bits. The key length is represented by *Nk* = 4, 6, or 8, which reflects the number of 32-bit words (number of columns) in the Cipher Key.

For the AES algorithm, the number of rounds to be performed during the execution of the algorithm is dependent on the key size. The number of rounds is represented by *Nr*, where *Nr* = 10 when *Nk* = 4, *Nr* = 12 when *Nk* = 6, and *Nr* = 14 when *Nk* = 8. The only Key-Block-Round combinations that conform to this standard are given in Fig. 10.2.

	Key Length (Nk words)	Block Size (Nb words)	Number of Rounds (Nr)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Fig. 10.2 Key-Block-Round Combinations

For both its Cipher and Inverse Cipher, the AES algorithm uses a round function that is composed of four different byte-oriented transformations:

A. Byte substitution using a substitution table (S-box),
B. Shifting rows of the State array by different offsets,
C. Mixing the data within each column of the State array, and
D. Adding a Round Key to the State.

10.3 CIPHER

At the start of the Cipher, the input is copied to the State array using the conventions described in Sec. 10.1.4. After an initial Round Key addition, the State array is transformed by implementing a round function 10, 12, or 14 times (depending on the key length), with the final round differing slightly from the first Nr-1 rounds. The final State is then copied to the output as described in Sec. 10.1.4.

The round function is parameterized using a key schedule that consists of a one-dimensional array of four-byte words derived using the Key Expansion routine described in Sec. 10.5. The Cipher procedure is described in the Fig. 10.3. The individual transformations - **SubBytes()**, **ShiftRows()**, **MixColumns()**, and **AddRoundKey()** – process the State and are described in the following subsections.

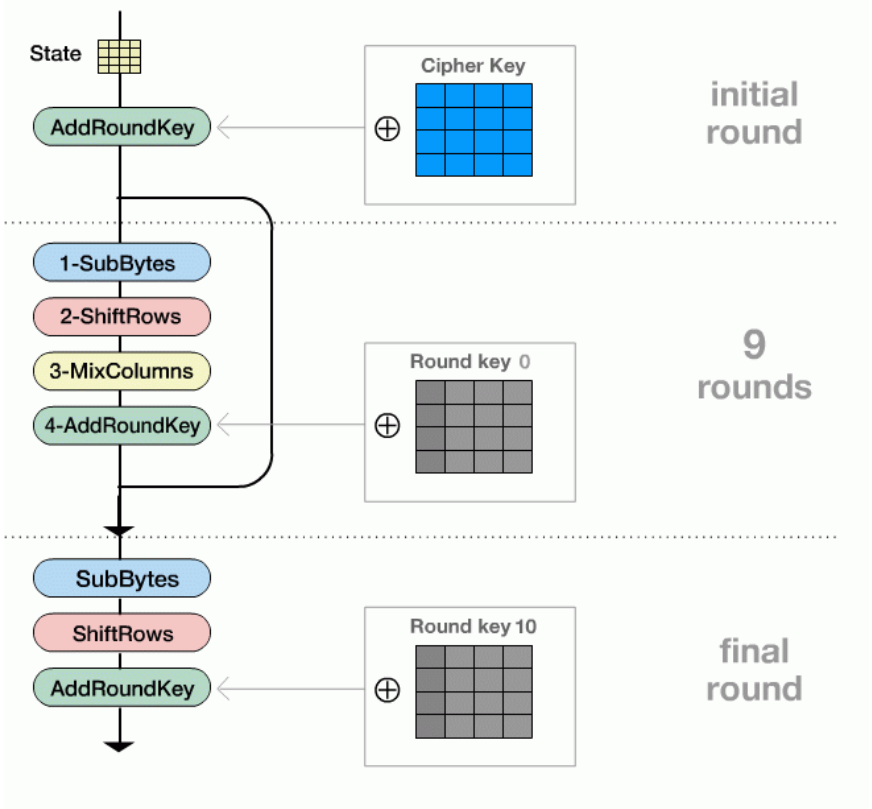


Fig. 10.3 Flow of AES Algorithm

As shown in Fig. 10.3, all ***Nr*** rounds are identical with the exception of the final round, which does not include the **MixColumns()** transformation.

10.3.1 SubBytes()Transformation

The **SubBytes()** transformation is a non-linear byte substitution that operates independently on each byte of the State using a substitution table (S-box). This S-box, which is invertible, is constructed by composing two transformations:

1. Take the multiplicative inverse in the finite field GF(2^8);
2. Apply the following affine transformation

$$b_i^{'} = b_i \oplus b_{(i+4)\bmod 8} \oplus b_{(i+5)\bmod 8} \oplus b_{(i+6)\bmod 8} \oplus b_{(i+7)\bmod 8} \oplus c_i \quad \ldots (10.6)$$

for $0 \leq i < 8$, where *bi* is the i^{th} bit of the byte, and c_i is the i^{th} bit of a byte *c* with the value {63} or {01100011}. Here and elsewhere, a prime on a variable (e.g. b') indicates that the variable is to be updated with the value on the right.

In matrix form, the affine transformation element of the S-box can be expressed as:

$$\begin{bmatrix} b_0^{'} \\ b_1^{'} \\ b_2^{'} \\ b_3^{'} \\ b_4^{'} \\ b_5^{'} \\ b_6^{'} \\ b_7^{'} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad \text{---------- (10.7)}$$

Figure 10.4 illustrates the effect of SubBytes() transformation on the state.

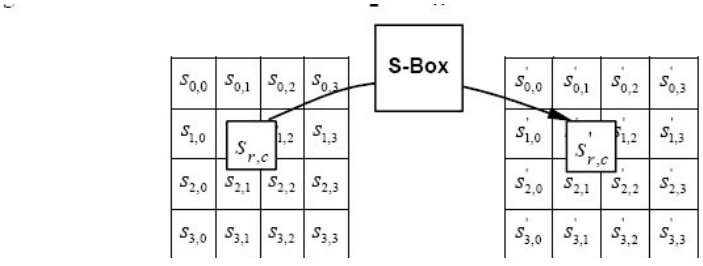


Fig 10.4 SubBytes() applies theS-Box to each byte of the state.

The S-box used in the **SubBytes()** transformation is presented in hexadecimal form in Fig. 10.5. For example, if $S_{1,1} = \{53\}$, then the substitution value would be determined by the intersection of the row with index '5' and the column with index '3' in Fig. 10.5. This would result in $S'_{1,1}$ having a value of {ed}.

		Y															
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**a**	**b**	**c**	**d**	**e**	**f**
X	**0**	63	7c	77	7b	f2	6b	6f	c5	30	1	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	4	c7	23	c3	18	96	5	9a	7	12	80	e2	eb	27	b2	75
	4	9	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	0	Ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	Fb	43	4d	33	85	45	f9	2	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	Ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	Dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	6	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	8
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	3	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Fig.10.5 S-box: substitution values for the byte XY (in hexadecimal format)

10.3.2 ShiftRows() Transformation

In the **ShiftRows()** transformation, the bytes in the last three rows of the State are cyclically shifted over different numbers of bytes (offsets). The first row, $r = 0$, is not shifted.

Specifically, the **ShiftRows()** transformation proceeds as follows:

$$S'_{r,c} = S_{r,(c+shift(r,Nb)) \bmod Nb} \quad \text{for } 0<r<4 \text{ and } 0 \le c < Nb, \text{---------- (10.8)}$$

where the shift value *shift*(*r*,*Nb*) depends on the row number, *r*, as follows (recall that ***Nb*** = 4):

$$shift(1,4) = 1; \quad shift(2,4) = 2; \quad shift(3,4) = 3. \quad \text{----------(10.9)}$$

This has the effect of moving bytes to "lower" positions in the row (i.e., lower values of *c* in a given row), while the "lowest" bytes wrap around into the "top" of the row (i.e., higher values of *c* in a given row). Figure 10.6 illustrates the **ShiftRows()** transformation.

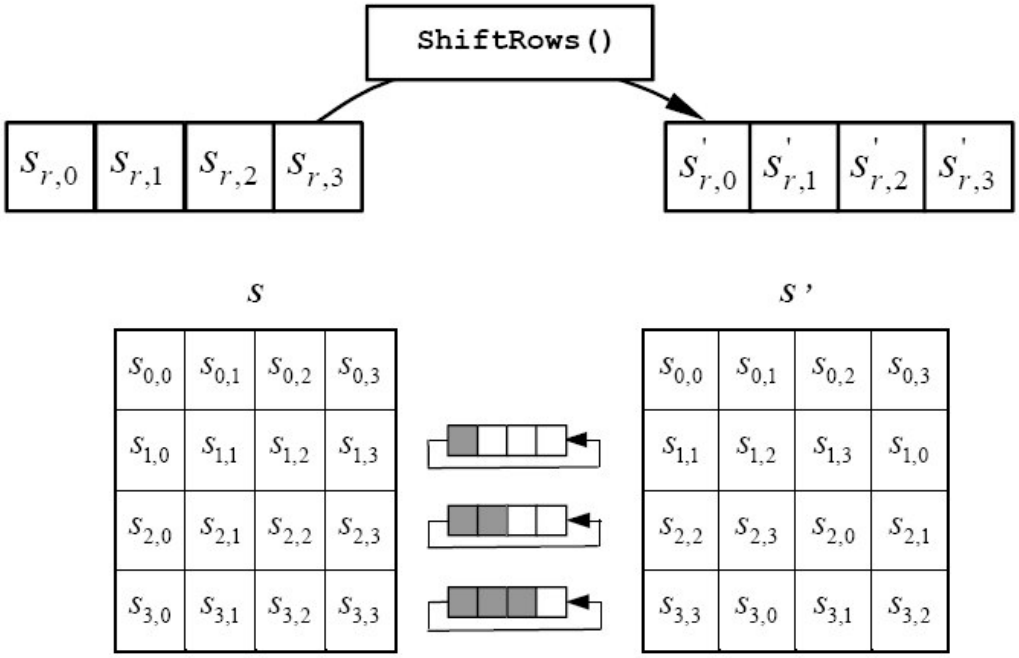


Fig. 10.6 ShiftRows() cyclically shifts the last three rows in the State.

10.3.3 MixColumns () Transformation

The **MixColumns ()** transformation operates on the State column-by-column, treating each column as a four-term polynomial. The columns are considered as polynomials over GF(2^8) and multiplied modulo $x^4 + 1$ with a fixed polynomial $a(x)$, given by

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} \quad \text{----------(10.10)}$$

This can be written as a matrix multiplication. Let

$s'(x) = a(x) * s(x)$:

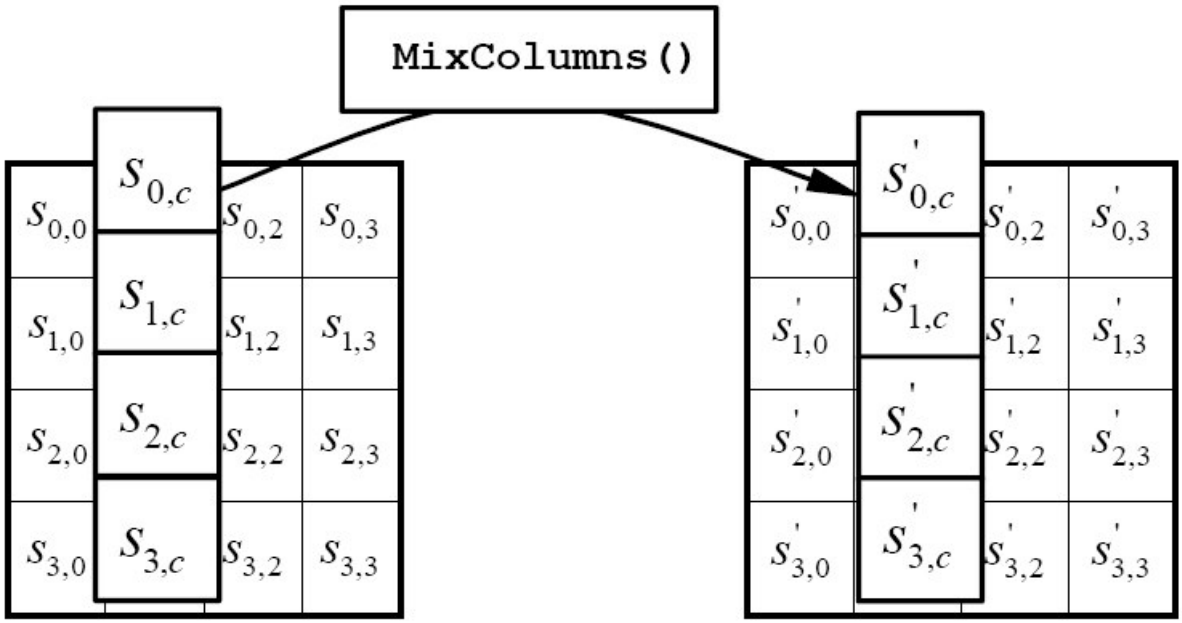


Fig 10.7 MixColumns() operates on the State column-by-column

10.3.4 AddRoundKey() Transformation

In the **AddRoundKey()** transformation, a Round Key is added to the State by a simple bitwise XOR operation. Each Round Key consists of ***Nb*** words from the key schedule. Those ***Nb*** words are each added into the columns of the State, such that

$$\left[S'_{0,c} + S'_{1,c} + S'_{2,c} + S'_{3,c} \right] = \left[S_{0,c} + S_{1,c} + S_{2,c} + S_{3,c} \right] + \left[w_{round+Nb+c} \right] \quad ...(10.12)$$

where [*wi*] are the key schedule words, and *round* is a value in the range $0 \leq round \leq \mathbf{Nr}$. In the Cipher, the initial Round Key addition occurs when *round* = 0, prior to the first application of the round function. The application of the **AddRoundKey** () transformation to the ***Nr*** rounds of the Cipher occurs when $1 \leq round \leq \mathbf{Nr}$.

The action of this transformation is illustrated in Fig. 10.8, where $l = round * \mathbf{Nb}$.

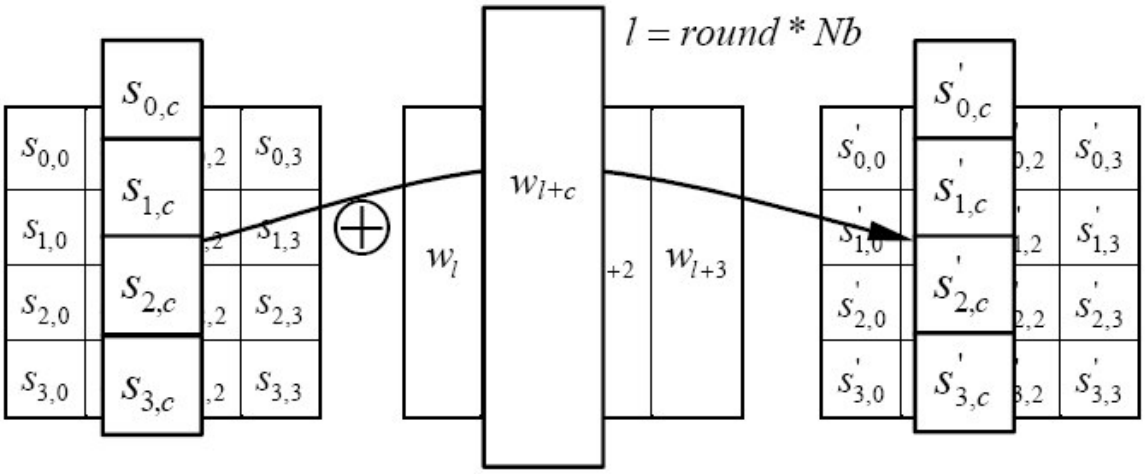


Fig. 10.8 AddRoundKey() XORs each column of the State with a word from the key schedule.

10.4 KEY EXPANSION

The AES algorithm takes the Cipher Key, *K*, and performs a Key Expansion routine to generate a key schedule. The Key Expansion generates a total of *Nb* (*Nr* + 1) words: the algorithm requires an initial set of *Nb* words, and each of the *Nr* rounds requires *Nb* words of key data. The resulting key schedule consists of a linear array of 4-byte words, denoted [*wi*], with *i* in the range $0 \leq i < Nb(Nr + 1)$.

SubWord() is a function that takes a four-byte input word and applies the S-box to each of the four bytes to produce an output word. The function RotWord() takes a word [*a*0,*a*1,*a*2,*a*3] as input, performs a cyclic permutation, and returns the word $[a_1,a_2,a_3,a_0]$. The round constant word array, Rcon[i], contains the values given by $[x^{i-1},\{00\},\{00\},\{00\}]$, with x^{i-1} being powers of x (x is denoted as {02}) in the field GF(2^8).

Here the first *Nk* words of the expanded key are filled with the Cipher Key. Every following word, w[i], is equal to the XOR of the previous word, w[i-1], and the word *Nk* positions earlier, w[i-*Nk*]. For words in positions that are a multiple of *Nk*, a transformation is applied to w[i-1] prior to the XOR, followed by an XOR with a round constant, Rcon[i]. This transformation consists of a cyclic shift of the bytes in a

word (RotWord()), followed by the application of a table lookup to all four bytes of the word (SubWord()).

It is important to note that the Key Expansion routine for 256-bit Cipher Keys (*Nk* = 8) is slightly different than for 128- and 192-bit Cipher Keys. If *Nk* = 8 and i-4 is a multiple of *Nk*, then SubWord() is applied to w[i-1] prior to the XOR.

10.5 IMPLEMENTATION ISSUES

1. Key Length Requirements

An implementation of the AES algorithm shall support *at least one* of the three key lengths specified 128, 192, or 256 bits (i.e., ***Nk*** = 4, 6, or 8, respectively). Implementations may optionally support two or three key lengths, which may promote the interoperability of algorithm implementations.

2. Keying Restrictions

No weak or semi-weak keys have been identified for the AES algorithm, and there is no restriction on key selection.

3. Parameterization of Key Length, Block Size, and Round Number

This standard explicitly defines the allowed values for the key length (***Nk***), block size (***Nb***), and number of rounds (***Nr***). However, future reaffirmations of this standard could include changes or additions to the allowed values for those parameters. Therefore, implementers may choose to design their AES implementations with future flexibility in mind.

4. Implementation Suggestions Regarding Various Platforms

Implementation variations are possible that may, in many cases, offer performance or other advantages. Given the same input key and data (plaintext or ciphertext), any implementation that produces the same output (ciphertext or plaintext) as the algorithm specified in this standard is an acceptable implementation of the AES.

11. SYNTHESIS RESULTS OF AES

11.1 SYNTHESIS INPUTS

11.1.1 VHDL or Verilog Code

The first requirement of any synthesis tool is HDL (Hardware Description Language) of your design. It may be in VHDL or Verilog format. Here AES (Advanced Encryption Standard) VHDL code is used as input to synthesis tool.

11.1.2 Synthesis Library File(LIB file)

It contains following information.

- Cell timing
- Cell functionality
- Cell netlist
- Design Rule Constraints

In this Design following Library used for Synthesis

Library Name	1) ECL065M2V0p6v1p0TD 2) REG
Technology	65 nm
No .of cells	53
Library Type	Standard Cell

Table 11.1 Library Information

11.1.3 Synopsys Design Constraints (.sdf)

Synthesis of any code requires timing constraints. Which include clock period, Input delay, Output Delay etc. In this Design following constraints are given to synthesis tool.

Clock Period	3.33 nsec
Input Delay	0.3 nsec
Output Delay	0.3 nsec

Table 11.2 Timing Constraints

11.1.4 Define Used Cell

Except all this files, depends on your requirement we can suggest tool that which cells to be used and which not from given .lib file. For this design out of 53 cells only 24 cells are given to tool for use in Synthesis procedure.

11.2 SYNTHESIS OUTPUT

11.2.1 Synthesized Netlist

Synthesis output is mainly contains synthesized netlist of given design code in VHDL or Verilog format. Here output netlist is in Verilog format.

11.2.2 Timing Reports

It Contains different timing results such as length of critical path in netlist, Clock periode in critical path, slack in critical path etc.This report is given in following table 11.3.

Levels of Logic:	27
Critical Path Length:	3.18
Critical Path Slack:	0
Critical Path Clk Period:	3.33
Total Negative Slack:	0
No. of Violating Paths:	0
No. of Hold Violations:	0

Table 11.3 Timing Report

11.2.3 Area Occupied

There is one another report which contains total area required by design cells. It doesn't contains area of routing required. It gives area in two categories. One is area required by Combinational Logic and another is area required by non-combinational logic. Addition of this two gives you total area.

Combinational Area:	14231.88
Non-combinational Area:	9406.8
Net Area:	0
Cell Area:	23638.68
Design Area:	23638.68

Total Number of Nets:	6338

Table 11.4 Total Area Occupied by design

11.2.4 Individual Cell Report

This tool also give information about each cell. Means which cell is used how many times in design, Its unit area as well as total area occupied by individual cell. This Results are shown in below table 11.5.

Sr No.	Reference	Library	Unit Area	Count	Total Area	Attributes
1	AO12X8	ECL065M2V0p6v1p0TD	4.68	22	102.96	
2	AO12X15	ECL065M2V0p6v1p0TD	4.68	10	46.80	
3	AOI12X5_	ECL065M2V0p6v1p0TD	3.12	416	1297.92	
4	AOI12X10	ECL065M2V0p6v1p0TD	4.68	2	9.36	
5	AOI22X5	ECL065M2V0p6v1p0TD	3.12	1542	4811.04	
6	AOI22X10	ECL065M2V0p6v1p0TD	6.24	7	43.68	
7	BFX15	ECL065M2V0p6v1p0TD	3.12	95	296.40	
8	BFX31	ECL065M2V0p6v1p0TD	4.68	4	18.72	
9	BFX62	ECL065M2V0p6v1p0TD	9.36	1	9.36	
10	D_FF	REG	14.04	670	9406.80	
11	IVX8	ECL065M2V0p6v1p0TD	1.56	455	709.80	n
12	IVX15	ECL065M2V0p6v1p0TD	1.56	570	889.20	
13	IVX31	ECL065M2V0p6v1p0TD	3.12	8	24.96	
14	MXX15	ECL065M2V0p6v1p0TD	6.24	3	18.72	
15	ND2X12	ECL065M2V0p6v1p0TD	3.12	322	1004.64	
16	NR2X6	ECL065M2V0p6v1p0TD	3.12	38	118.56	
17	NR2X11	ECL065M2V0p6v1p0TD	3.12	96	299.52	
18	NR2X22	ECL065M2V0p6v1p0TD	6.24	2	12.48	
19	NR2X33	ECL065M2V0p6v1p0TD	9.36	1	9.36	
20	OAI12X5	ECL065M2V0p6v1p0TD	3.12	847	2642.64	
21	OAI12X10	ECL065M2V0p6v1p0TD	4.68	3	14.04	
22	OAI12X19	ECL065M2V0p6v1p0TD	9.36	1	9.36	
23	XNRX10	ECL065M2V0p6v1p0TD	6.24	284	1772.16	
24	XR2X15	ECL065M2V0p6v1p0TD	7.8	9	70.20	
Total				5408	23638.68	

Table 11.5 Individual Cell Report

It also tells about cell attribute means cell is combinational or Non-Combinational type. Meaning of each attribute is given in following table 11.6.

By using this table we can plot following charts of comparison.

Fig 11.1 shows Unit area of Individual cell. From which we can analyze that Flip-Flop requires larger area out of all cells. And in case of Combinational cells BFX62, NR2X33 and OAI12X19 requires larger area out of all others.

Attributes	Meaning
B	black box (unknown)
Bo	allows boundary optimization
D	dont_touch
Mo	map_only
H	Hierarchical
N	Noncombinational
R	removable
S	synthetic operator

Table 11.6 Attribute Meaning used in Synthesis Report

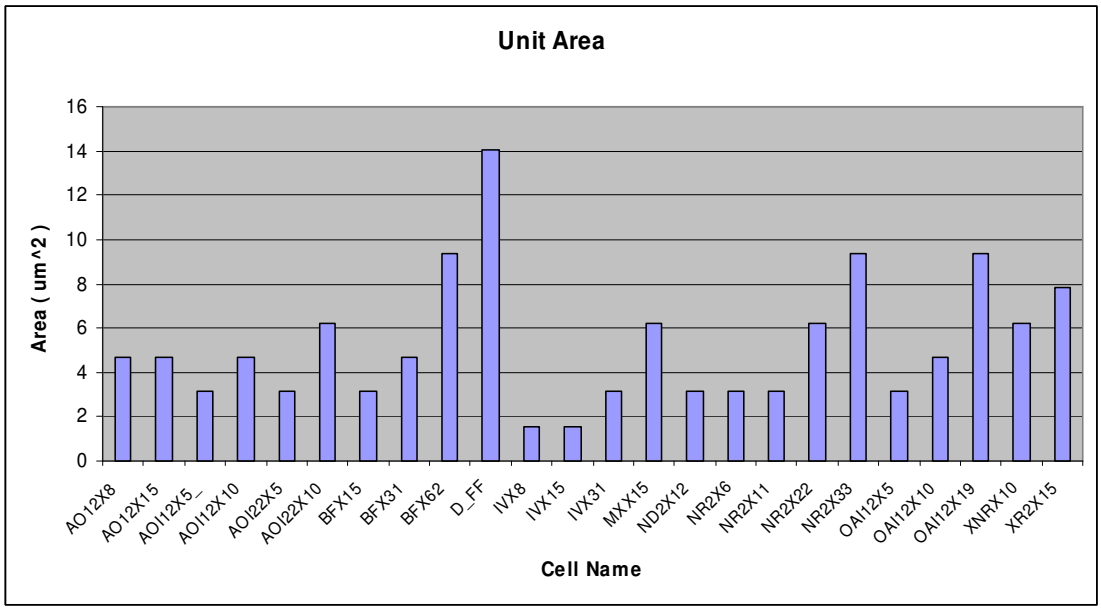


Fig 11.1 Comparison of Individual Cell Area

Whereas, Fig. 11.2 Contains Comparision of No. of cells used in design, which shows that AOI22X5 is most frequently used cell in library, which uses more than 1500 times in dsign.

Fig 11.3 shows total area occupied by Individual cell in Design. From which we can analyze that Flip-Flop requires larger area out of all cells. And in case of Combinational cells AOI22X5 and OAI12X5 requires larger area out of all others.

Fig 11.4 shows Combined plot of Normalized cell count and total area occupied by Individual cell in Design. From which we can analyze that area occupied by Flip-Flop is larger than area required by AOI22X5 cell, even if no. of cell of AOI22X5 is larger than required by Flip-flop. This is because that individual area of Flip-Flop is much larger than required by AOI22X5 cell.

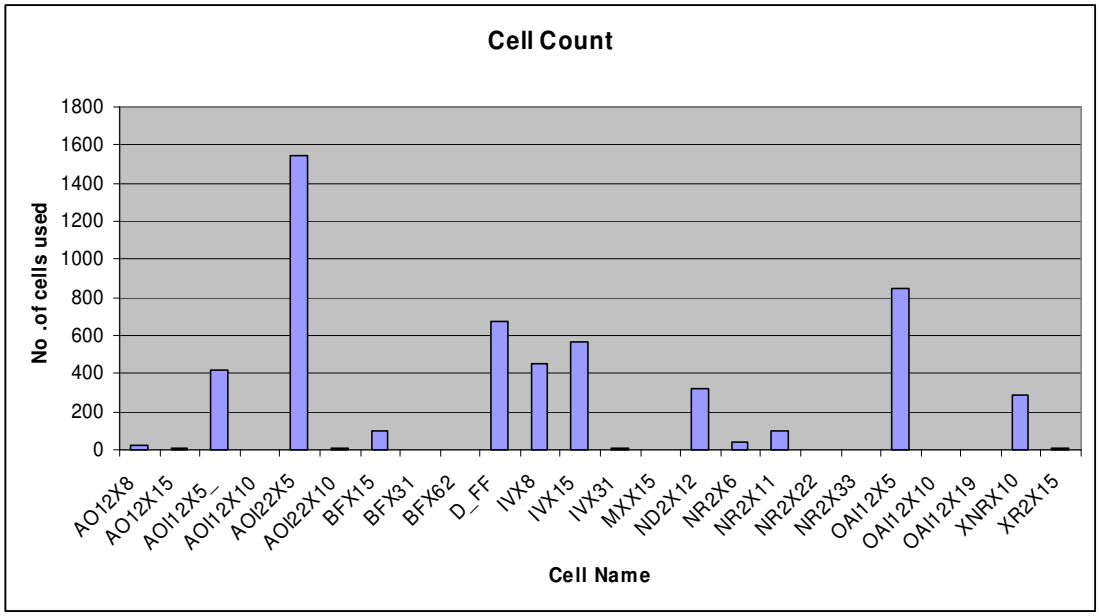


Fig 11.2 Comparison of Individual Cell Count Used

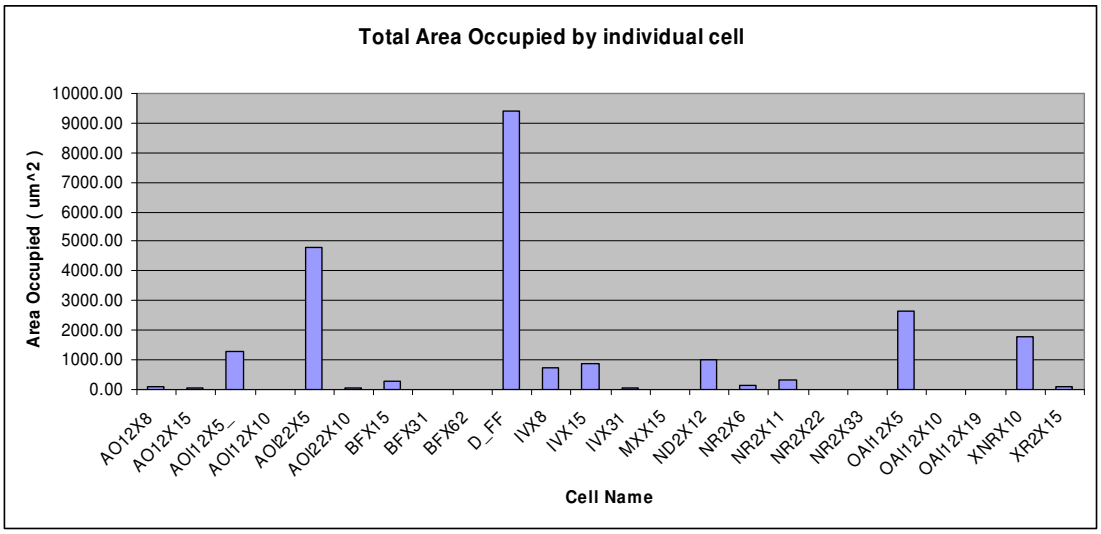


Fig 11.3 Comparison of Total occupied by Individual Cell in Design

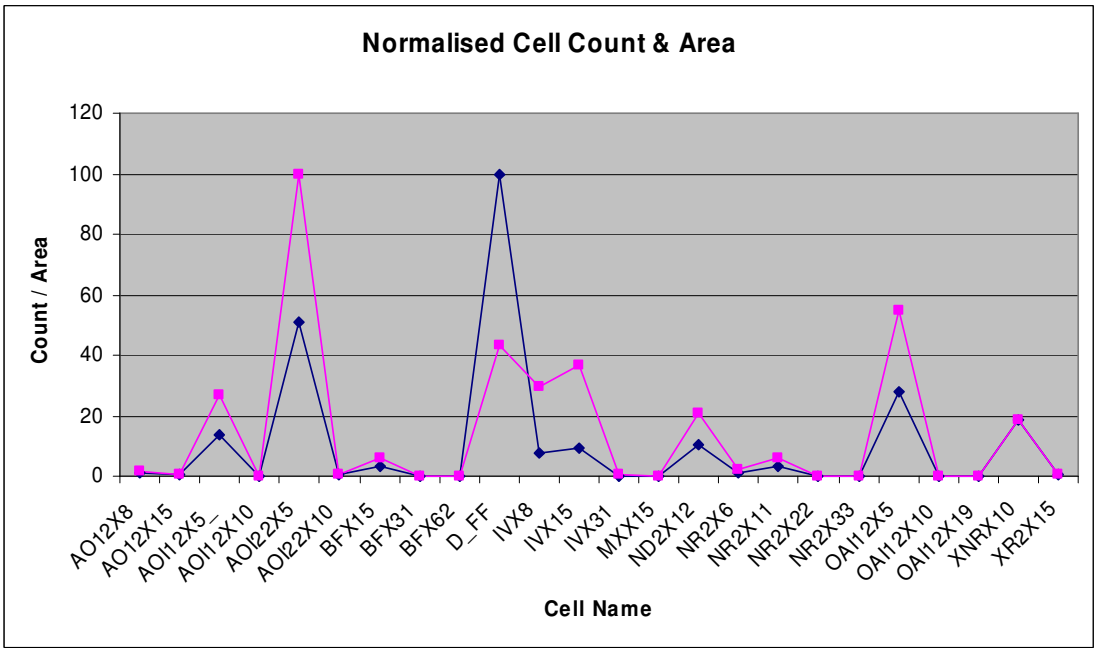


Fig 11.4 Comparison of Normalized Cell Count and Corresponding occupied area

12. PNR RESULTS OF AES

12.1 INPUTS TO PNR TOOL

12.1.1 Technology File(.tf))

Parameters used in PNR

RouteTypeName	CTSRoute
NonDefaultRule	DBLCUT_RULE
TopPreferredLayer	5
BottomPreferredLayer	3
PreferredExtraSpace	1
PostOpt	YES
RouteClkNet	YES
MaxSkew	180ps

12.1.2 Synopsys Design Constraints (.sdf)

It contains timing constraints and clock definition.

Clock Periode	3.33 nsec
Input Delay	0.3 nsec
Output delay	0.3 nsec

12.1.3 Reference Libraries

It contains .LIB (Library) or .DB (Database) file.

Both of this contains following information.

- Cell timing
- Cell functionality
- Cell netlist
- Design Rule Constraints

12.1.4 Netlist File

It is synthesized output of our design. It mainly contains synthesized in VHDL or Verilog format. It may be in .edf (Electronic Design Interchange Format). Here this netlist is in Verilog format.

12.1.5 Floorplan description File

It defines combinational and non-combinational area in total area. It also defines blockages in design. So it will gives guidelines to tool that at which place which type of cells should be placed.

12.1.6 Library Exchange Format (.LEF) file

It contains the units, drawing patterns, layers, design rules, vias, and parasitic resistance and capacitance of the manufacturing process.

12.1.7 GDSII – Files

It contains the physical layout information; to create reference libraries.

12.2 OUTPUT OF PNR TOOL

12.2.1 Synopsys Delay Format (.sdf)

It contains post-floorplaning timing.

Setup mode	all	reg2reg	in2reg	reg2out
WNS (ns)	0.004	0.287	0.004	3.122
TNS (ns)	0.000	0.000	0.000	0.000
Violating Paths	0	0	0	0
All Paths	1469	670	1228	129

12.2.2 Synopsys Parasitic Extract Format (.spef or .spf)

It contains all parasitic capacitance for different load condition.

12.2.3 Optimized Netlist (.v or .hv)

It is Optimized netlist after PNR. It may be in .v which is flattened type netlist or may be in .hv format which is hierarchical format.

12.2.4 Graphical Data Stream (.gds)

It contains updated gds format which is physical layout information.

12.3 OUTPUT DESIGN IMAGES:

In PNR first of all floorplan is decided. It contains information about where Combinational and non-combinational cells will be placed. In this design ,it is in '+' shape. Where in rectangle combinational cells will be placed and in other part, means in boarder of rectangle non-combinational cells will be placed. It also contains Obstacles , which will not allow any cell to place in that area.

Now as per your input netlist and floorplan this design is placed in your given area. Than according to given timing constraints this design is optimize and after multiple of iteration this design is finally placed. Floorplan view of this design given in fig. 12.1. In this figure blue color indicates placed cell and other is blank area.

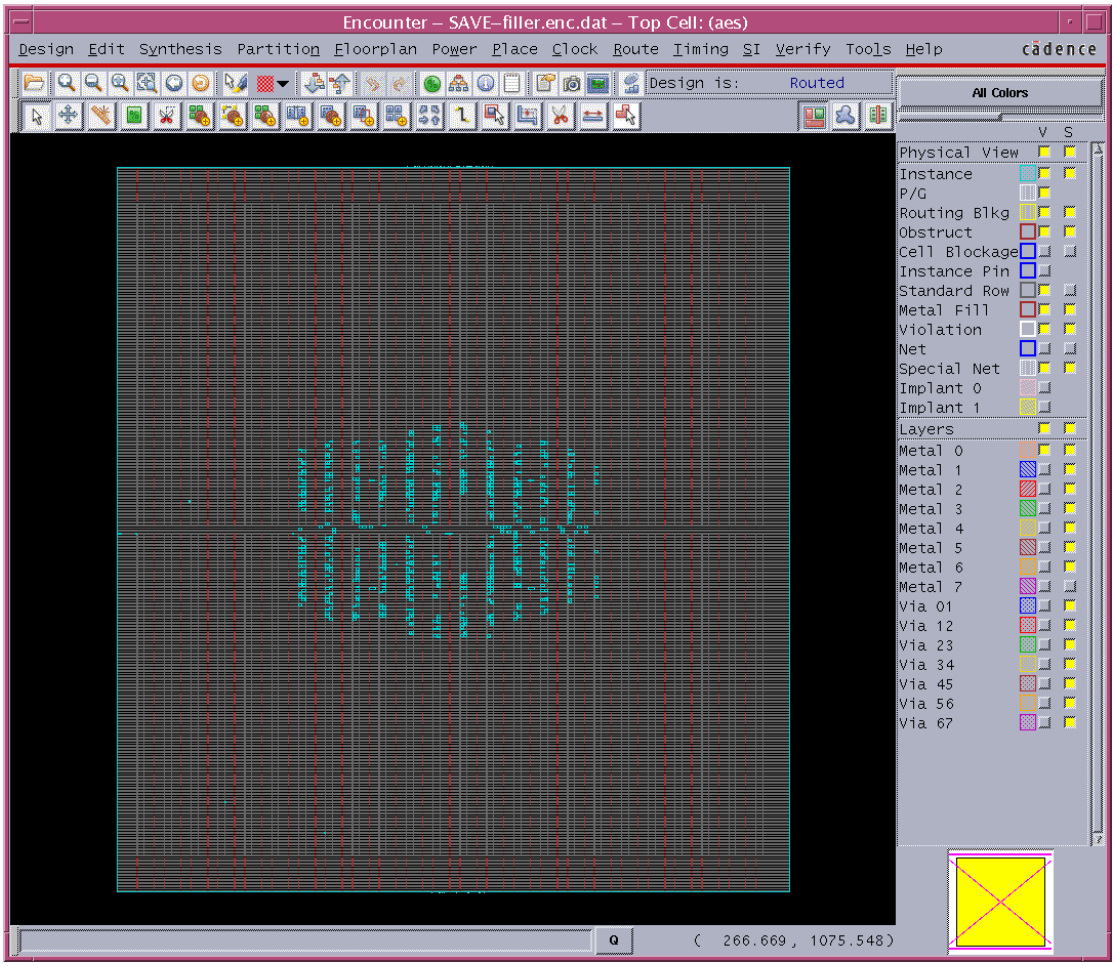


Fig 12.1 Placed Design Floorplan View

Fig 12.2 indicates placed designs view with metal layers. In this circled view is zoom in fig 12.3. In this figure it can see that cell is shown as blue colored. As well as it is also seen that it is connected with VDD and GND lines selected above and below it.

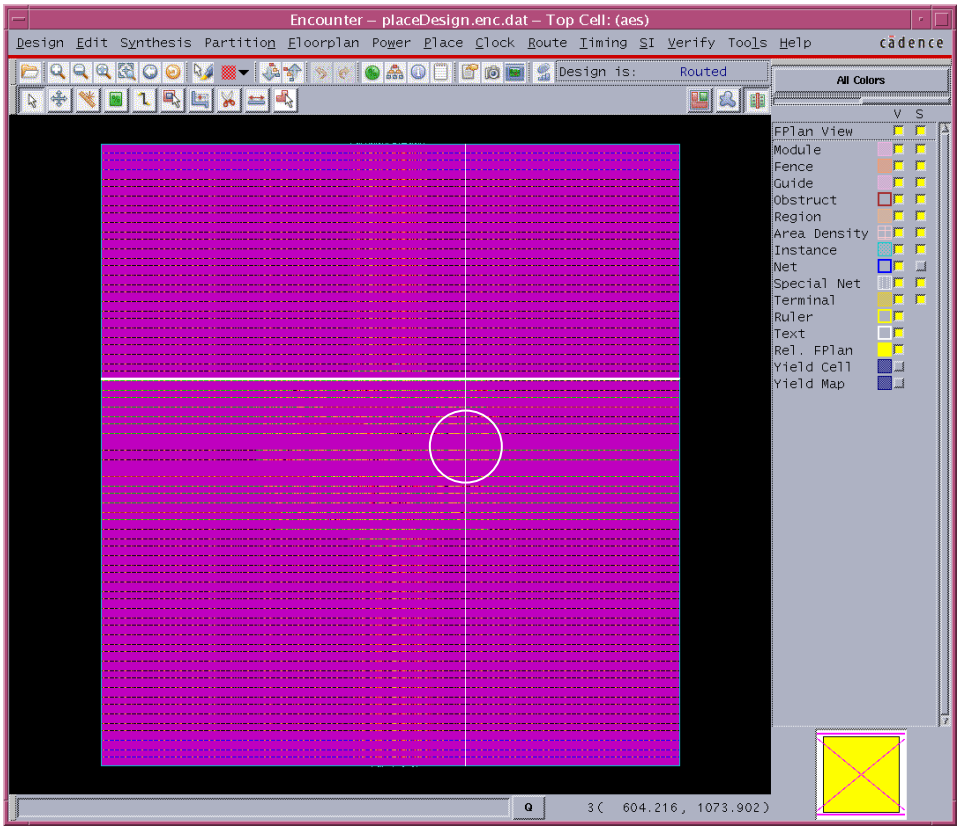


Fig 12.2 Placed Design With Routing Nets

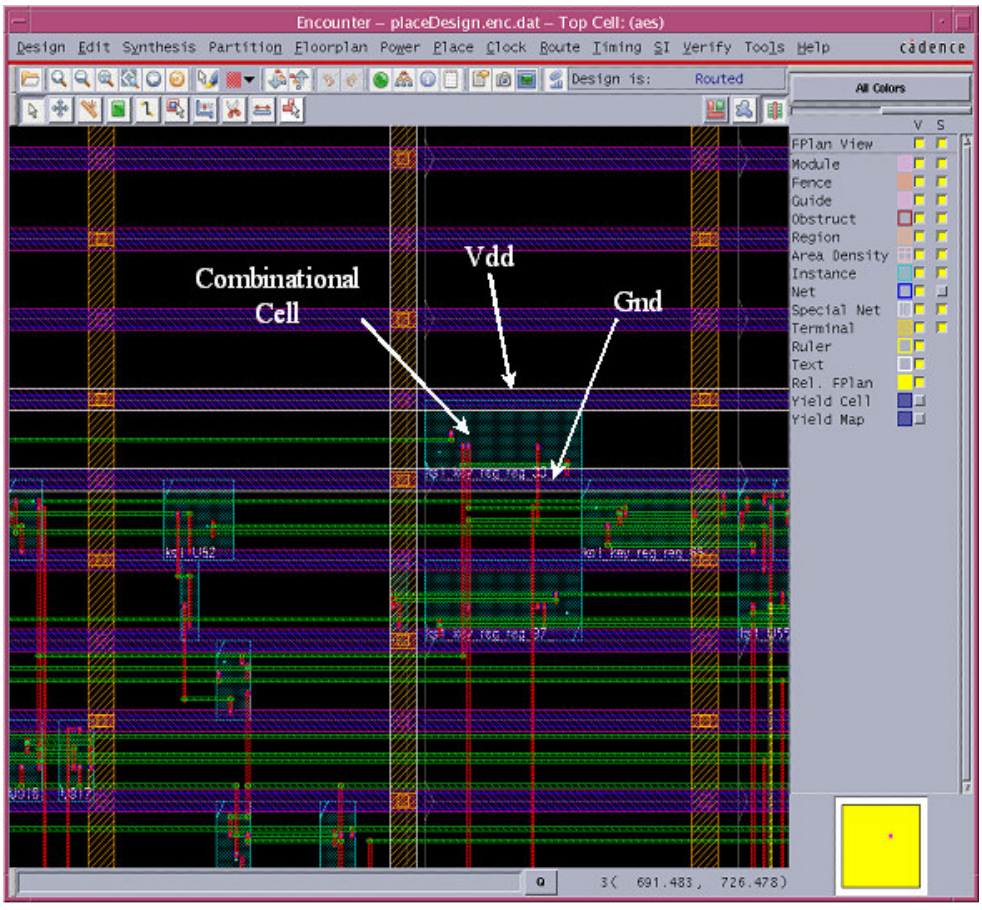


Fig 12.3 Zoomed Placed Design View

After placement and final routing of design fillers are added into the design. This filler might be a dummy cell in case of combinational area or may be dummy flip-flop in case of non-combinational area. Here cell contains pin on it whereas fillers doesn't have any pin information on it. So in this way filler cell can be separated from design cells.

Figure 12.4 shows physical view of design after adding filler to it. Here also same way you can identify filler cells from design cells. In this case you can identified it by using routing lines. Filler cells don't have any routing lines whereas design cells contain routing lines.

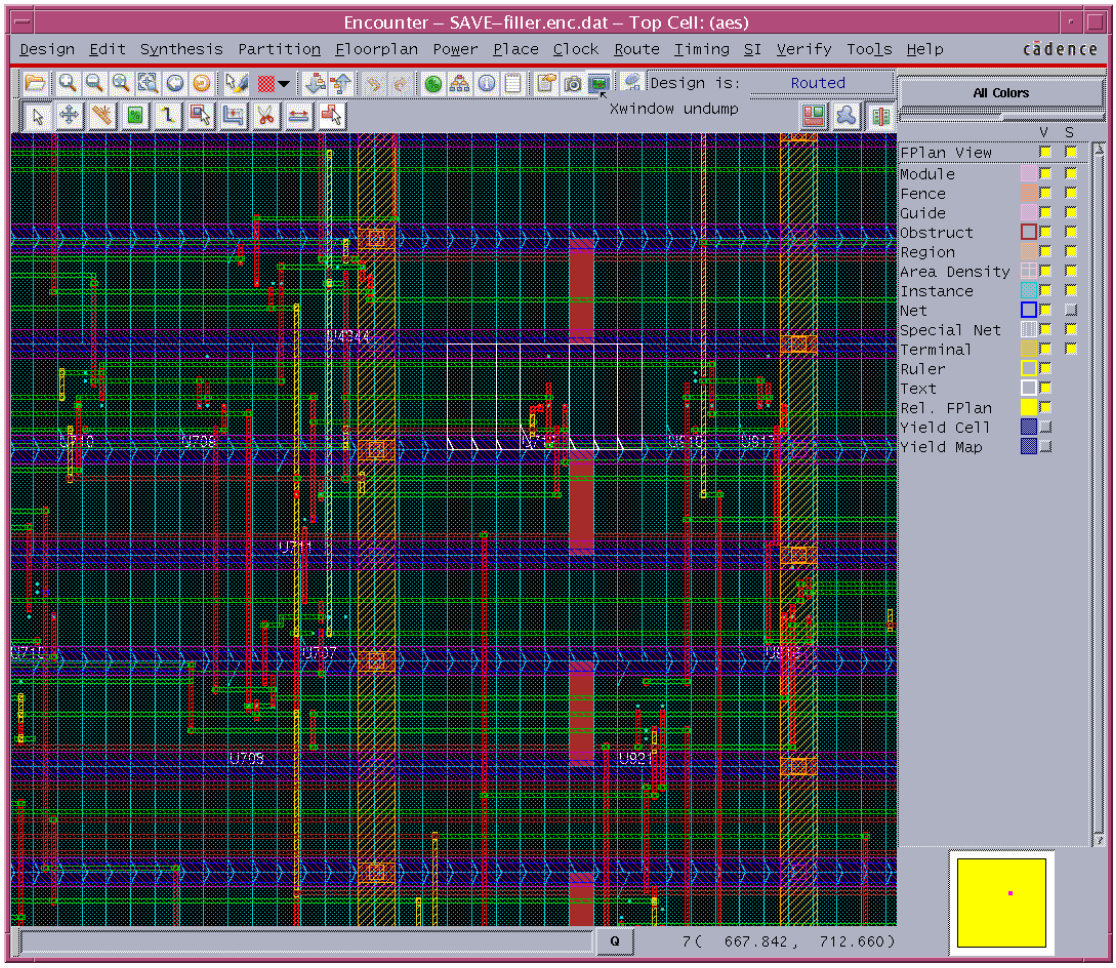


Fig 12.4 Netlist View after adding Filler

After completing PNR it will generates new optimized netlist. This netlist is shown in below fig 12.5.

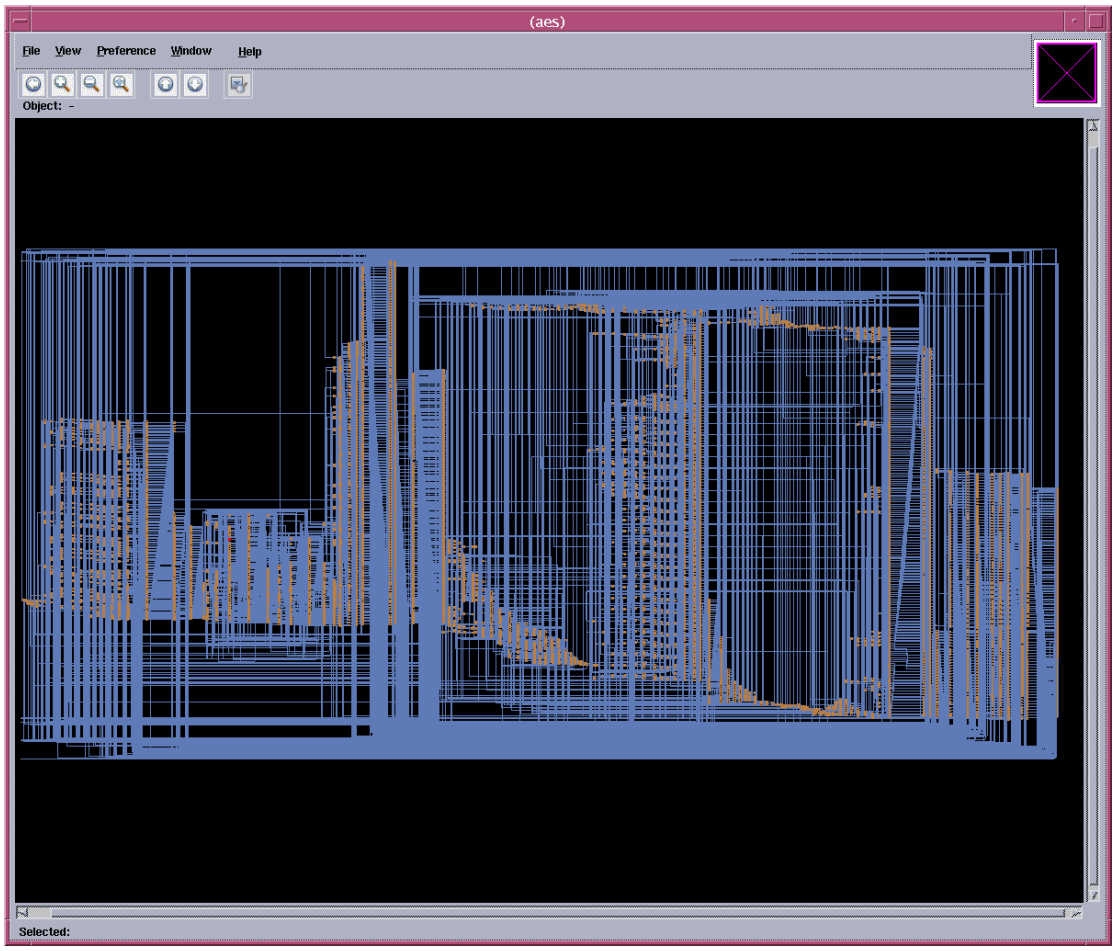


Fig 12.5 Output Optimized Net

13. CONCLUSION

This report contains information related to Structured ASIC type of semi-custom design style. This design style reduces design cycle time and cost of design. Here first standard cell library is prepared which further used in Synthesis and PNR. Other issues related to making layout are also discussed. These layouts are used to extract information about timing and area which will be used in Synthesis and PNR. This report also gives brief overview of Synthesis and PNR flow of any design using structured ASIC. Advanced Encryption Standard (AES) is more widely used in security systems. VHDL code of AES is used to demonstrate complete flow of Synthesis and PNR.

14. REFERENCES

1. Federal Information , Processing Standards Publication 197 November 26, 2001
2. A book on Gate to GDSII – Place and Route Methodology by Lee Eng Han
3. http://www.st.com
4. http://www.ccse.kfupm.edu.sa
5. http://www.micro.deis.unibo.it
6. http://www.primo.ismb.it.
7. http://www.cs.bc.edu
8. http://www.opencores.org
9. http://fg.gladman.plus.com
10. VLSI CAD Flow: Logic Synthesis, Placement and Routing by Srini Devadas