DESIGN AND CHARACTERIZATION OF A STANDARD CELL LIBRARY FOR THE FREEPDK45 PROCESS

By

NARESH ANNE

Bachelor of Engineering

Andhra University

Visakhapatnam, India

2008

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 2010

DESIGN AND CHARACTERIZATION OF A

STANDARD CELL LIBRARY FOR

THE FREEPDK45

PROCESS

Thesis Approved:

Dr. Chris Hutchens

Thesis Adviser

Dr. James E. Stine, Jr

Dr. Louis G. Johnson

Dr. Mark E. Payton

Dean of the Graduate College

ACKNOWLEDGMENTS

The continuous and unwavering support from many people made this thesis possible. Foremost, I would like to thank my mother who stood brave in educating us after the loss of my father. Also the support from relatives, siblings and friends is unforgettable. I express my sincere thanks to Dr. Chris Hutchens for being my advisor and being with me all the time patiently understanding and sharing his valuable knowledge and experiences. Without his review, the thesis wouldn't have been correct and complete. I express my gratitude to Dr. James Stine for giving an opportunity to learn by taking me into this project and supporting me. I can never forget the initial push and the support thereafter given by Dr. Louis Johnson. I thank all of them for being on my graduate committee. I would like to thank Farshad Dailami and Dan Liddell from Mentor Graphics for their support in learning the complex design tools. I extend my thanks to Dr. Stine's students Jun Chen and Ivan Castellanos whose contributions have simplified this work greatly. Also my thanks to all the lab-mates in MSVLSI lab Srinivasan, Rehan, Ran, Henry, Yong, Guanglei, Swati and friends like Redeat who made my Masters a wonderful experience.

GLOSSARY

Abbreviations:

ADC	-	Analog to Digital Converter
ALF	-	Advanced Library Format
ASIC	-	Application Specific Integrated Circuit
BSIM	-	Berkeley Short Channel IGFET Model
CAD	-	Computer Aided Design
CMOS	-	Complementary Metal Oxide Semiconductor
DAC	-	Digital to Analog Converter
DRC	-	Design Rule Check
EDA	-	Electronic Design Automation
ELC	-	Encounter Library Characterizer
FIR	-	Finite Impulse Response
GDS	-	Graphic Database System
GUI	-	Graphical User Interface
HDL	-	Hardware Description Language
IC	-	Integrated Circuit
IEDM	-	International Electron Devices Meeting
IGFET	-	Insulated Gate Field Effect Transistor
IITC	-	International Interconnect Technology Conference
IO	-	Input-Output
IP	-	Intellectual property
ITRS	-	International Technology Roadmap for Semiconductors

LEF	-	Layout Exchange Format
*.lib	-	Liberty
LVS	-	Layout Versus Schematic
MGC	-	Mentor Graphics Corporation
MIPS	-	Micro-processor without Interlocked Pipeline Stages
MOSFET	-	Metal Oxide Semiconductor Field Effect Transistor
MSVLSI	-	Mixed Signal VLSI lab
NCSU	-	North Carolina State University
OSU	-	Oklahoma State University
PC	-	PipeCleaner
PDK	-	Process Design Kit
P&R	-	Place and Route
QA	-	Quality Assurance
RAM	-	Random Access Memory
*.sdc	-	Synopsys delay constraint
SOC	-	System On Chip
SPICE	-	Simulation Program with Integrated Circuit Emphasis
VCAG	-	VLSI Computer Architecture Group
VHDL	-	Very High Speed Integrated Circuit Hardware Description Language
VLSI	-	Very large scale integration

Parameters:

C_d	-	Diffusion node capacitance
C_{in}	-	Input Capacitance
C_{out}	-	Output Capacitance
C_{ox}	-	Gate oxide capacitance
d, $t_{d,}$ t_{pd}	-	Propagation delay
f	-	Effort delay
g	-	Logical effort

g_x	-	Horizontal grid
g_y	-	Vertical grid
h	-	Fan-out
p	-	Parasitic delay
t_d, t_{pd}	-	Propagation delay
T_s	-	Input slew or rise/fall time of inputs
V_{th}	-	Threshold voltage at which the outputs are measured

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

CHAPTER I

INTRODUCTION

Mentor Graphics Corporation (MGC) and VLSI Computer Architecture Research Group (VCAG), Oklahoma State University (OSU), started the PipeCleaner (PC) project. PC, an application specific integrated circuit (ASIC) represents some of the most commonly used blocks in many integrated circuit (IC) applications today. In this chapter an overview of the ASIC's components, its functionality and some of the applications MGC's intended usage are presented. A brief description of the FREEPDK45[1, 2], a non-proprietary 45 nm process on which the design is based is given. The design that was completed previously needed some improvements which are also discussed. This thesis is about optimization and characterization of the existing cell library to re-implement the PC-ASIC. The design is intended for use with testing the Electronic design automation (EDA) tools by MGC and cannot be fabricated. However, the procedures established for characterizing the cell library can be used as a reference for any other cell library.

1.1 Components and functionality of the PC-ASIC:

The ASIC contains the following components:

1. A digital finite impulse response (FIR) filter implemented with a Microprocessor without Interlocked Pipeline Stages (MIPS) like processor.

2. Random Access Memory (RAM), for FIR filter calculations and coefficient storage.
3. 6-bit Analog to Digital Converter (ADC).
4. 6-bit Digital to Analog Converter (DAC).

The function of the IC can pretty well be understood from its components. The input is an analog signal, which is digitized by the ADC and fed to the processor. The processor filters this signal and gives it back to the DAC, where it is again converted back to an analog signal and output. The filter coefficients are stored in the memory which is also used by the processor for the storage of any intermediate results.

1.2 PC-ASIC & MGC:

To effectively demonstrate their EDA (Electronic design automation) tool capabilities such as software quality or solving specific customer problems, MGC needed an ASIC which reflects real world user designs. Although the company gets use-case data from many of the users for doing case studies, it cannot be shared universally or used to develop debugging tutorials due to intellectual property (IP) law violations. Hence, it is proposed to design the PC-ASIC using a non-proprietary and foundry-agnostic process.

Some of the purposes, MGC states it wants to use the ASIC for are listed below:

1. Example Kits: The PC-ASIC is used as input data for example kits. The example kits are developed for demonstrating specific capabilities of the tools like physical verification. The IC provides a framework for the user to try out the tasks and explore the tools.
2. Training Material: The IC is used to develop training materials for the internal staff as well as for the users.
3. Quality Assurance (QA) checks: The example kits can be used for QA check during the testing and verification phase of the software release cycle. Since, the IC

represents a real world design; it provides a good foundation for quality checks of the tools.

4. Customer Assistance: This is an area that's being worked on. If a user cannot provide data to the customer service personnel, the teams can use the PC-ASIC data to reproduce a problem that the customer is experiencing and solve them.

1.3 FREEPDK45:

FREEPDK is a 45 nm open-source process with much of the contributions coming from North Carolina State University (NCSU) and Oklahoma State University (OSU) for Very large scale integration (VLSI) research and education purposes with MGC being one of the sponsors. It is distributed under the open-source Apache License and may be freely used and modified. The technology is intended to work with BSIM4 predictive technology model. BSIM stands for Berkeley short channel IGFET (insulated gate field effect transistor) model. It addresses the physical characteristics of a Metal oxide field effect transistor (MOSFET) in the designs below 100 nm. The Process design kit (PDK) includes:

- Technology library
- Tech files and display resources
- Design rules compatible with Calibre
- Standard cell library[1, 2]
- Memory compiler[1, 2]
- Analog blocks ADC and DAC[1, 2]
- Setup file for characterization of the standard cells[3-6]
- Design flow for place and route developed by [J. Chen, I. Castellanos and J. Stine]

The commercial PDKs have IP restrictions on using them for academic purposes and classroom instruction. Also it's a very difficult procedure for the academic institutions to maintain the

design flow for the EDA tools to implement an IC. FREEPDK resolves the problem by being open-source and including the scripts for design flows within the PDK [1, 2].

Scaling down the devices to nanometer level has brought in many design challenges, variation being the most dominant factor. The variability in the physical parameters i.e. doping concentrations, width, length and many others at the nanometer regime affects the electrical characteristics of the devices to a great extent. Care has to be taken to understand these effects on designs in order to obtain desired yields. FREEPDK aims at including this variation awareness into it, so that these design intricacies are accurately modeled in order to support their understanding by novel users, students before going into the industry. The minimum feature size and amount of variation to be expected are developed with references to the ITRS and conference publications such as IEDM and IITC[7].

FREEPDK thus allows the student researchers to test and validate their designs, the different architectures, System on chips (SOCs). The design kit can be downloaded by typing the following command at the user's Linux terminal:

svn co https://svn.unity.ncsu.edu/osi/freepdk45/trunk .

This will download the PDK into the user's current directory. However, the user needs to have the EDA tools installed in his Linux machine and have internet access. Useful information on the latest news, design rules for the PDK can be found at

http://www.eda.ncsu.edu/wiki/FreePDK45:Contents .

1.4 Proposed changes for the existing PC-ASIC:

1. The standard cell library included with the PDK is generated automatically from a software tool Cadabra, from Synopsys, Inc. The cells generated had long poly

interconnects and some unnecessary layer extensions. These needed to be optimized for better performance.

2. The cells are to be characterized including all device and interconnect parasitics.
3. The previously existing PC-ASIC had the digital core, analog blocks and Input/output (IO) pads connected to a common power supply. It is proposed to have isolated power domains in order to observe and determine the tools' capability to work under such a situation. Doing layout verses schematic (LVS) checks after the place and route of the entire chip is quite challenging when we have different power nets.

In addition, there were several existing layout problems. The poly interconnects in the existing cell library were long. Their excessive lengths made them difficult to understand, and rework for greater cell tightness. Cells with larger drive strength had wide PMOS transistors and laid out as a single transistor without the use of fingers. The poly runs over the active diffusion areas were very long. After some investigation, it was also found that the cell height could and should be reduced as well. Hence it was proposed to redo the standard cell library manually.

1.5 Tasks accomplished:

The cell library was reworked manually with the changes noted above, abstracted and characterized including the parasitics. The PC-ASIC is re-implemented or laid out using this new library and with separate power domains for the IO, core digital and analog blocks. The IC has been physically verified for design rule check (DRC) and LVS errors. The design flows for doing the characterization and place and route have been obtained along with the PDK [1, 2].

CHAPTER II

STANDARD CELL LIBRARY DESIGN

2.1 Introduction

Modern integrated circuits are very large containing over a billion transistors. For example a 5mm x 5mm chip may contain over a 750 million transistors or approximated at 30 transistors per um^2. This makes it clear that the job can be done only with the help of computer aided tools (CADs). Designs are completed and tested at a higher level using a *hardware description language* (HDL) like *Verilog* or *Very High Speed Integrated Circuit HDL (VHDL)*. Implementation at the device level is achieved by conversion to an interconnection of simple well defined blocks much like the Lego toys. These blocks are referred to as the standard cells. Since the interconnection is made by an automated tool, the blocks have to be designed following a specific rule set in order to achieve a final chip that is DRC/LVS (Design rule check/ layout vs. schematic) clean. In this chapter, we will take a close look at many of the rules needed to specify the standard cell format to produce error free designs.

2.2 Standard cell library

The Standard cell library describes a list of cells that a synthesizer may use to implement error free designs. The cell library information required to implement an ASIC design are summarized on next page:

1. Circuit schematics, layouts, circuit netlists (from schematics), and parasitic extracted netlists (from layouts). In this work, the schematics and layouts are designed using the Virtuoso tool from Cadence; netlists are extracted using Calibre Interactive from Mentor.
2. Abstracted views of each of the standard cell. The abstract view of a cell contains the bounding dimensions of the cell, routing obstructions and pin locations. These are required for placing and routing of the final chip by the Place and Route (P&R) tool. Through a process called Abstraction using the Abstract tool from Cadence, we get these views. The information is described in a text file in LEF-format.
3. The timing, power, functionality, and operating conditions should be given in a standard industry file format, Synopsys Liberty format (or *.lib file). This data is used by other design tools like synthesizer and the P&R tool.
4. Verilog/VHDL models for all the cells.

Figures showing this information for an example Inverter cell are shown below.

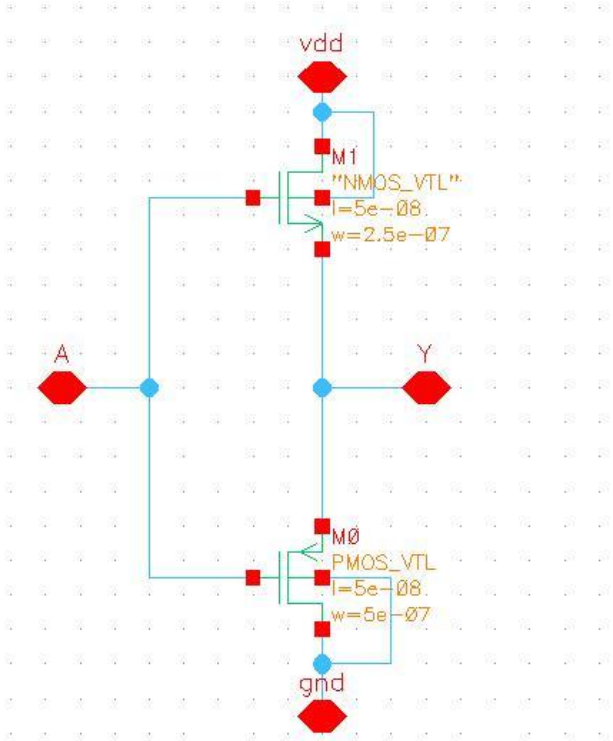


Fig 2.1: Inverter Schematic

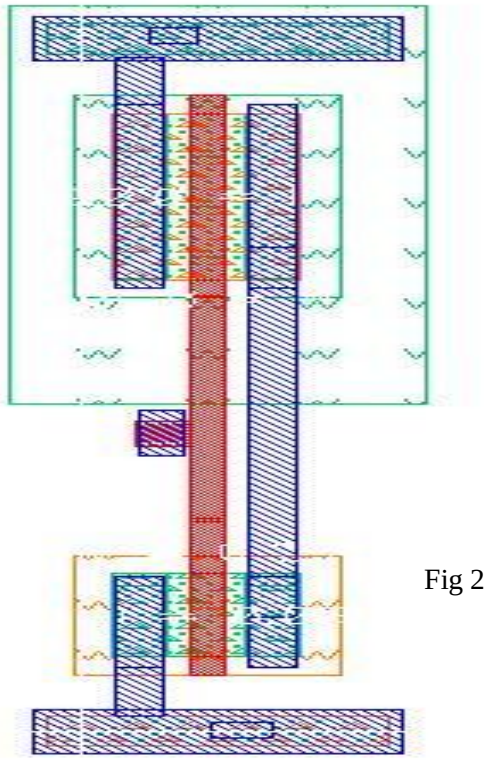


Fig 2.2 Inverter Layout

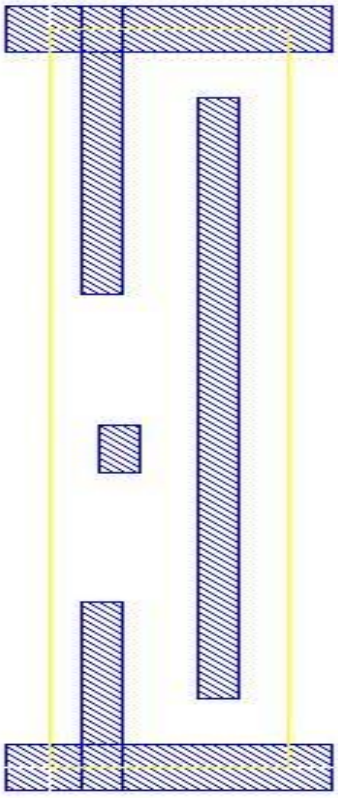


Fig 2.3 Inverter Abstract View

```
.SUBCKT INVX1 A Y gnd vdd
*.PININFO A:B Y:B gnd:B vdd:B
MM0 Y A vdd vdd PMOS_VTL W=5e-07 L=5e-08
MM1 Y A gnd gnd NMOS_VTL W=2.5e-07 L=5e-08
.ENDS
```

Fig 2.4: Circuit netlist Inverter (Schematic)

```
.SUBCKT INVX1  A GND VDD Y
MM1 N_Y_MM1_d N_A_MM1_g N_GND_MM1_s GND__2 NMOS_VTL L=5e-08 W=2.5e-07
+ AD=2.625e-14 AS=2.625e-14 PD=7.1e-07 PS=7.1e-07
MM0 N_Y_MM0_d N_A_MM0_g N_VDD_MM0_s N_VDD_MM0_b PMOS_VTL L=5e-08 W=5e-07
+ AD=5.25e-14 AS=5.25e-14 PD=1.21e-06 PS=1.21e-06
.include "INVX1.pex.netlist.INVX1.pxi"
.include "INVX1.pex.netlist.pex"
.ends
```

Fig 2.5: Netlist with parasitic information Inverter (Layout)

Fig 2.4 and fig2.5 show the netlists extracted from the inverter schematic and layout respectively. The schematic netlist contains the net connections, *width* and *length* parameters of the transistors whereas the layout netlist includes the *source* and *drain diffusion areas* and *perimeters* as well. The two files included in the layout netlist in fig2.5: "INVX1.pex.netlist.pex", "INVX1.pex.netlist.INVX1.pxi" have details of interconnect resistances and capacitances respectively. The netlists are used for the following purposes:

1. Schematic netlist - LVS.
2. Layout netlist - Timing extraction.

Table 2.1 lists the standard cells in the FREEPDK Library.

Table 2.1: List of Standard Cells

No.	CELL NAME	FUNCTION
1	AND2X1	Y = A.B
2	AND2X2	Y = A·B
3	AOI21X1	Y=NOT(A·B + C)
4	AOI22X1	Y=NOT(A·B + C·D)
5	BUFX2	Y = A
6	BUFX4	Y = A
7	CLKBUF1	Y = A
8	CLKBUF2	Y = A
9	CLKBUF3	Y = A
10	DFFNEGX1	D-Type Flip-flop with negative edge clock
11	DFFPOSX1	D-Type Flip-flop with positive edge clock
12	DFFSR	D-Type Flip-flop with positive edge clock and negative SET and negative RESET
13	FAX1	YS = A ⊕ B ⊕ C YC = A·B + B·C + C·A
14	FILLCELL_1,2,4,8,16	Filler Cells
15	HAX1	YS = A ⊕ B YC = A·B
16	INVX1	Y = NOT(A)
17	INVX2	Y = NOT(A)
18	INVX4	Y = NOT(A)
19	INVX8	Y = NOT(A)
20	LATCH	D-Type Latch with positive clock level
21	MUX2X1	2 to 1 Multiplexer
22	NAND2X1	Y = NOT (A·B)
23	NAND3X1	Y = NOT (A·B·C)
24	NOR2X1	Y = NOT(A+B)
25	NOR3X1	Y = NOT(A+B+C)
26	OAI21X1	Y = NOT((A+B) ·C)
27	OAI22X1	Y = NOT((A+B) ·(C+D))
28	OR2X1	Y = NOT(A+B)
29	OR2X2	Y = NOT(A+B)
30	TBUFX1	Y = A·EN; Y = HiZ for(NOT E)
31	TBUFX2	Y = A·EN; Y = HiZ for(NOT E)
32	XNOR2X1	Y = NOT (A ⊕ B)
33	XOR2X1	Y = A ⊕ B

The cells can be categorized as follows:

1. Logic Cells implementing Boolean logic.
2. Latches and Flip-flops to implement the state storage etc.
3. Clock Buffers.
4. Buffers/Inverters.

As we can see from table 2.1, the cell name is followed by the number of inputs it has. There are cells ending with X1, X2 etc. These implement the same logic function but have relative drive strengths of 1 and 2 respectively. Also there are three types of buffers: regular buffers (BUF), clock buffers (CLKBUF) and tri-state buffers (TBUF). Let us understand why we need them.

2.3 Drive Strength:

When the synthesizer compiles the design from a behavioral description into a collection of standard cells, often large nets are created, i.e. many gates connected to a single net. The greater the number of devices (gates & interconnects) connected to a net, the greater drive strength required to sustain maximum clock speed. This also happens as the Place and route tool wires in the standard cells. As a result, we need to design cells with a wide range of buffer drive strengths in the library, in order to support the synthesizer and P&R tool in the optimal buffering of large nets. A 2X cell drives twice the load driven by 1X cell, a 4X twice that of a 2X and so on. The 1X load is the load presented by the minimum sized inverter in the cell library. The dimensions of INVX1 in this library are:

PMOS_VTL W=0.5e-06 L=0.05e-06

NMOS_VTL W=0.25e-06 L=0.05e-06

Cox for this process $= 26.67$ fF/um^2 (obtained from simulation)

$$C_{load1x} = (0.5 * 0.05 * 26.67) + (0.25 * 0.05 * 26.67) = 1\ fF\ (approx)$$

If the net is larger than what the cells in the library can drive, the compiler breaks them into smaller nets using buffering to drive each branch.

2.4 CLOCK Buffers and Clock Tree Synthesis:

The clock signal is another very large net that regularly requires very high drive currents. These nets are broken down into simpler nets and designed as a tree as shown in fig2.6[8]. But, there are skew problems which are frequently made worse just by adding buffers. Skew is the time difference between the clock signals arriving at different clocked elements in a chip. Fig 2.6 just gives an idea of how big nets are broken into simpler ones.

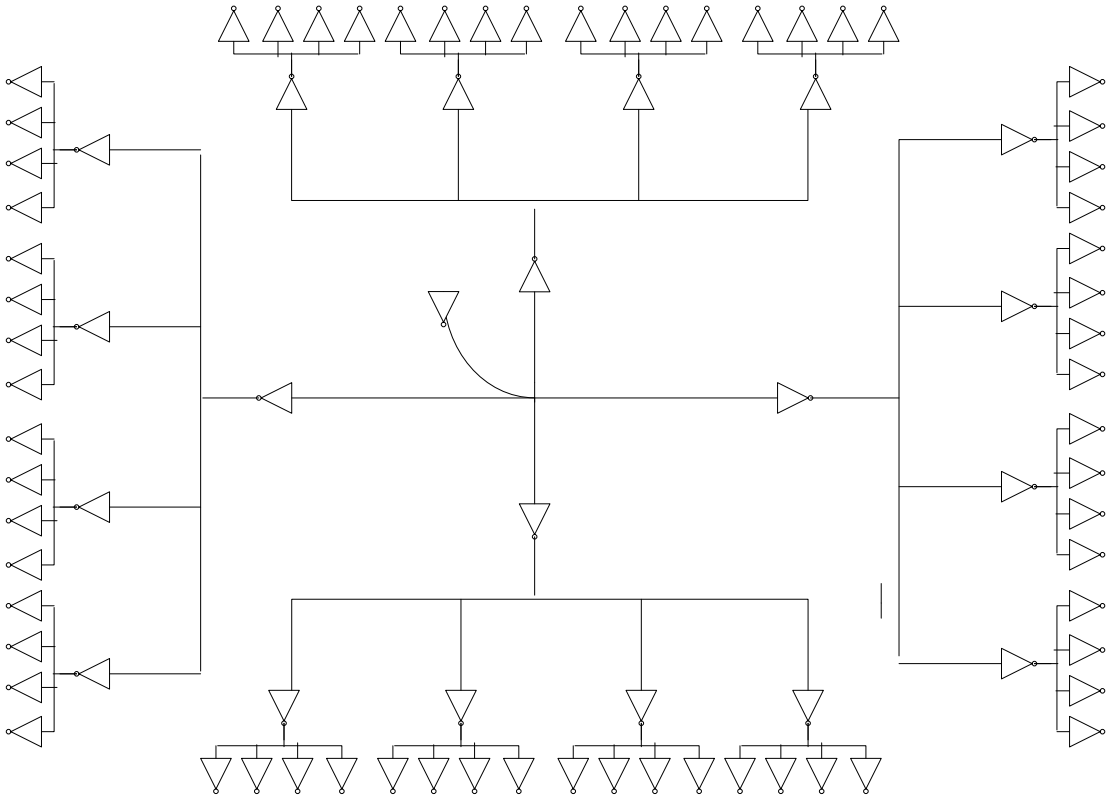


Fig 2.6: Clock Tree

The Place and Route tool automatically designs a balanced clock tree, and then adjusts the buffers depending on the load a particular clock net sees to partially compensate the skew differences. For this effort the library needs specific cells referred to as the clock buffers, CLKBUF in our

library. CLKBUF buffers have high drive strength and near equal rise/fall time delays and noise margins (not exactly equal because the PMOS are scaled only by 2 instead of beta which is approximately 3). Also the transistors are made wider to reduce the variation (*threshold mismatch*, greater device area lessens mismatch[9]) among the different buffers in the clock tree and hence the skew. The global clock drives the clock distribution network, which in turn drives the individual functional blocks. The main sources of clock skew are summarized below[10]:

1. A random skew that occurs because of the manufacturing variations which affect transistor parameters like *width, length*, *threshold voltage*, *oxide thickness* and the *wire width* and *thickness*.
2. A systematic skew due to the differences in the load seen by different branches of the clock distribution network.
3. Drift in parameters due to temperature gradient also affects the delay and hence the skew.
4. High frequency environmental variations like power supply noise cause jitter that leads to variation in delay for the clock buffer branch causing skew.

2.5 Design rules for the Standard Cells:

In the Standard cell based design flow, after the behavioral code is synthesized into a collection of standard cells, the place and route tool, SOC Encounter places them in sets of rows. Since the process is automated it follows specific rules to make routing simple and correct by construction. Wiring between the standard cells is done by the router following a grid. The tool routes the wires only on the grid as shown in fig2.7 except at the m1 level. The standard cells must also have dimensions that are a multiple of the grid value. The tool places the standard cells such that their origin lies on the grid points. To have the power rails abutted without any DRC errors, the height and width of all the cells should be multiples of the grid value.

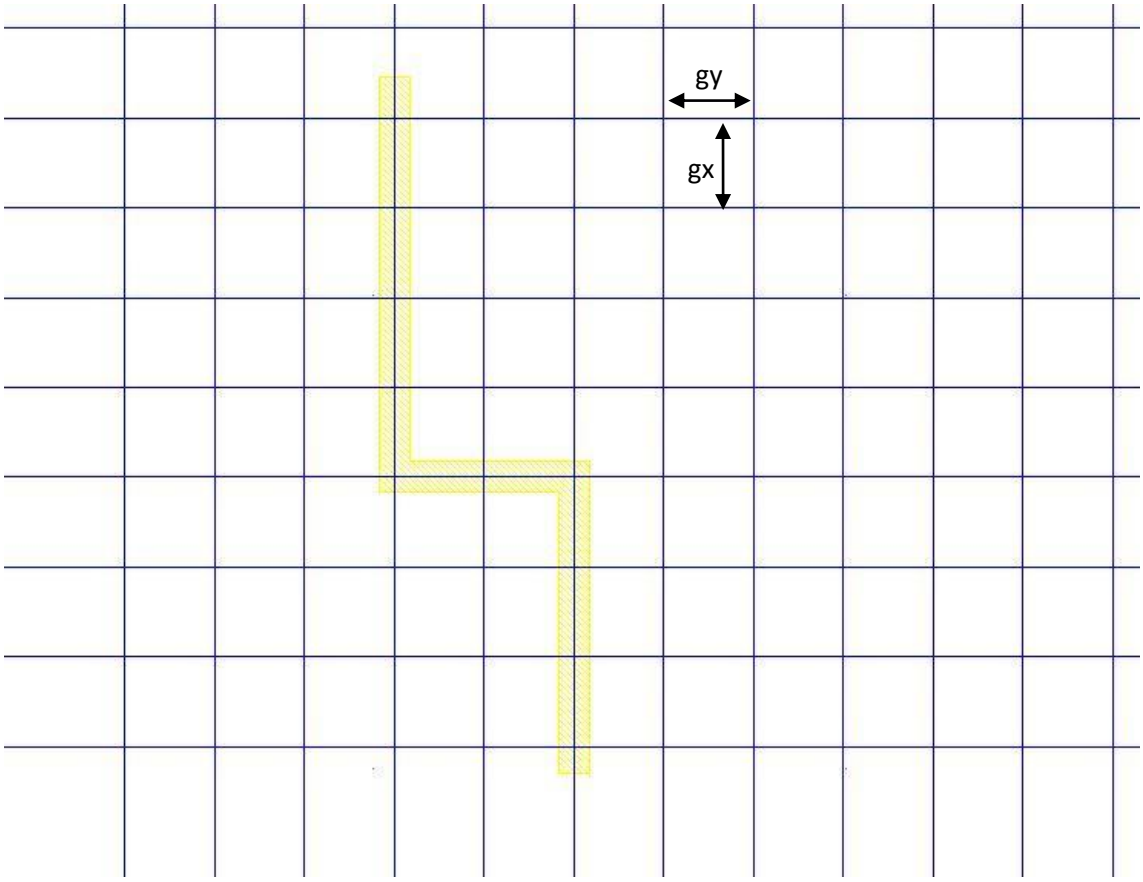


Fig 2.7: Routing is done by following the grid. gx and gy are horizontal and vertical grid values.

2.5.1 Grid:

Fig 2.7 demonstrates library cell gridding. gx and gy in the figure represent the horizontal and vertical grid spacing/values respectively. *The values are obtained based on how close two identical wires can be placed while, accommodating via to via switching between horizontal and vertical layers and minimum area for each metal layer without creating DRC errors*. Fig 2.8 gives an idea of selecting the appropriate grid size. It shows four M2-M1 vias in the FREEPDK process. The minimum spacing at which they can be placed without a DRC error gives the grid value. Often to avoid confusion both the horizontal and vertical grid spacings are chosen to be the same. The grid value used for FREEPDK project is 0.19 um.

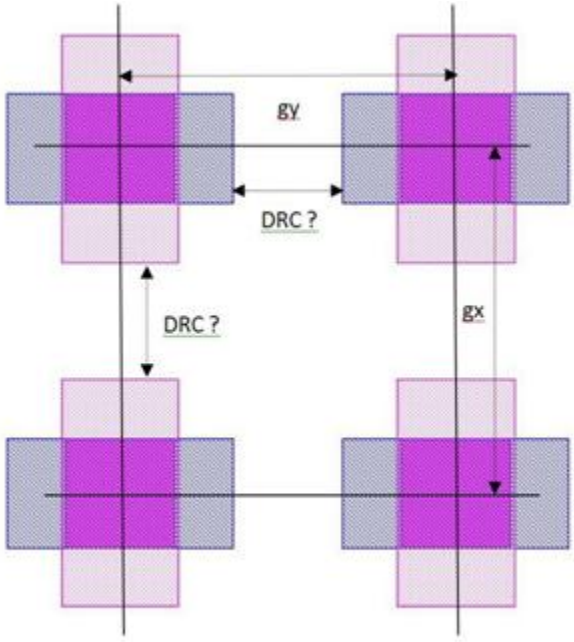


Fig 2.8: Selecting grid value

2.5.2 Horizontal and Vertical layering technique:

Routing the metal layer haphazardly creates dead-ends and results in some nets that are land locked resulting in larger chip area, longer interconnects and adds significant parasitics increasing delays and power and reduces the bandwidth of the chip. A simple rule followed from the initial stage of designing with standard cells to the final routing of the chip is:

- Metal 1 freeform within the cell and horizontal for the power rails, VDD and VSS.
- Even metal layers 2, 4, ..., 2n always vertical.
- Odd metal layers 3,5, … always horizontal.

It's not necessary that one should follow the same pattern but it should be noted that alternate metal layers always run perpendicular to each other. In this way, it is ensured no metal layer blocks a routing path. When designing mixed signal ASICs, the pattern should be communicated to all the members in the design team to avoid any problems when stitching together the chip. In

order to connect between the layers vias are used. For example, as shown in fig2.9 to connect points A and B, we go horizontally in metal1 and then switch to metal2 that runs vertically using a via.

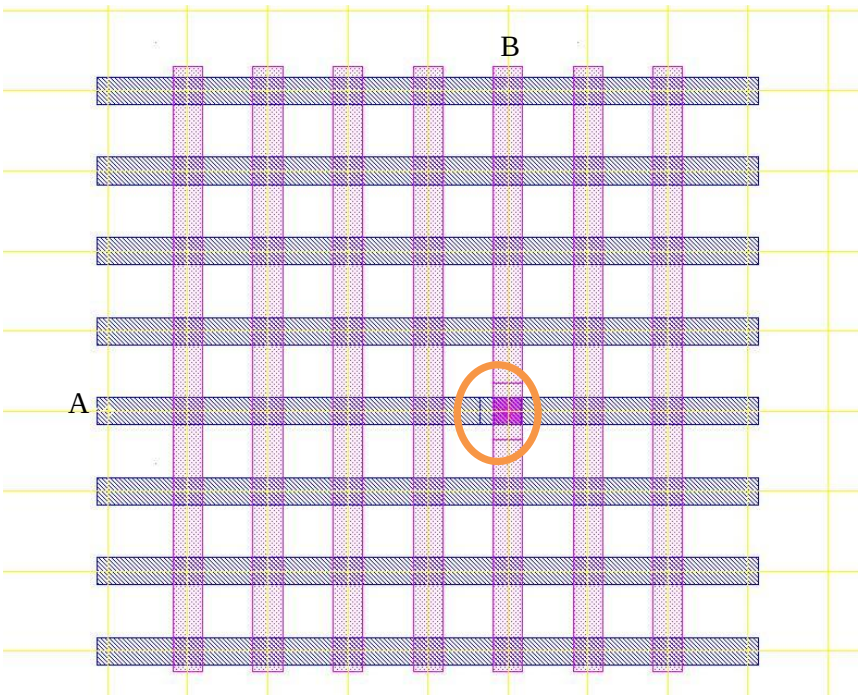


Fig 2.9: HV grid

By following this directional technique we are accommodating extra wires horizontally and vertically thus being able to connect more nets within the same area. This results in the total wire run length between two nets being as short as possible, reducing parasitic delay and power consumption.

Observation:

Do we have to strictly follow the direction rules always? The answer depends on the situation. If the jump is only one or two grids and we are not going to route any extra wires within that space we may continue with the same layer without switching. Fig 2.10 shows a situation which shows the switching between layers is a bad choice. By switching to metal2 for the short jumps we are reducing the reliability as well as blocking the area for any vertical wiring that may be come from above.

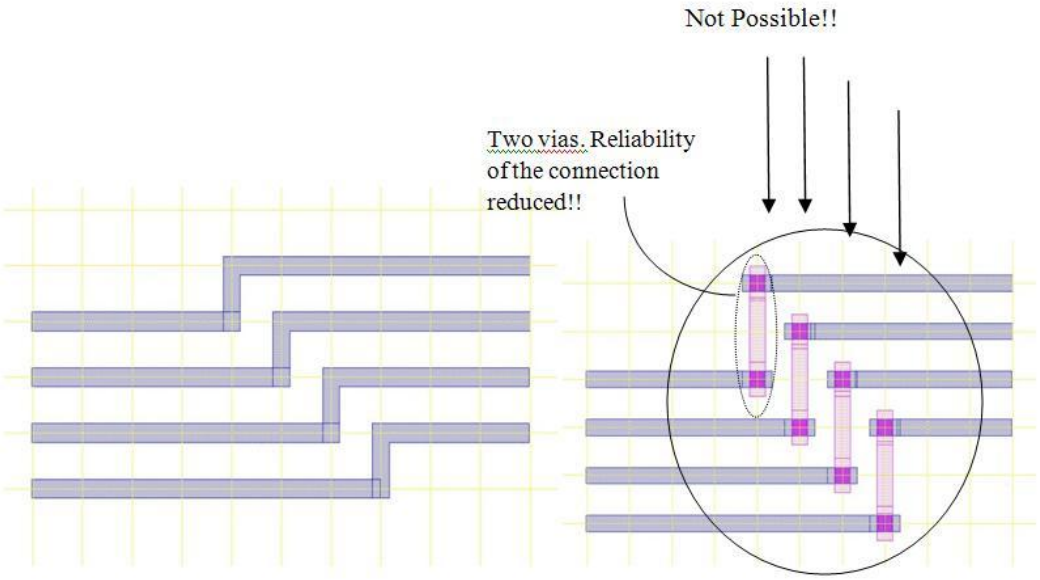


Fig 2.10[8]: Bad choice of switching layers

2.5.3 Pins:

All the input and output pins of all the standard cells should be on the grid intersection points of horizontal and vertical grids so that the tool can find grid points and route efficiently. This is because the P&R tool places the vias only at these intersection points as discussed earlier.

2.5.4 Fixed Height, Variable Width:

Digital layout with particularly interest on standard cells should have fixed height and variable width (integer grid width) for all the cells, because we are constrained by the P&R tool. The height is determined by the most complex cell in the library. Cell complexity may come in terms of the number of transistors, internal routing or both which leads to more wiring path as in the case of a D-Flipflop with set and reset inputs (fig2.11) or due to device sizes as can be the case with series PMOS structures. For example in a 3 input NOR gate, the PMOS transistors in the pull-up have to be made wide enough to adjust for the poor rise time and high leakage currents due to the parallel NMOS transistors in the pull-down.

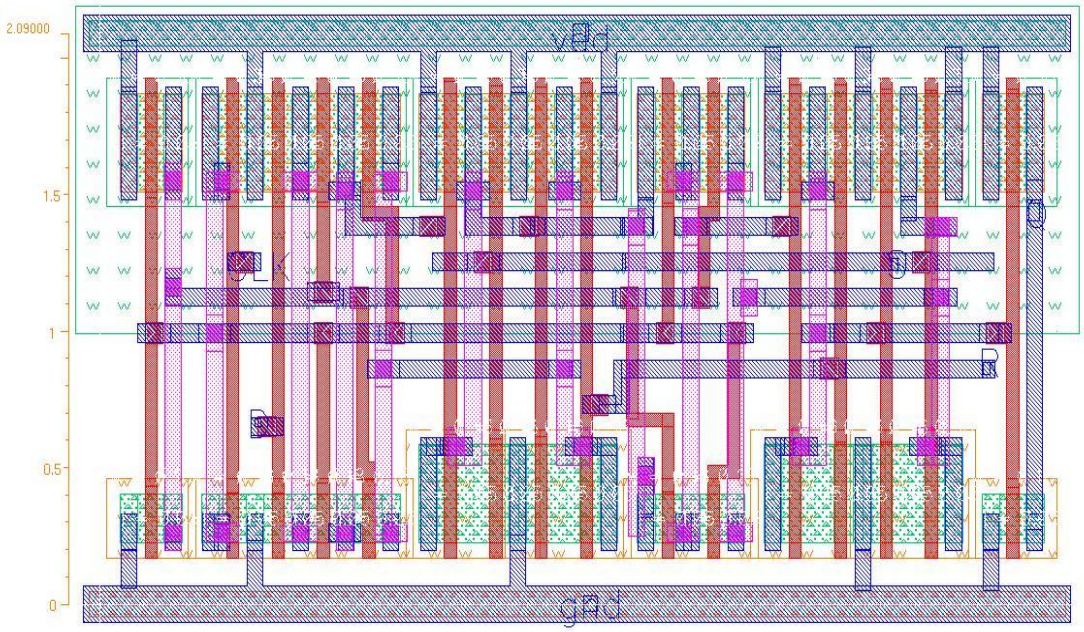


Fig 2.11: D-Flipflop with Set and Reset inputs.

If we need cells with high drive strength buffers, the transistors are made wider. Transistors are folded, drawn as fingers so that the cell fits into the fixed height. There are no restrictions on the width of the cell except that it should be a multiple of horizontal grid value. In this way, when the cells are placed on the power rails, they are abutted without any DRC errors.

2.5.5 Contiguous N-Well:

In any process, there is a spacing rule defining how close two identical layers can be placed. The N-well spacing is a large value than the transistor spacing. If each cell has its own well, they have to be separated by a greater distance whereas if all the cells in a row have a common N-well, then the cells can be placed such that it just satisfies the transistor spacing rule leading to greater circuit density. Fig 2.12 gives an idea of the spacing rules in the two cases.

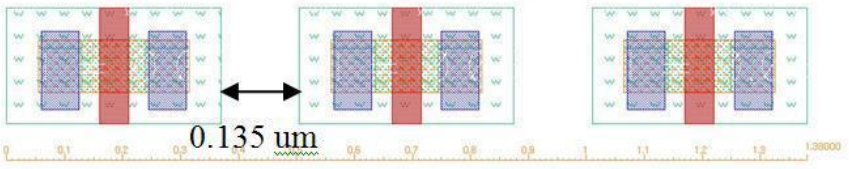
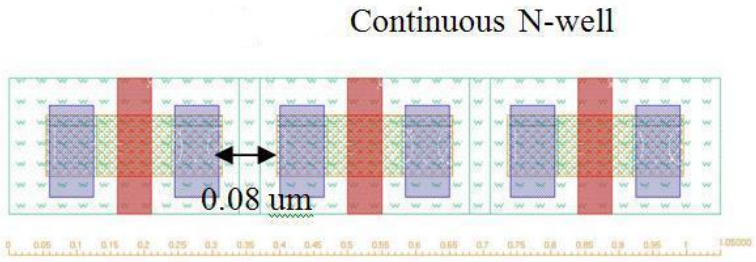


Fig 2.12 Nwell spacing



An important observation is, how far to extend the N-well for each cell to achieve a DRC free chip from the P&R tool. The answer depends on the design of FILLER CELL.

2.5.6 FILLER CELLS:

There are two approaches to design a FILLER CELL:

Approach 1: FILLER0

After the final optimization phase during the place and route, there are gaps left out between the cells in different rows. These gaps will cause DRC errors. To avoid them and make the N-well continuous the gaps are filled with the FILLER CELLs. Fig 2.13 shows a minimum dimension filler cell. It has no active devices, just the power rails and an N-well[11]. Usually there are a number of such cells in the library whose widths are multiples of a unit grid width. With this approach, the n-well should be at least extended to the cell boundaries in all the standard cells to

get a clean chip after place and route. As a safety measure, it might be extended a little beyond the cell boundary.

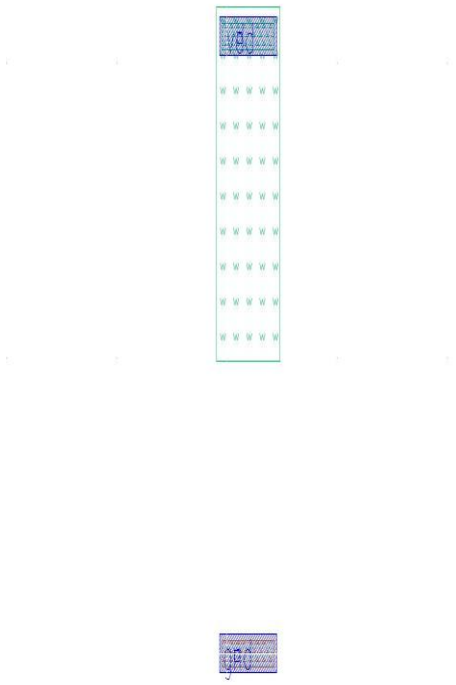


Fig 2.13: FILLER0 CELL Approach 1 where the boundary width is one grid.

Approach 2: FILLERn

This approach says the filler cells have to achieve more than just filling the gaps. The main idea besides filling the gaps is that these cells act as decoupling capacitors[12]. The decoupling capacitors prevent the sagging of power supply in the event of drawing large currents from the power rails. An example of such an event is when a register is being set or reset. When this approach is considered, the N-well in the standard cells should be extended beyond the cell boundary by a distance equal to half the FILLERn CELL width to ensure a DRC clean chip from P&R tool as shown in fig2.14.

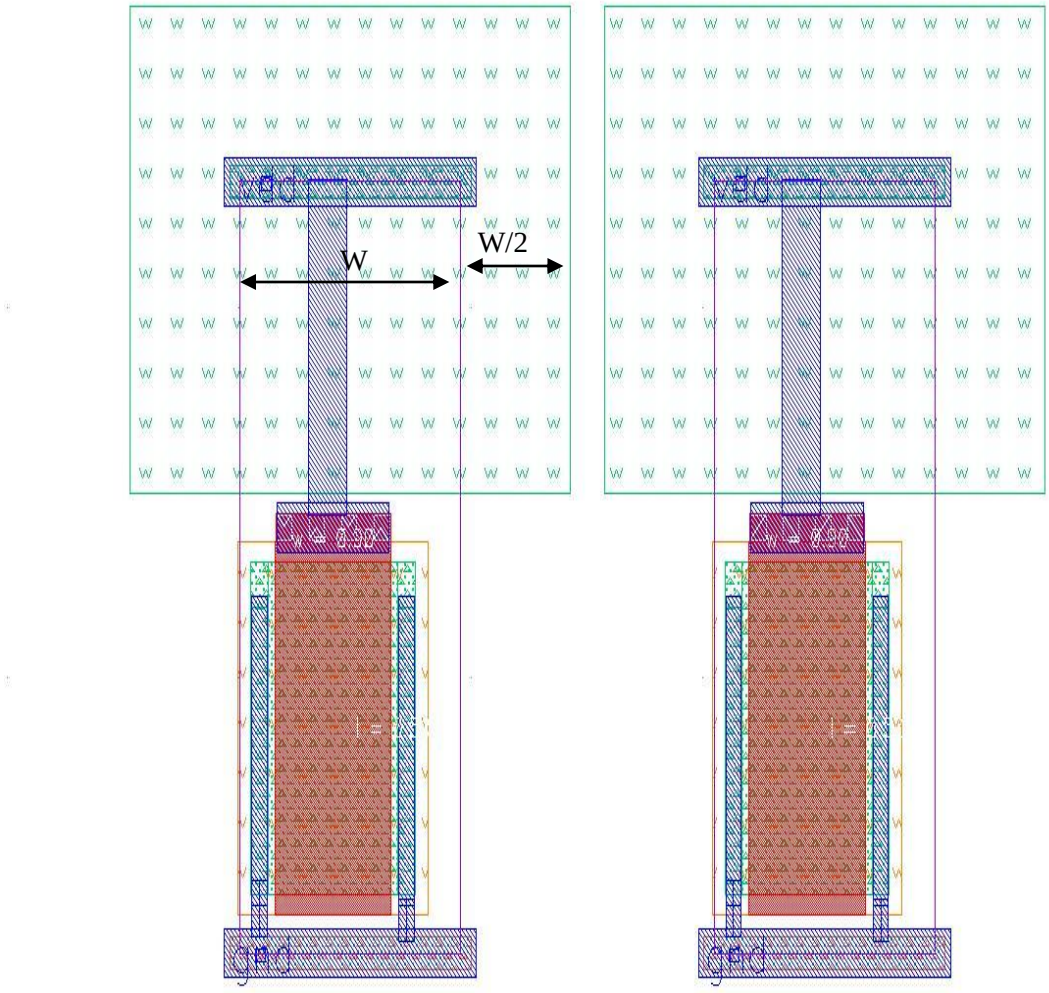


Fig 2.14: FILLERn cell and N-well extension for the standard cells

When two standard cells are placed such that the N-wells overlap, the result is a successful scenario. But when they are placed so that there is a gap, we should have enough space to accommodate a FILLERn CELL there to avoid DRC errors. But if the gap is insufficient to place a FILLERn cell then a FILLER0 cell can be used there. This requires the FILLER0 cell to be a single grid in width. The approach adopted here is a hybrid approach taking advantage of both by using FILLERn cells in the larger gaps for supply decoupling and FILLER0 to ensure a continuous N-well.

2.5.7 Half-design rule:

When we abut two cells adjacent to each other, we have to be careful about how it affects the internal components. If the spacing is not correct DRC errors can result. To ensure the errors don't occur, when abutting cells, all the layers should be set at specific distances from the cell boundaries by at least half the minimum spacing rule between them. In fig2.15, the violet box represents the cell boundary (the P&R tool abuts these boundaries when placing two cells side by side) and cyan box represents the boundary beyond which no internal layers of the cell may exist.

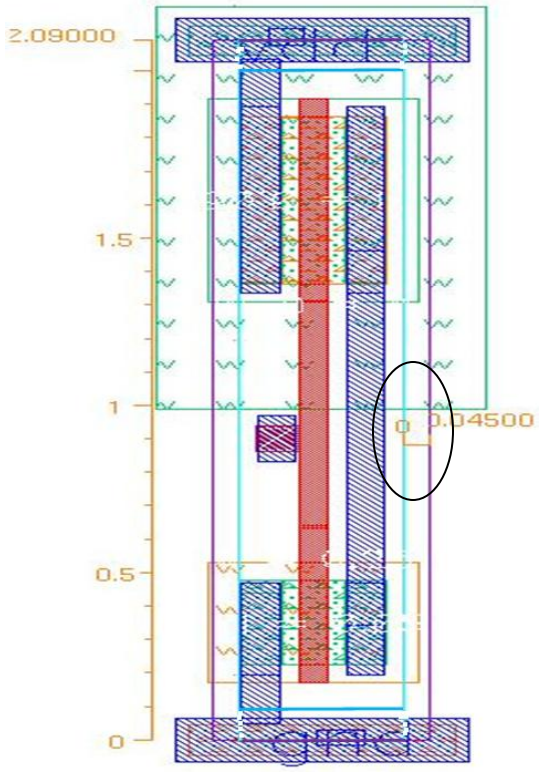


Fig 2.15: Standard Cell design template

Table 2.2 lists the distances from the P&R boundary, which no internal layer of a cell should cross. *Nwell* is made continuous and *active layer* has the maximum spacing rule after the nwell, which is 80 nm for the FREEPDK process. So, 45nm a little more than half of this spacing rule is

chosen. When this rule is followed, since the other layers have a spacing rule less than this, it's guaranteed that DRC errors don't occur when two cells are abutted horizontally or vertically.

Table 2.2: Rules for the internal layers from P&R boundary

Horizontal	45nm
Vertical	45nm

2.5.8 Antenna rules:

The antenna rule checks whether all the gate inputs are tied to diffusion before metal one is processed. This is because during dry etching of the metal layers charge builds upon the interconnect wires which can destroy the gate oxide if not allowed to discharge through the substrate. In the modern deep submicron processes where the oxide thickness is scaled down to a few nanometers, enforcing this rule is very important. For this process FREEPDK, the oxide thickness is 1.1 nm.

In order to make sure all gates are tied down, reverse biased protection diodes are connected to the FET inputs of all library cells as shown in fig2.16. If a gate is driven by the output from another node in metal1 and within a library cell, then it becomes unnecessary to add the protection diodes. The antenna rules for this process are yet to be incorporated.

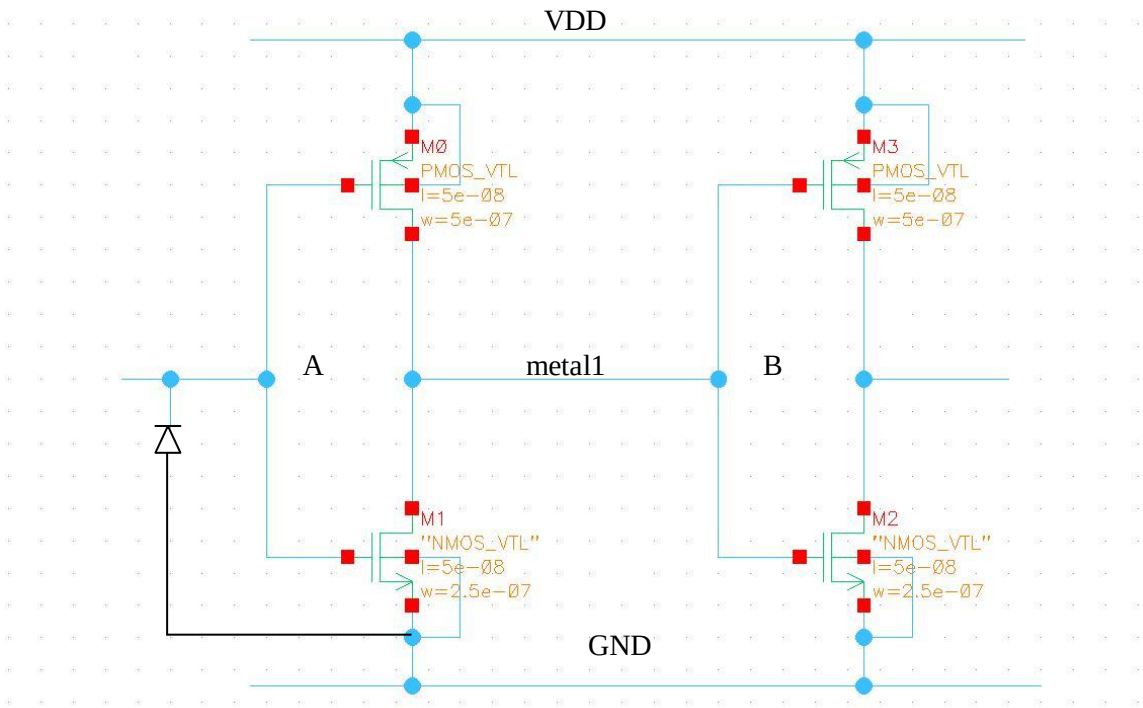


Fig 2.16[8]: Input of A is floating, so it needs a tie-down. However, the input B doesn't need any diode, since it is tied down by the output of inverter A and the internal run of the cell is short.

CHAPTER III

CHARACTERIZATION OF THE STANDARD CELL LIBRARY

3.1 Introduction

The VHDL/VERILOG simulator needs to have process specific software descriptions of each logic function in the library, such as the rise time, fall time and propagation delays, etc. for multiple circuit conditions of loading and input slews. This information along with other device parameters and physical representation of each of the gates are collectively known as the standard cell library. When synthesizing the behavioral Verilog description into a collection of standard cells, the synthesis tools need to have access to this library description information which is obtained by a process referred to as characterization.

The main objectives attained by the characterization process:

- ✓ Logic function of each cell.
- ✓ Load, each cell input will present to a signal connecting to it.
- ✓ Speed of the cell under different input rise/fall time and output loading conditions.
- ✓ Power consumed by the cell.

Cell Characterization is the process of simulating a standard cell with an analog simulator or an automated characterization tool to extract this information and convert into a format that other tools can utilize. Characterization requires; adequate logic, timing, power consumption for each

cell in the library.

Cell Characterization can be completed by analog simulation using *Spectre/HSPICE* simulator, whose output can be evaluated to generate the timing characterization data or by using an automated tool to tabulate this data. However, using an automated tool like *Encounter library characterizer (ELC)* makes the process clean, easy and error free when setup properly. The tool uses an analog simulator to simulate the design, and wraps up a nice interface to automate the process and give the results in the standard Synopsys liberty file format.

The chapter focuses on developing a tutorial for characterizing any given standard cell library, explaining not only the tool setup but how the above mentioned objectives are reliably attained and what drives us to select the different parameters such as the input slew, loads etc. that need to be set up. To know how delay is estimated, but more importantly understand its origin so that we can achieve realistic specifications for library timing, a delay model is presented.

3.2 RC Delay model

Model: A model is a simplified representation of a physical element for carrying out analysis on, so that when the entity is experimented or simulated, it yields results comparable to those obtained from analysis.

A delay model for example is used to estimate the delay of a circuit. The circuit is an interconnection of different circuit elements like transistors, resistors, capacitors etc, some of which are linear and others non-linear. They should be modeled to perform analysis on the circuit and obtain the delay. In the RC delay model, the transistor is modeled as a switch in series with a resistor (fig3.1), with an effective resistance chosen to match the average amount of current delivered by the transistor and a capacitor representing the diffusion node capacitance, with its

other end connected to ground. The gate capacitance of the transistor is modeled as a capacitor connecting between the incoming signal input and the ground.

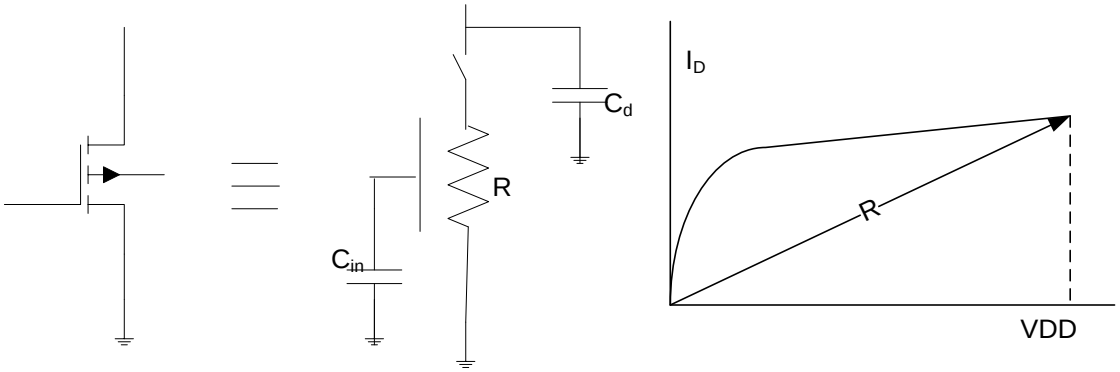


Fig 3.1: RC model for a transistor

3.2.1 Linear Delay Model: *[10]*

The propagation delay of a gate *d* (normalized with 1X inverter), can be defined as

$$d = f + p \text{ [10]}$$

f: effort delay

p: parasitic delay

3.2.1.1 Effort delay: The complexity (*g*) and the fan-out (*h*) of a gate contribute to the effort delay and can be written as

$$f = gh \text{ [10]}$$

The complexity (*g*) can also be called as logical effort which determines the capability of a logic gate in delivering output current compared to an inverter, given the inputs of a logic gate experience the same capacitance as that of an inverter input. This can be calculated from

$$g = \frac{Cin_{gate}}{Cin_{inv}} \text{ [10]}$$

It can be seen that the logical effort thus depends on two factors:

1. The device scaling of NMOS and PMOS devices. If the devices are scaled to account for the mobility differences of electrons and holes, then the capacitance each input sees increases as the number of inputs increase. However, the parasitic delay goes down because of reduced resistance of the wide transistor.
2. If all the gates use minimum sized transistors, then the logical effort is same for all of them. But in this case the parasitic delay for complex gates becomes large.

Thus, for complex gates that are beta-matched the input sees a larger input capacitance and hence a greater logical effort indicating they take longer to drive a given fan-out compared to a simple inverter. The complexity of a NAND2X1 compared to INVX1 is as shown in fig3.2:

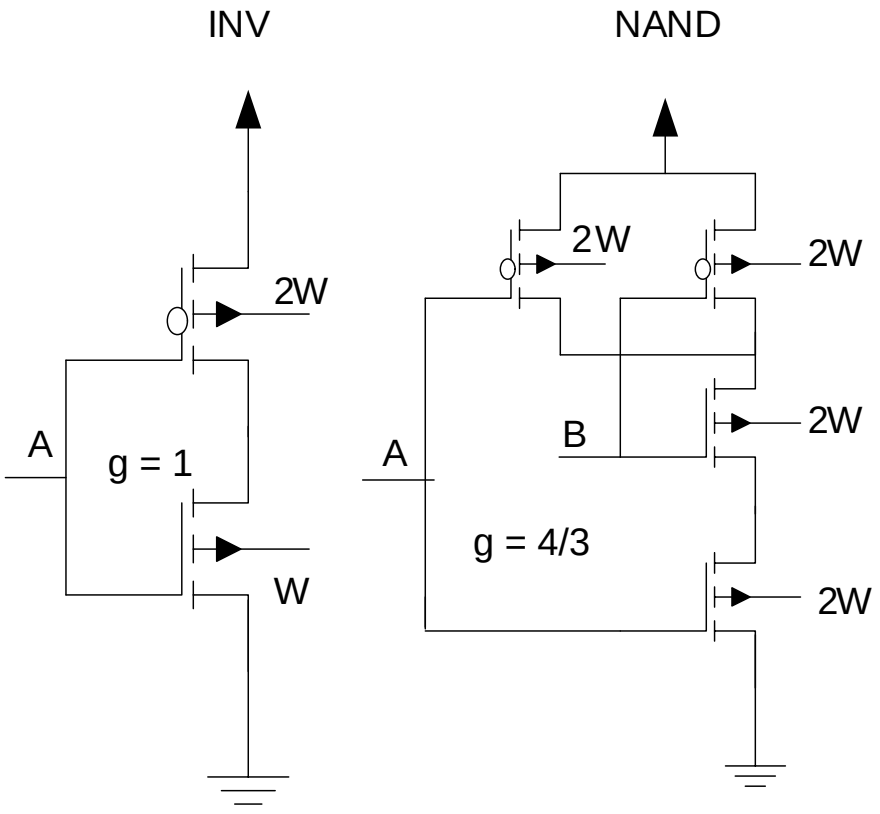


Fig 3.2[10]: logical efforts of INVX1 and NAND2X1

The fan-out (*h*) is defined as the number of identical copies of itself, a gate is driving. Or simply

$$h = \frac{C_{out}}{C_{in}}$$

C_{out} is the load capacitance being driven.

C_{in} is the input capacitance of the gate.

So, the greater the load that has to be driven, the greater is the fan-out and hence resulting in greater delay. The cell library will be characterized for different fan-outs/loads. The 1X load has been established in section 2.3.

3.2.1.2 Parasitic delay:

The delay of the gate when it is driving no external load is called the parasitic delay and is constant for a given gate. This can be estimated from the Elmore delay model. Each of the transistors is modeled as a switch in series with a resistor and a capacitor and the delay can be calculated depending on how the inputs change. Let us see an example to understand how the Elmore delay is calculated:

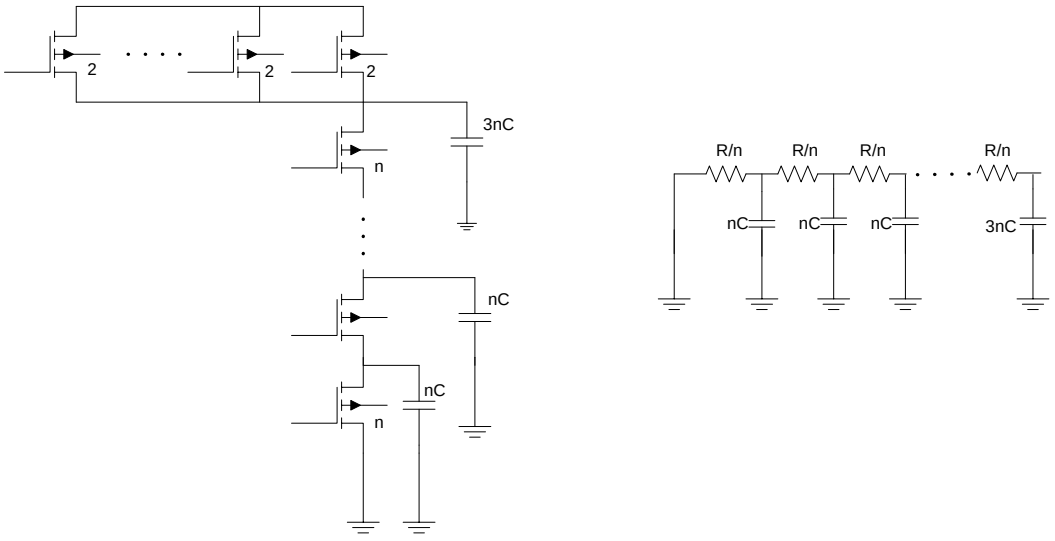


Fig 3.3: n-input NAND gate and its equivalent RC delay model when all the inputs are rising.

The Elmore[13] delay estimates delay as

t_{pd} = *sum over each node the product of resistance between that node and the supply, and the capacitance on the node.*

In fig3.3, the devices are assumed to be beta-matched accounting for equal rise and fall resistances. We are calculating the fall delay, i.e. when all the inputs are rising.

$$t_{pd} = R(3nC) + \left[\frac{R}{n} + \frac{2R}{n} + \frac{3R}{n} + \cdots + \frac{(n-1)R}{n}\right] nC$$

$$= R(3nC) + \frac{n(n-1)}{2n} RnC$$

$$= \left(\frac{n^2}{2} + \frac{5}{2}n\right) RC$$

In this manner delay can be calculated from the model. An important observation to be made here is that the delay through a gate increases quadratically as the number of devices in series increase.

Determining the worst case parasitic delay in the FREEPDK library:

As we just saw the delay is quadratically increased by connecting devices in series, the worst case parasitic delay occurs for a 3 input NOR gate where the pull-up network has 3 PMOS transistors connected in series. Let us calculate its delay. Note the 3 input NAND would also be as large were it not for the higher mobility of NMOS.

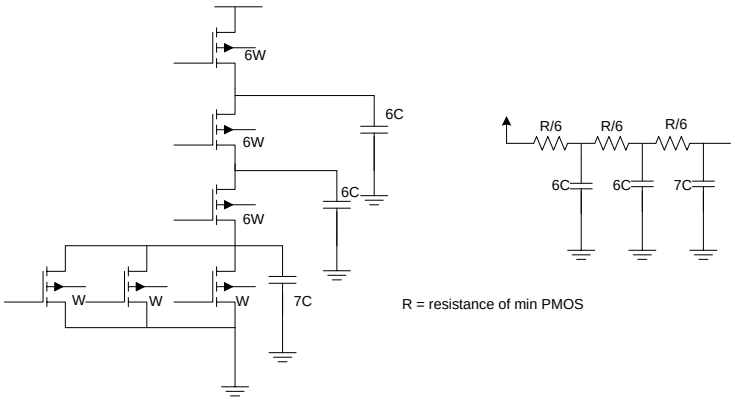


Fig 3.4: Elmore delay for a 3 input NOR gate

$$t_d = \frac{3R}{6} * 7C + \frac{2R}{6} * 6C + \frac{R}{6} * 6C$$

$$= \frac{21+12+6}{6} RC$$

$$= 6.5RC$$

Calculating R *(resistance of minimum PMOS transistor L = 50nm, W = 90nm)*

Process parameters:

$I_{on,PMOS}$ = -213 uA/um

Vdd = 1.1V

$$R = \frac{V}{I} = \frac{1.1}{213 * .09} = \frac{1.1}{19.17u} = 57K\Omega(approx)$$

Table 3.1 compares the calculated value to resistances of minimum PMOS transistors for IBM 0.18um and TSMC 0.18um processes:

Table 3.1: $R_{min, PMOS}$ for different processes

PDK	FREEPDK45	IBM 0.18um	TSMC 0.18um
$\mathbf{R_{min, PMOS}}$	57 KΩ	32.7 KΩ	24 KΩ

Calculating C *(source or drain diffusion capacitance of minimum transistor)*:The diffusion capacitance (C_d) results because of the reverse biased pn diode that forms between the *active* and *nwell* regions in case of a PMOS transistor as shown in fig3.5. Also, there are sidewall capacitances as shown in fig3.6:

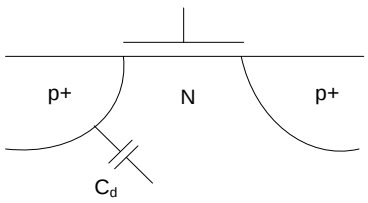


Fig 3.5: Diffusion capacitance (Nwell diode)

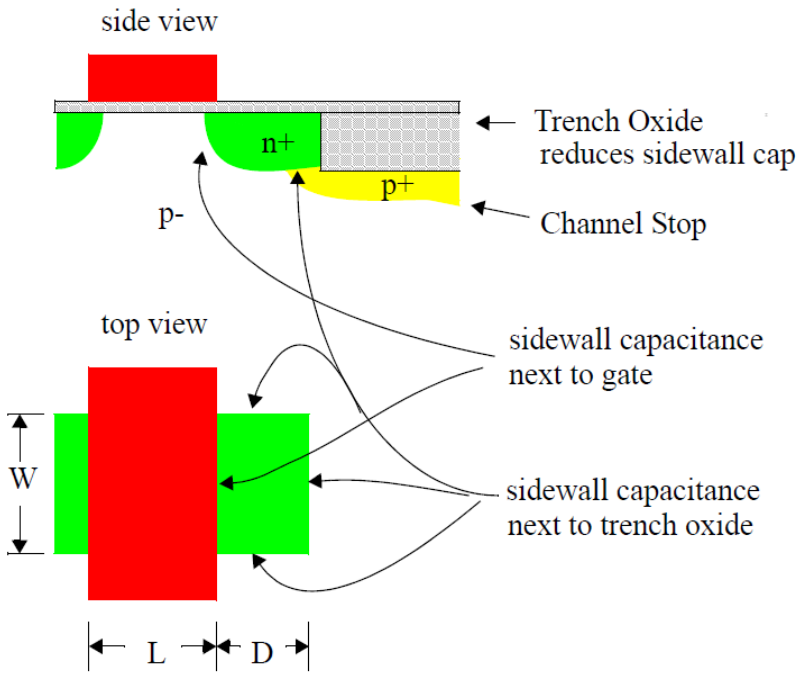


Fig 3.6 by [Dr. Louis Johnson in 4303 class notes, Parasitic R, L and C]: Sidewalls contributing to Diffusion capacitance

Therefore, total diffusion capacitance is,

$$C_d = \left(\frac{C_d}{A}\right) WD + \left(\frac{C_d}{P}\right)(2D + W) + \left(\frac{C_{dg}}{P}\right) W$$

Where, $\left(\frac{C_d}{A}\right)$ = diffusion capacitance per unit area

$\left(\frac{C_d}{P}\right)$ = diffusion capacitance per unit perimeter next to thick oxide

$\left(\frac{C_{dg}}{P}\right)$ = diffusion capacitance per unit perimeter next to gate

W = width of the transistor

D = Diffusion beyond the poly gate.

The above values for the FREEPDK process are:

$\frac{C_d}{A} = 0.5 fF/um^2$

$\frac{C_d}{P} = 0.5 fF/um$

$\frac{C_{dg}}{P} = 0.3 fF/um$

$W = 0.09um, for\ a\ minimum\ PMOS$

$D = 0.07um$

Substituting the values, we get

$$C_d = \{(0.5 * 0.09 * 0.07)\} + \{(0.5) * (2 * 0.07 + 0.09)\} + \{0.3 * 0.09\}$$
$$= (0.00315 + 0.115 + 0.027)\ fF$$
$$= 0.15\ fF$$

The $\left(\frac{C_d}{P}\right) 2D$ term in the equation remains constant with increase in *width* W of the transistor. But for the NOR3X1 gate rise time delay is calculated by scaling the total capacitance with W which gives capacitance a little higher.

Therefore, the parasitic delay of a NOR3 gate, when all the inputs are falling is

$$t_d = 6.5RC = 6.5 * 57K * 0.15f = \mathbf{55.57}\boldsymbol{ps}$$

The library should be characterized by inputs which don't rise faster than **55.57ps.**

***Summary** of RC delay model:*

The linear delay model for a logic gate demonstrates that the delay through a gate has two terms, the first one an effort delay which is proportional to the complexity of the gate and is linear in relationship with fan-out and the second one is inherent parasitic/intrinsic delay dependent on gate topology.

$$\boldsymbol{d = gh + p}$$

The model can be represented in the form of a graph as shown in fig3.7:

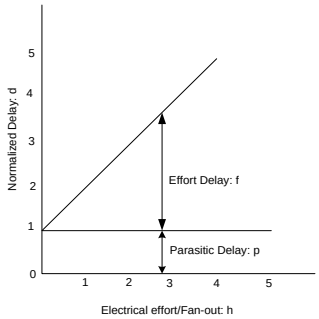


Fig 3.7[10]: Linear Delay model, *delay vs. fan-out*

The y-intercept gives the parasitic delay and it can be seen that the effort delay depends on the fan-out linearly.

The simple but useful linear delay model gives us a good estimate of delay, its contributors and how it can be modeled. However, it cannot be used as such to characterize a cell because of model limitations, some of which are:

1. The slope of the incoming signal affects the delay because until or unless the signal reaches fully high it doesn't turn ON/OFF the transistors completely. Hence the slower the signal rises, greater is the delay. This effect has to be modeled for more accurate delays.
2. The arrival timing of different gate input signals affects the delay.
3. Gate and D/S to body capacitances are nonlinear.
4. Effective switch resistance is nonlinear.

Incorporating all these effects into the delay model, makes the analysis complicated and results unpredictable and hence simulation is the only means by which we get the accurate delays needed by *Encounter (P&R), Synopsys Design Compiler (Synthesizer)* tools and is carried out by the *Encounter Library Characterizer.* The *Encounter Library Characterizer contains a SPICE simulator.*

3.3 Encounter Library Characterizer (ELC):

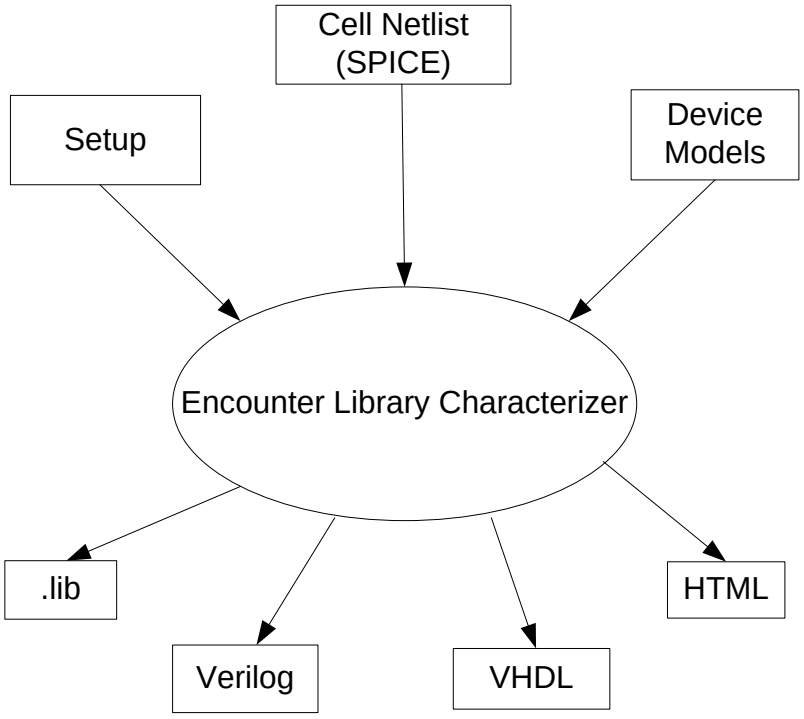


Fig 3.8: I/O Encounter Library Characterizer.

Let us review what Cadence describes as the capabilities of the Encounter Library characterizer. Fig 3.8 shows its inputs and outputs.

ELC performs a series of operations to complete the characterization:

1. The cell netlists are analyzed to determine the logic function and the type of logic used i.e. which circuit family whether CMOS, Pass transistor or tristate logic for combinational cells.
2. The different electrical specifications such as the pin-to-pin delay, setup and hold-time constraints, pin direction are defined for the combinational and sequential cells.
3. Accepts input parameters such as the supply voltage, temperature, input slew rates, output loads and process corners from the setup file. With the transistor model coming

from the device model file, the HSPICE simulator is invoked and executed. The timing, power and logic results are summarized and delays tabulated.

4. The characterized data is obtained in *advanced library format (*.alf)* that can be converted into any of the formats shown in fig 3.8. The liberty file format is a often used standard and recognized by many CAD tools.

We will look at how to set up each of the inputs to the tool to yield correct results.

3.4 ELC inputs:

3.4.1 SPICE Netlists:

The netlist defines how the devices are connected i.e. node connections between the transistors. It also defines the device geometries: *length*, *width*, *diffusion areas* and *perimeters*. These help in estimating the parasitics where the per unit values of capacitance, resistance come from the device model file. These netlists are extracted from the layout using the Calibre Interactive tool. To stress, it's the extracted layouts not schematics that must be used because only then are the parasitics accurately estimated. Calibre Interactive can also be used to perform the physical verification DRC/LVS, and parasitic extraction (PEX). It has both a gui and command line option to work with. A tutorial on how to work with this tool can be found at this reference[14].

3.4.2 Device model file:

The device models come with the given PDK and consist of both passive and active device models. Typically in the library we are only interested in transistor models. It defines various parameters such as the threshold, per unit capacitances, resistances, oxide thickness, etc. If we are characterizing a cell at different corners i.e. slow-n slow-p, slow-p fast-n, fast-p fast-n, typical-n typical-p then we must have the transistor models for all such cases. The cell library presented and developed here has only been characterized for the typical case.

3.4.3 Setup file:

Setup.ss file defines many parameters which determine the characterization conditions:

3.4.3.1 Process Corner definitions:

The operating conditions of a transistor affect the propagation delays. The temperature affects the parameters like threshold voltage, mobility of electrons and holes. The transistor thus runs faster or slower. All the standard cells are in general characterized for 3 sets of conditions: best case, worst case and the typical case. The best case is when both the PMOS and NMOS transistors run faster than usual, with a higher power supply (typically +10% more than nominal) available and at the lowest expected operating temperature. The worst case is when both transistors are slow, at the higher expected temperature and lower supply voltage (-10% usually). Typical case is at the room temperature when both the transistors are typical and the nominal VDD. We have to characterize at different conditions because the temperature and vdd on a chip vary from day to day and with different applications as well as with die to die and in addition they may not be uniform throughout the die. The chip has to function, unaffected by these slight variations. So if we have the timing information under different conditions, the synthesizer can synthesize the design that can meet the timing across the corners.

3.4.3.2 Intrinsic delay and input slew:

The intrinsic delay is the inherent parasitic delay exhibited by a logic gate for the transistors to turn ON/OFF. The transistor currents cannot switch faster than this. The input slew is the rise/fall time of the input. Usually the output of a gate is connected to input of another, and hence when we are characterizing the logic gate individually it should not be given inputs that rise/fall faster than the intrinsic delay. This should be considered, while setting up the input slew. The rise time value is calculated using the linear delay model in section 3.2.1.2 and found to be **55.57ps**. An easy approach to calculate it through simulation is as shown in the fig3.9:

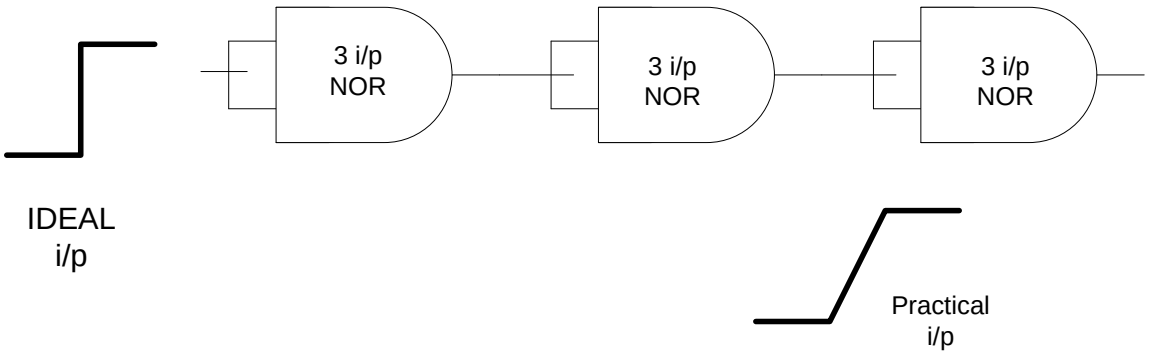


Fig 3.9: Input slew rate calculation.

By connecting three NOR3 gates from the FREEPDK cell library in series and driving the first with an ideal source, we will get 1X rise/fall at the output from the second as shown in fig3.9. Taking this slew rate as the fastest any input can rise/fall, we are making a conservative estimate, to ensure the cell to be characterized is driven at a safe slew rate. The cell is characterized for different slew rates that are decreasing multiples of this, since this will be the case when multiple gates are connected in parallel increasing the load factor or fan-out. Having this information, the synthesizer will do a better job in optimizing the design. Specifically for the NOR3s of this library the rise time is found to be **25pS** from simulation versus 55pS for hand analysis.

Table 3.2: Parasitic delays of NOR3X1 gate

RC delay model	55 pS
Simulation (tr/tf)	25/22 pS

Table 3.2 shows the rise times for a NOR3 gate obtained from RC delay model and simulation, which highlights the importance of simulation for accurate results.

3.4.3.3 Output net Capacitance:

The capacitance on the output node also contributes to the delay. Hence the cells are characterized for different loads to get the timing, to be used by the synthesizer. The 1X load is determined using the minimum sized inverter (INVX1) of the cell library and is found to be **1 fF** in section 2.3.

The characterization data is output in the form of a matrix with the input slew on one axis and the output load on the other axis. An example of how we setup the slew and the load is as shown below:

```
Index X1{
        Slew = 0.025n 0.05n 0.075n 0.1n 0.125n;
        Load = 0.05f 1f 2f 4f 8f;
};

Index X2{
        Slew = 0.025n 0.05n 0.075n 0.1n 0.125n;
        Load = 0.05f 2f 4f 8f 16f;
};
```

Note the use of 25pS as the fastest rise time. The choice of the 25pS as fastest rise **and** fall time is conservative. Though the fall time of the NORs, inverters etc. will be faster, the choice of 25pS result in conservative timing results from ELC. Any cell whose name ends in X1 will be characterized for the slews and loads defined in the X1 index, and for those which end in X2 will be characterized for twice the load as X1. If we have cells with more drive strengths, then we define more index statements. In the FREEPDK library cells, the maximum drive strength is 8X. The 0.05 fF in the load index is to obtain the parasitic delays of the gates.

3.4.3.4 Characterizing pin to pin delay:

ELC provides the pin to pin delays for the cells. This is the time that a change at the input pin takes to affect a change at the output pin. Pin-to-pin delay is defined as demonstrated in fig3.10 where vth is the threshold voltage level at which the delay is measured and is defined as half of logic high value or VDD/2.

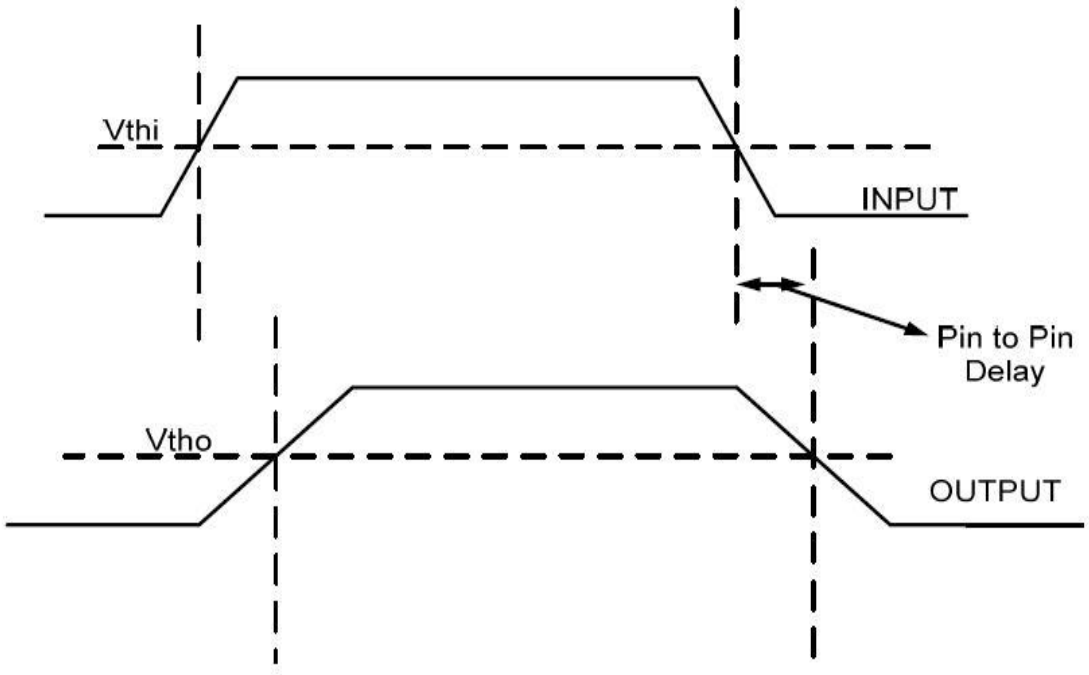


Fig 3.10: Pin to pin delay

3.4.3.5 Characterizing set up and hold times:

When characterizing a sequential cell, the timing constraints have to be met for the data to propagate correctly through it. Let us understand them. Fig 3.11 shows a negative edge triggered D-Flipflop.

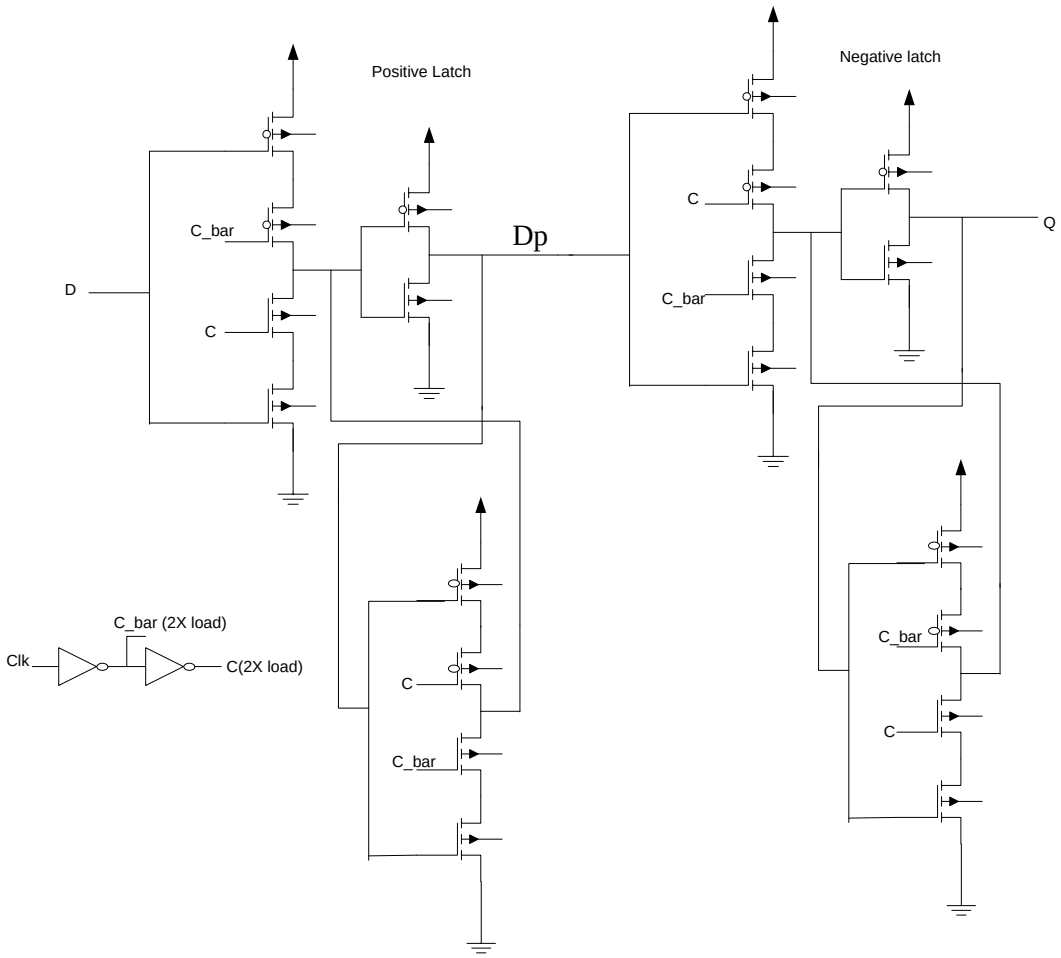


Fig 3.11: Negative edge triggered D Flip-flop

For the latch to properly acquire D, the signal should properly set up at Dp before the internal clocks (C & C_bar) disable the forward tri-state and enable the feedback tri-state. Note that both tri-states switch nearly simultaneously but due to the later arrival of C to the feedback tri-state it is an intrinsic delay later when a "0" is stored and an intrinsic delay earlier when a "1" is stored. (As a result the forward tri-state is selected stronger than the feedback tri-state.) The D to Dp delay is two gate delays each with a 1X load while the Clk to C is also two gate delays with each having a 2X load. Now depending on gate geometries i.e. minimum geometry, optimal delay or beta matched, the set up times can be either positive or negative depending on the relative delay of Clk to C and D to Dp. When the clock goes negative the output of the positive latch is

propagated to the output after a delay called *clock to Q delay.* The data should be stable until the clocked transistors in positive latch are turned off by the clock when it goes negative. Or else, the output of forward tri-state cannot be at a stable value. This time constraint is called the *hold time.* Since there is a delay from the external *Clk* to *C & C_bar,* the actual clock *Clk* may arrive at the same time the data arrives or even before the data arrives depending on the delay from D to Dp. The definitions of *set up, hold times* and *Clock to Q delay* are summarized below:

Set up time is the amount of time the data input of the sequential logic should remain stable before the active clock edge, so that the correct value is latched at the output. *Keep in mind that negative set up time values can be valid.*

Hold time is the minimum time the input signal to the sequential logic should remain stable after the active clock edge, so that the correct value is latched at the output. *Note zero values for hold times can be valid.*

Clock to Q delay is the time taken by the output to become stable after the active clock edge.

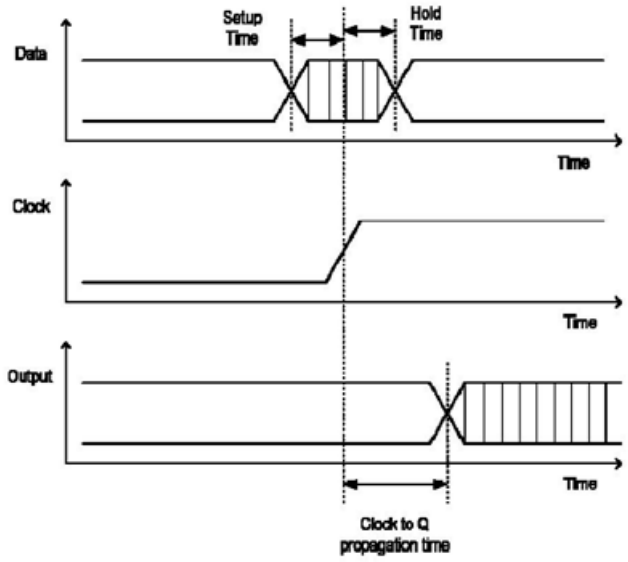


Fig 3.12: Set up time, hold time and Clock to Q delays

Binary Search: ELC characterizes the sequential cells by simulating them repeatedly to find the minimum set up and hold times to achieve the correct output within the pre-determined clock to Q delay. It uses the binary search algorithm to determine these timings. For example, in the set up time calculation, two initial values for which the simulation passes and fails respectively are given in the setup file. In the first iteration, the simulation is done by allowing the data to change with respect to clock at the midpoint of these two values. If it passes then the pass set up time value is updated or else the fail time is updated, and the iterations are continued until the specified resolution is obtained. The resolution determines when to quit the algorithm. If successive results don't show an improvement greater than this value the simulations are stopped. The resolution is chosen to be equal to the intrinsic delay of the fastest gate in the library. For INVX1, the rise time delay is *3RC = 3 * 57K * 0.15f = 0.03ns*. Similarly the hold time is also determined.

Bisec 6n 6n 0.1ns

In the above statement the initial pass and fail *set up* times are +6n and -6n with respect to clock and the final resolution is 0.1ns. Here '+' means before the active clock edge and '-' means after the active clock edge which is quite confusing. 5 to 10 times the worst case parasitic delay (5*25 = 125 pS) which would be greater than the delay through the first stage latch (positive latch incase of –ve edge sensitive FF) may be chosen for the pass and fail values. But the values chosen are much greater than this because the library is being characterized for a worst case input slew of 125ps which leads to a greater propagation delay and hence more *set up* time. An alternate and more accurate approach is to simulate the reset or preset DFF using the slow-slow process to find the initial window timing. The setup file is presented in Appendix1 and the timing information for the cells in the library in HTML format is included in Appendix2.

CHAPTER IV

ASIC INTEGRATION

4.1 Introduction

The different blocks we have such as the *Analog to Digital Converter (*ADC), *Digital to Analog Converter (*DAC), memory and the standard cells should be integrated according to the structural Verilog/VHDL netlist physically to complete the PC-ASIC. This is done with the help of a CAD tool and the process is called place and route (P&R). *SOC Encounter* has been used for this project. The basic design flow developed by the *VLSI computer architecture group, OSU* has been used to do the P&R. The main objectives of this chapter will be modifying the floorplan for the design to have separate power domains and discuss the problems involved while doing LVS for the final design using *Calibre* tool from MGC.

4.2 Setup

We have the following data available:

1. Structural Verilog code representing the design we want to implement interconnecting the ADC, DAC, memory and the Standard cells.
2. Timing information for the standard cells obtained from the characterization process, and other blocks in the liberty (**.lib)* file format. The **.lib* files for analog blocks and memory blocks just contain the input and output pin capacitances.

3. Abstract views of the Standard cells and other blocks in *layout exchange format* (*.*lef*). One of the LEF files should include the technology information for the process. All the design rules for the metal layers, vias, and antenna rules should be included to do the place and route without creating errors.
5. The delay constraint information in the *Synopsys design constraint (*.sdc)* format. This is obtained during the synthesis of behavioral code and describes the timing and loading of the primary inputs and outputs including the clock. It is needed by the P&R tool during timing optimization and clock tree synthesis.
6. Input/output (IO) assignment file. This describes the placement of pads around the chip. The input and outputs are from the ADC/DAC blocks and should be placed close to each other. The blocks are placed near these pads. The power pads are distributed around the chip, with the analog power pads placed again near to the I/O pins.

Place and route is now, placing the custom blocks manually and asking the tool to place the standard cells and interconnect them so that it functions according to the delay constraints given.

The files used to do P&R for this project are listed below:

1. top.vh
2. stdcells.lib, pads.lib, ADC_5bit_sc_5.lib, dac_op2.lib, myram.lib
3. stdcells.lef, pads.lef, ADC_5bit_sc_5.lef, dac6_op2.lef, myram.lef
4. top.sdc

The *stdcells.lib* and *stdcells.lef* (characterization data and abstract file for the new standard cells) are obtained in this work and the other files came from the previous work done by [J. Chen and J. Stine].

4.3 Design Import:

It's a good idea to create a separate directory in which we can run the encounter tool because

it creates many files during execution. All the above files should be copied into it. After the encounter install directory is included in the shell $PATH (try *echo $PATH* to find out) we can start the tool from the terminal using *encounter –win* command which brings up the gui as shown in fig4.1.

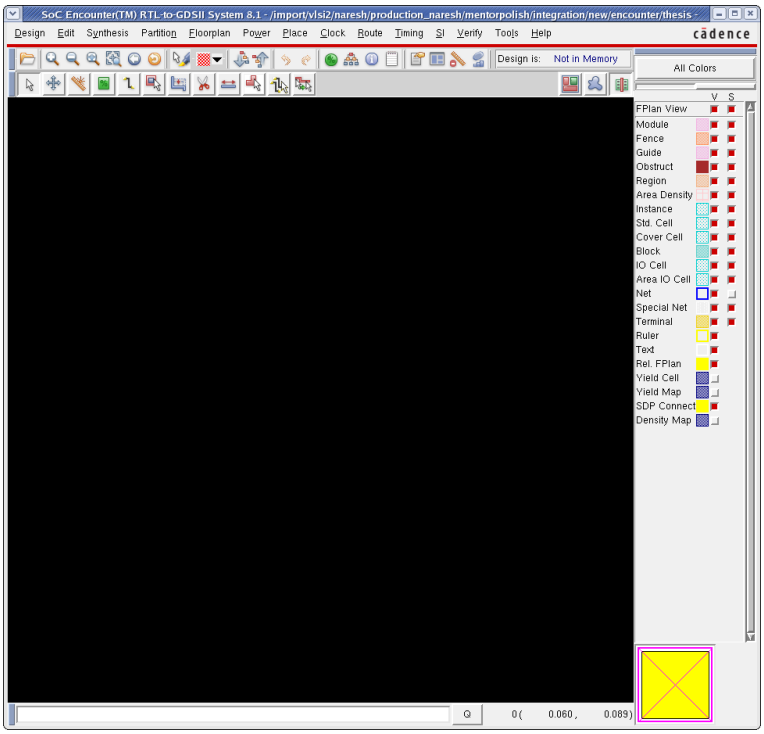


Fig 4.1: Encounter GUI (copyright 2005, 2010, Cadence Design Systems. Inc)

The input file paths are filled in (fig4.2) and in the *Advanced Tab* shown in fig4.3, the power net names are specified. Since the library has been characterized only for the typical case, we can only specify the path to that file in the *common timing libraries* field. We have separate power names for the core, analog and IO which are vdd, vdda, vddio and a common gnd respectively. The design import configuration should be saved. It is saved as **.conf* file. If something goes wrong we can then restart the encounter again and just use this file to import the design back.

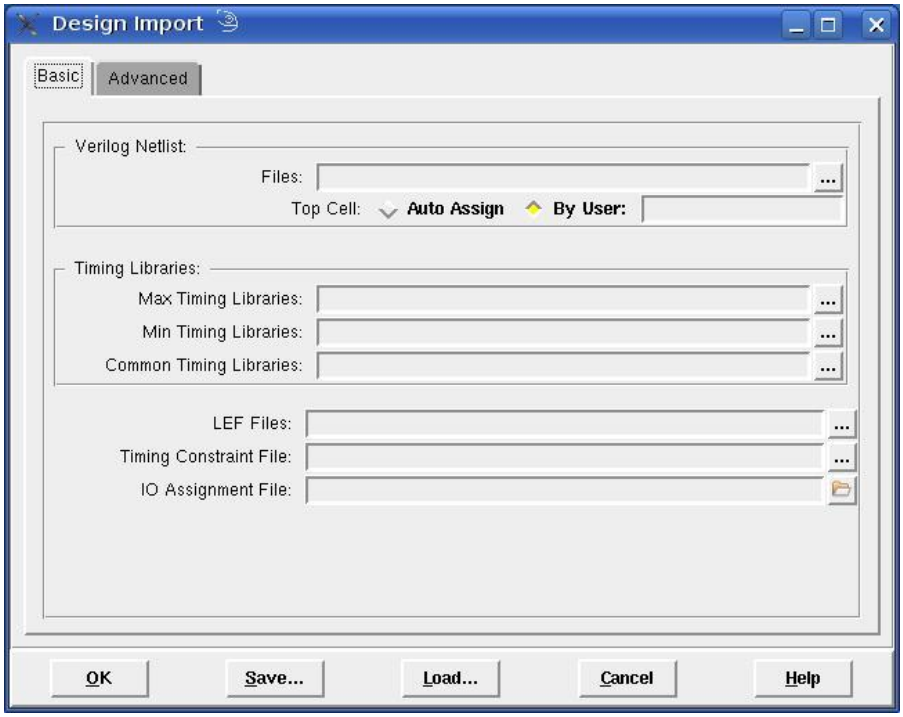


Fig 4.2: Design import – specifying the input files

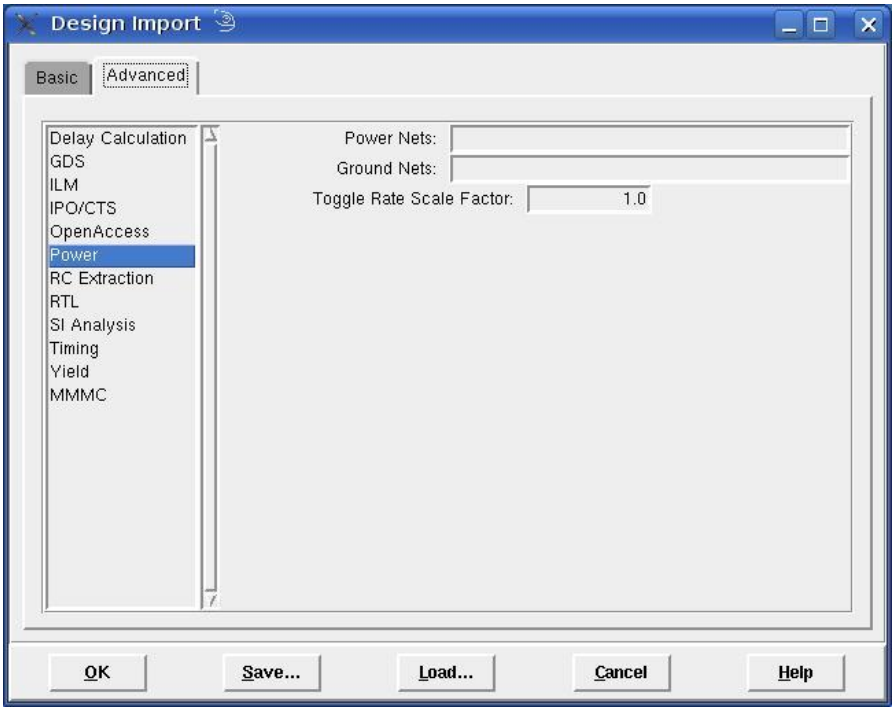


Fig 4.3 Design import specifying the power domain names

4.4 Floorplanning:

The initial floorplan for this design has been developed by [Dr. Junchen and Dr. James Stine, VCAG, OSU]. In this work it has been modified to include two analog pads and isolated power rings for the analog blocks. Also the power rings and stripes have to be changed to metal5 and metal6 from metal1 and metal2 to reduce the voltage drop on them.

After the design import, the GUI can be seen as shown in fig4.4. If there are any errors they will show up in the terminal window and have to be corrected before proceeding. The errors result usually because of syntax errors in LEF file. The abstraction procedure during which the LEF is obtained is explained in the thesis[15].

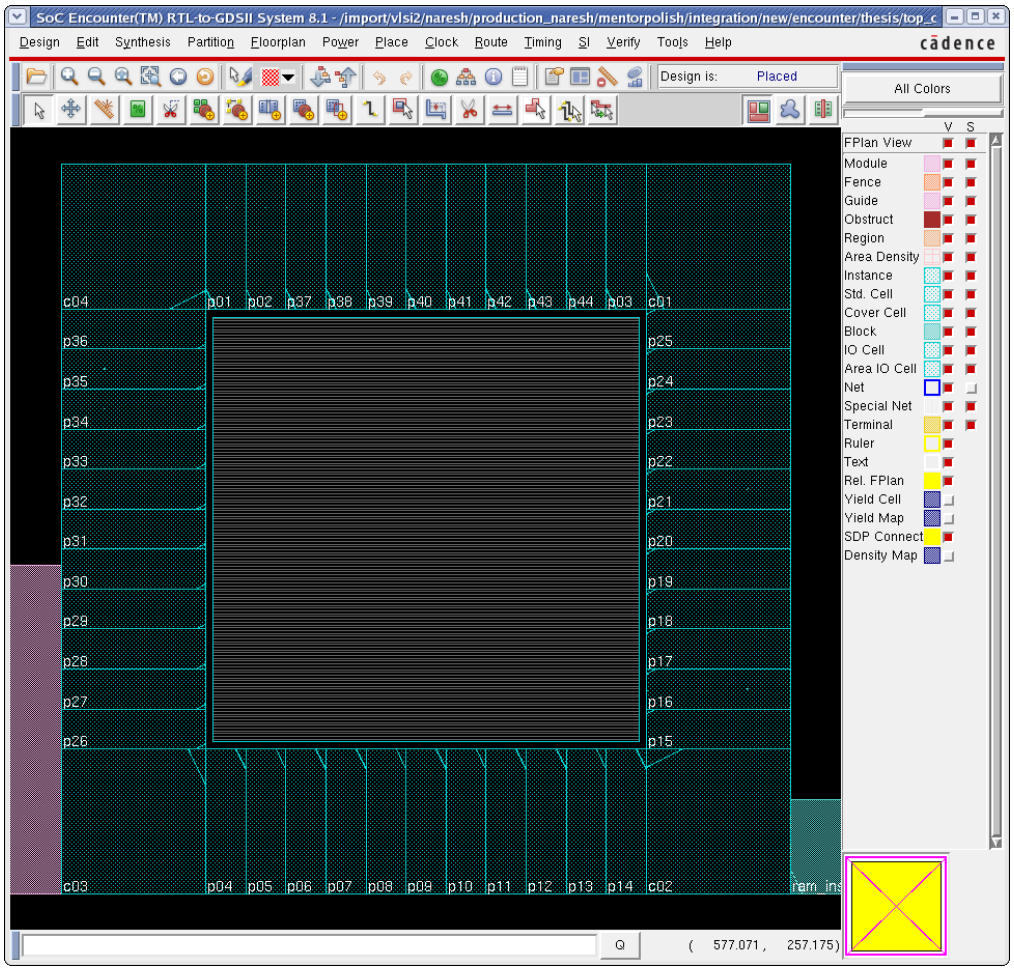


Fig 4.4: Floorplanning (copyright 2005, 2010, Cadence Design Systems. Inc)

We can see the following elements in fig4.4:

1. Pink area represents the amount of digital logic we have.
2. There are four custom blocks: ADC, DAC and two instances of memory on the right.
3. We can see there are some grey rows surrounded by the IO pads. The logic and other blocks have to be placed in this area.
4. There are four corner pads. The IO pads are designed in such a way that the digital core vdd and gnd rails form a continuous strip when any two pads are abutted. The corner pads join the strips on four sides to form a ring.

Now, to set some parameters, select *Floorplan -> Specify*

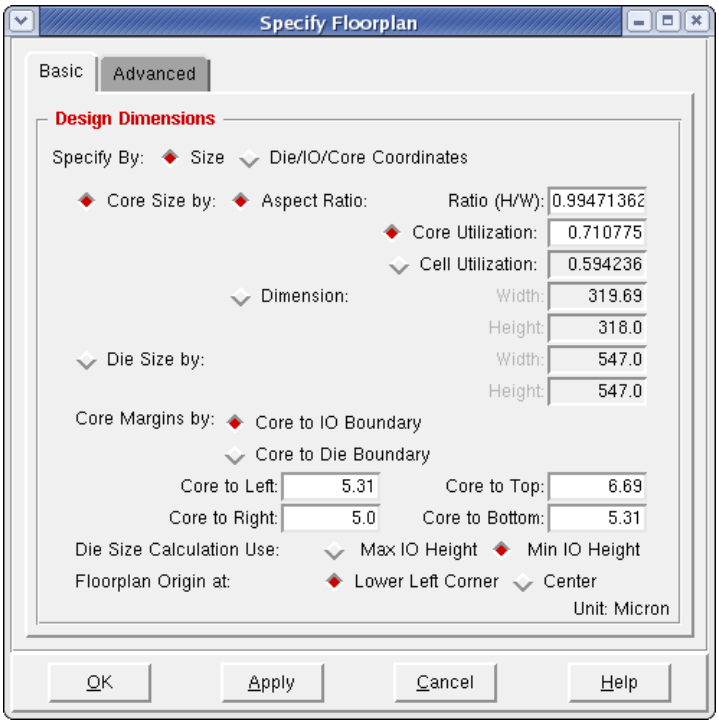


Fig 4.5: Specifying floorplan-1

Aspect ratio and *core utilization* can be left at their default values. An aspect ratio close to 1 means the chip will look like a square. The core utilization indicates how dense the standard cells should be placed. For a large chip this has to be reduced if the timing is not being met after P&R,

so that the standard cells are placed loosely to provide additional space for inserting buffers and meet the timing.

Core margins – Apart from the power rings in the pads mentioned earlier, we need to have another set of power rings between the pads and the core boundary. The Standard cell rows are connected to these rings. There should be enough space between the IO boundary and core boundary to accommodate the power rings. Since these rings are very long, the higher metal layers metal5-horizontal and metal6-vertical which have less resistance are used. The metal layers are chosen to be 0.95u which is 5 times the grid value.

A problem observed was, when spacing numbers to leave sufficient gap for the power rings between IO and core are given in the *Core margins* field, instead of shrinking the core, the die is expanded and pads moved out to accommodate the rings. The pad dimensions are set such that when they are placed around the die a continuous power ring is formed. Hence expanding the die will cause gaps between the IO pads causing breaks in the power ring. This is solved by choosing *specify by -> Die/Io/Core coordinates* option in fig4.5. The form that appears is shown in fig4.6.

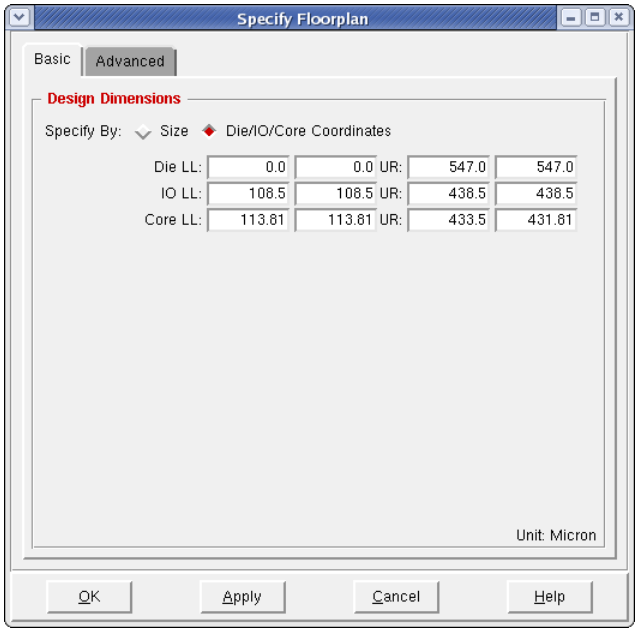


Fig 4.6: Specify floorplan-2

LL represents lower left corner and *UR* represents upper right corner in the fig4.6. Die and IO positions are left unchanged. Looking at the IO coordinates, the core options are changed to (113.81, 113.81) and (433.5, 431.81). This shrinks the core area leaving sufficient margin for power rings.

In the *Advanced Tab* the fields including *row height*, *Bottom IO Pad Orientation* and others should be verified if they are correct. The row height is equal to the height of the standard cells. If it's incorrect it can be changed only in the SITE *core* definition in the LEF file.

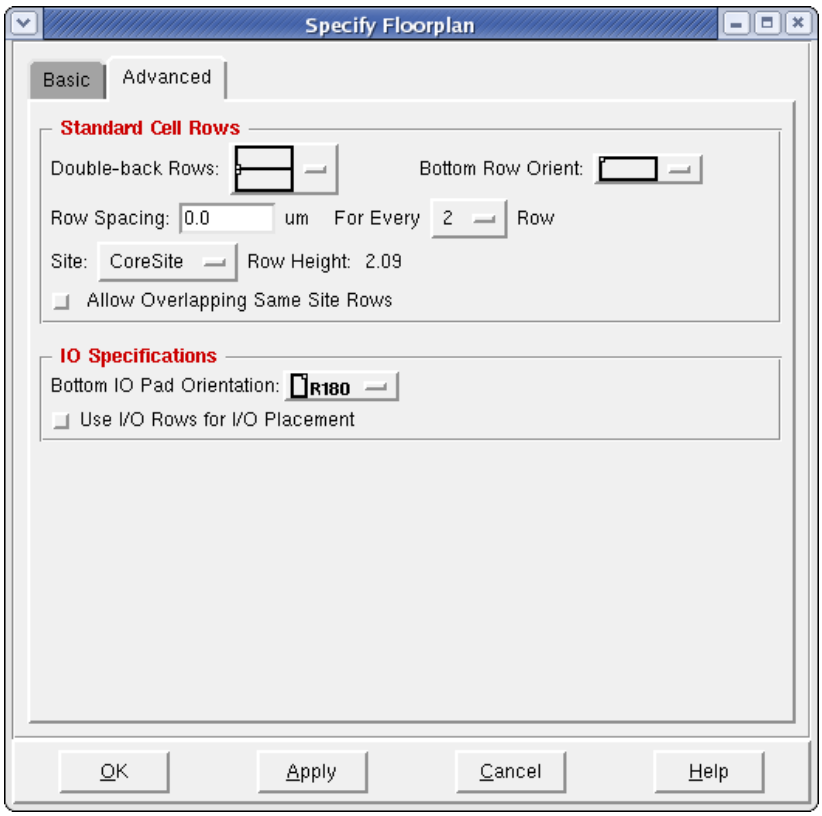


Fig 4.7: Specify floorplan-3

SAVE!! The design should be saved, when considerable effort has been spent and expected results are observed. An undo feature is not available in Encounter. The file can be saved using *Design -> Save Design as -> SOCE* in SOCE format. When something wrong occurs later, we can restore the design using this file rather than starting over.

Block Placement:

There are four blocks in this IC designed by [J. Chen and J. Stine]:

1. ADC
2. DAC
3. Memory – 2 instances

When we click on a block, we see a set of flight lines showing all the connections coming in and out of the block. Depending on the connections to the IO pads and complexity of wiring it leads to, they are placed at appropriate locations inside the core area. After that, all the standard cell rows in those areas should be deleted. This can be done by selecting *Floorplan -> Edit Floorplan -> core rows -> cut*

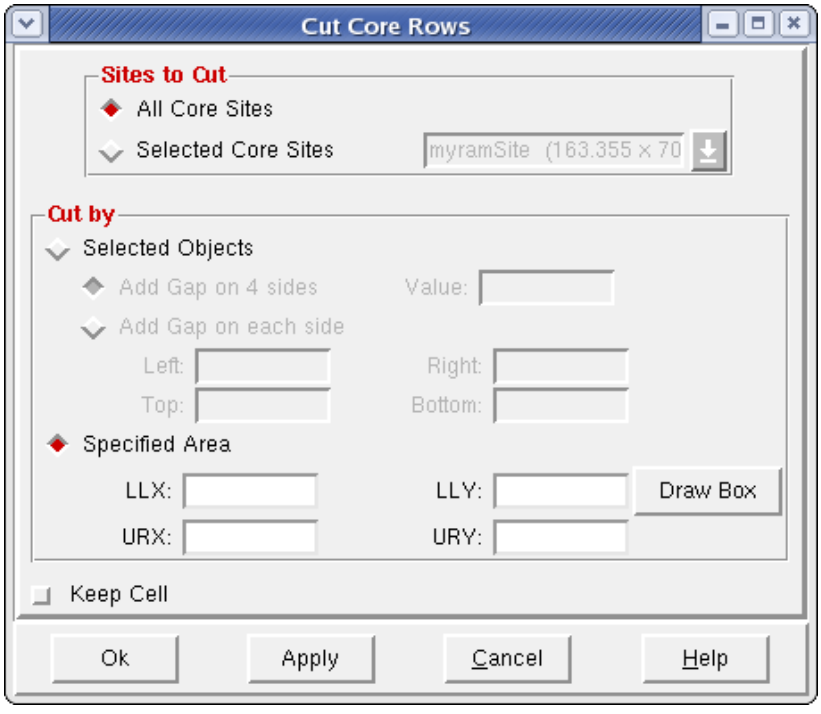


Fig 4.8: Cut the core rows in the areas where blocks are placed.

Select the option *Cut by -> Specified area* and then click the *Draw box* option to select the areas in which the rows should be deleted.

Power Planning:

In this step the power rings are added around the core and then the stripes are created to form a grid like structure. Since the analog blocks have a different power supply, they should have their own rings around them. The main objectives of designing a power grid are to counter the following effects:

1. *IR drop on the power rails*: The devices in a grid and the power rails form a voltage divider. The effective resistance of a grid should be much greater than the resistance on the power rails to avoid any voltage drop on the power rails and ensure the gates don't fail due to insufficient vdd. This is taken care of by dividing the power distribution network into smaller grids.
2. *Inductance on the Power lines*: When a group of gates suddenly source/sink current from/to the power rails, the inductance on the rails oppose it, causing delay. Gridding reduces the inductance on each grid, and hence the delay, increasing/meeting the performance.
3. *Electron migration*: By gridding, all the smaller power grids are connected in parallel with each other dividing the average current among the grids. If there were no gridding or insufficient grids the power ring might be destroyed because of electron migration. This can be overcome by having wider metal slightly if at all because of the increased ac resistance due to skin effect.

More discussion on selecting the width of power lines and planning the power distribution network can be found in references [16-18].

After placing the blocks and power planning, the floorplan looks as shown in fig4.9. The original floorplan is developed by [J. Chen and J. Stine] which is modified to do P&R with the optimized (reduced height) standard cells and to accommodate multiple power domains.

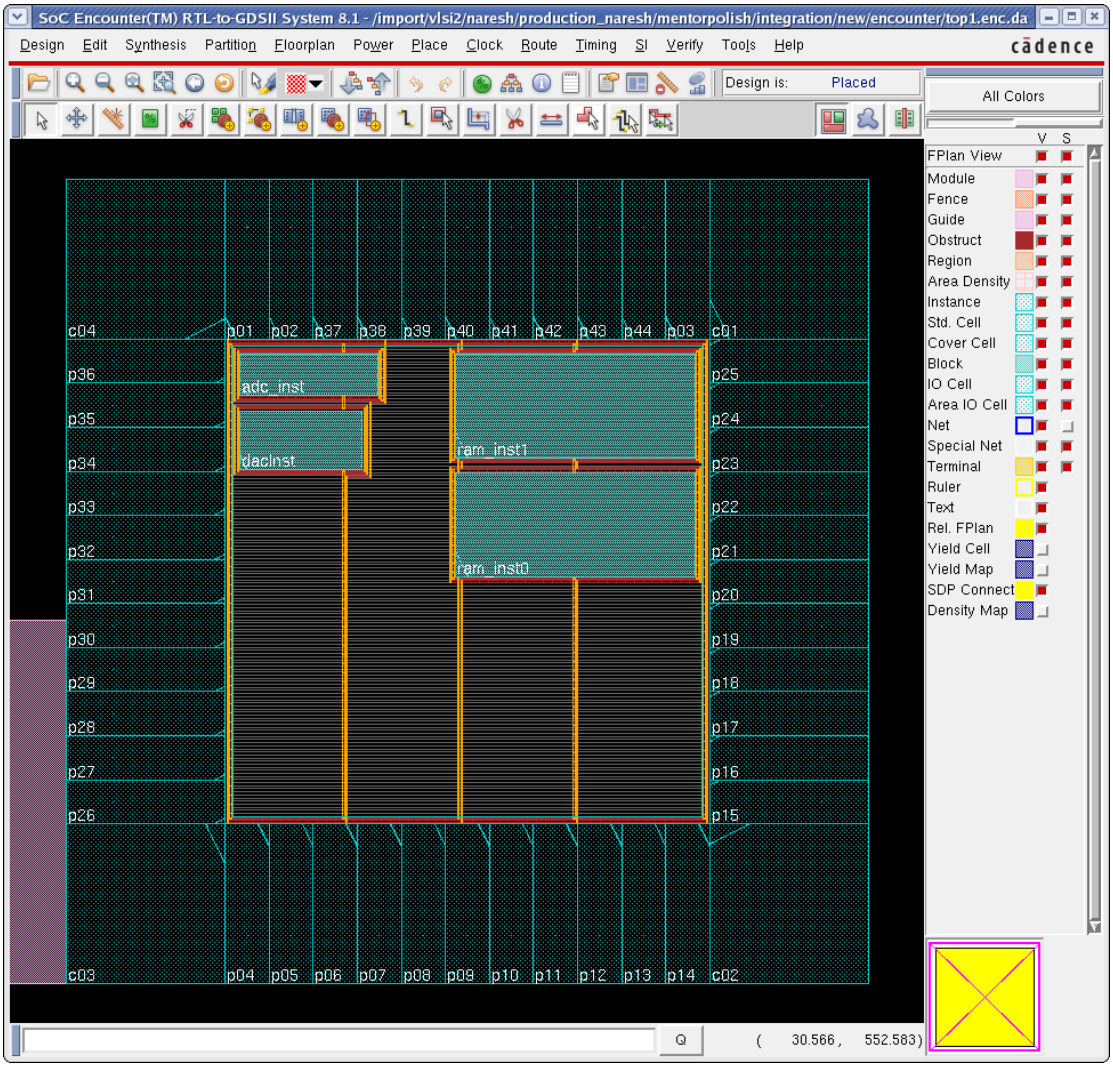


Fig 4.9: Final floorplan[2] (copyright 2005, 2010, Cadence Design Systems. Inc)

SAVE!! The floorplan can be saved so that if there any future changes in the timing or LEF files, we can just repeat the design import step with the new files and use the same floorplan to get to this point instead of repeating everything again. To save floorplan select *Design -> save -> floorplan.* The file will be saved with **.fp* extension.

Sroute

This step connects all the standard cell rows to the power rings and stripes and the power rings to the power pads. The analog blocks are connected to the pads manually.

The next steps that follow are:

1. Placing the Standard cells.
2. The First optimization phase (*Pre-Clock tree synthesis*) during which a *Trial route is* performed that estimates the amount of wiring and indicates how poorly timing requirements are being met. For small designs i.e. implementing a counter, we don't need a clock tree and it may well satisfy the timing requirements. But the PC-ASIC is a very large design and requires a clock tree.
3. *Clock tree synthesis* is done by the tool to overcome the skew problem.
4. Second optimization phase, (*Post-Clock tree synthesis*) to evaluate how the timing has improved after the clock tree has been established and optimized by adding buffers as needed.
5. Route the design.
6. *Post-route* optimization to improve the timing, by further buffer tweaking.
7. Adding Filler cells to fill the gaps between the standard cells in the rows.
8. Checking and fixing any design rule violations.

They have been integrated into a script developed by [J.Chen, I. Castellanos and J. Stine], and is used to complete the above steps.

When the design cannot meet the timing or routing errors result, this is a strong indication that a change in floorplan is required and the P&R process must be repeated. For example, we can reduce the *core utilization* parameter to place the standard cells more loosely resulting in additional space for buffering and routing. Or we might leave gaps between the standard cell rows to leave room for wiring or move the analog blocks providing improved routing opportunities.

After the P&R process has been completed and we see no routing and timing errors the *gds* (Graphic Database System) file and the Verilog netlist (obtained after the optimizations done by

Encounter tool) are saved and imported into the *Virtuoso* layout editor for physical verification of the final chip.

4.5 Physical Verification:

The tool *Calibre* from MGC has been used to complete the physical verification. DRC and LVS rules have been provided with the PDK. The design rule check could be completed with a few errors that were corrected manually. But for doing the layout vs. schematic, there were a few problems.

First, the structural Verilog netlist has to be converted into a SPICE netlist using the *Calibre* tool. The command to do this is:

v2lvs -v final.v -o source.net -s cells.sp -s0 gnd -s1 vdd

final.v – The structural Verilog netlist output from *Encounter.*

source.net –The SPICE netlist that is going to be generated from the above Verilog file.

cells.sp – The file that contains the SPICE netlists for all the standard cells. It will be included in the *source.net* file.

s0 and *s1* options define the gnd and vdd as global connections in the output *source.net* file.

source.net now has the top level SPICE netlist and the individual SPICE netlists for all the standard cells. This is because when we do LVS for a large design it is done hierarchically. Also we need to include the SPICE netlists for the analog blocks and the memory into this file.

Thus, *source.net* includes:

- ✓ Top level SPICE netlist
- ✓ SPICE netlists for all the standard cells

- ✓ SPICE netlists for the analog and memory blocks

This IC has multiple power names:

1.	Core (standard cells):	vdd	gnd
2.	ADC/DAC	vdda, vdda!	gnd!
3.	Memory	vdd!	gnd!
4.	IO	vddio	gnd

The '!' symbol used at the end of some of the sub-cells in the memory and analog blocks created many problems. The following statements must be included in the *source.net* file to solve the issues:

```
.GLOBAL vdd
.GLOBAL vdd!
.GLOBAL vdda
.GLOBAL vdda!
.GLOBAL vddio
.GLOBAL gnd
.GLOBAL gnd!
```

These statements ensured that the supply names are global and the verification tool *Calibre nmLVS* recognizes all of them, not depending on where they are in the hierarchy.

```
*.J vdd vdd!
*.J vdda vdda!
*.J gnd gnd!
```

The *.J statement connects any two nets so that they are interpreted to be the same. So we now have only 3 power supply names at the top level vdd, vdda, vddio and a common gnd irrespective of the inconsistencies in the names between the blocks and their sub-cells.

With these changes incorporated into the *source.net* file, it is compared with the layout using the Calibre tool and the design passed LVS.

CHAPTER V

RESULTS AND CONCLUSION

An optimized standard cell library is developed by redoing the layouts manually for the cell library designed earlier, for the FREEPDK45 process. The cell library has been characterized for timing, power and functionality data. It also contains the abstract views for doing the place and route. This library can be used for research in academic institutions or industry to test new architectures or implementing an ASIC. The cell library has been documented and the users may add new cells to the library if needed following the specifications set in *Chapter2*.

5.1 Evaluation of the results:

The timing results obtained are evaluated and compared with the linear delay model presented in *chapter3*.

Table 5.1: Timing characterization data for NOR3X1 gate

Ts[ns] / CL[pF]	0.015	0.03	0.045	0.06	0.075
5e-05	0.022	0.026	0.030	0.034	0.037
0.001	0.024	0.028	0.033	0.037	0.040
0.002	0.027	0.031	0.035	0.039	0.043
0.004	0.032	0.035	0.040	0.044	0.048
0.008	0.041	0.044	0.048	0.053	0.057

Table 5.1 shows the propagation delays (rounded to 3 decimal places) obtained for a NOR3X1 gate for different input rise/fall times (Ts) and loads (CL). The first row gives the parasitic delays for the gate. Subtracting these values from the delay values gives the effort delays. The plot showing how delay varies with the load for different input slew rates is shown in fig5.1:

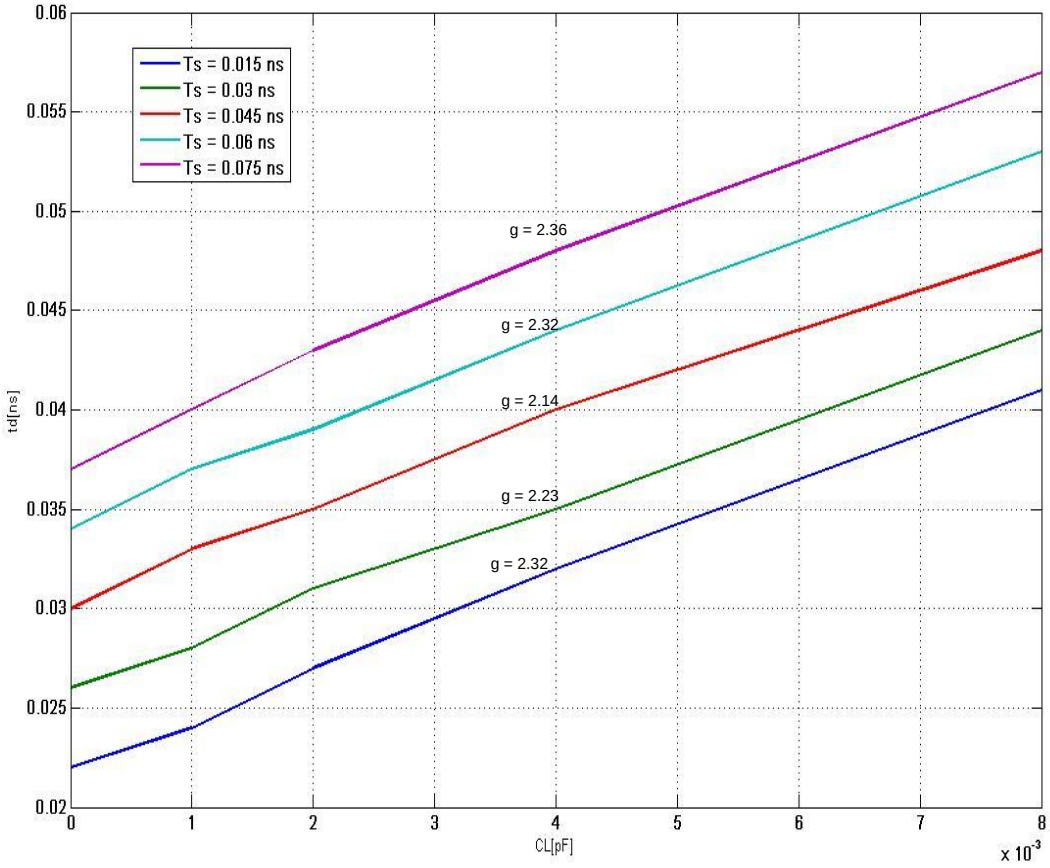


Fig 5.1: Delay vs. load

Observations from fig5.1:

1. The parasitic delay for input slew rate of 15 pS is 22 pS. This agrees with the value obtained through simulation in *section 3.4.3.2* which is 25pS. However, the slight difference is due to the fact that, the value obtained in *section 3.4.3.2* is for an input slew rate of 25ps.

2. The parasitic delays (y-intercept) through the gate increase with input slew rates. This is because the transistors in the pull-up/pull-down networks are not completely switched ON/OFF respectively until the input signal reaches its final value. So the leakage current acts against the charging/discharging of the output node.
3. The effort delays increase linearly with the increase in output load that has to be driven. The slope of the lines (g) gives the logical effort. This should be constant for a given gate independent of loads/input slew rates, which can be verified from the graph. Also it agrees with the theoretical value (7/3 = 2.33).

5.2 Comparison of results (new vs. existing):

The new standard cells (laid out manually) are compared with the previous cell library (generated by an automated tool):

1. **Physical representations:**

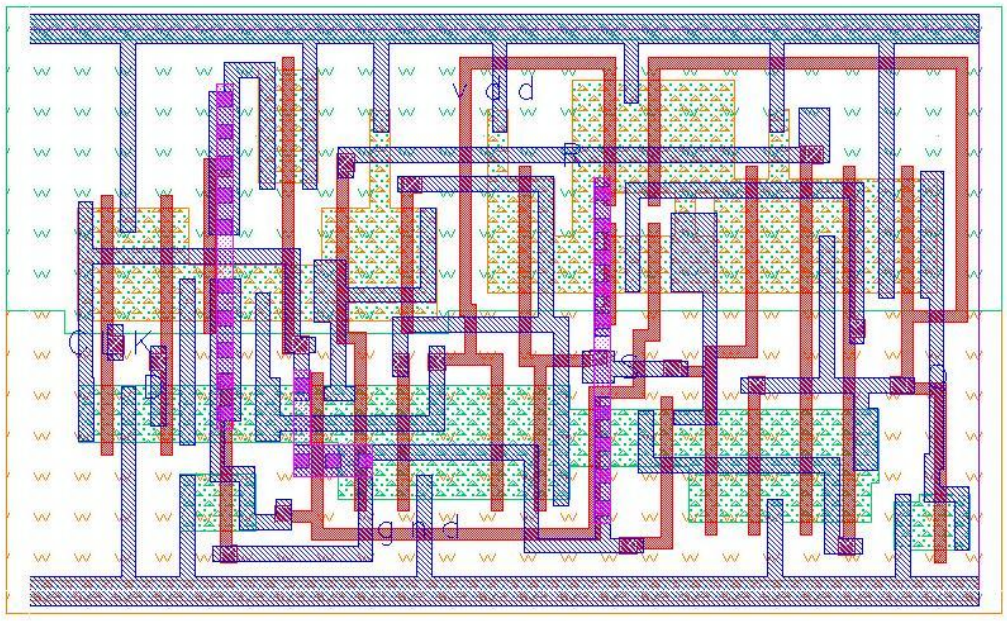


Fig 5.2 Layout generated by software tool for DFFSR cell.

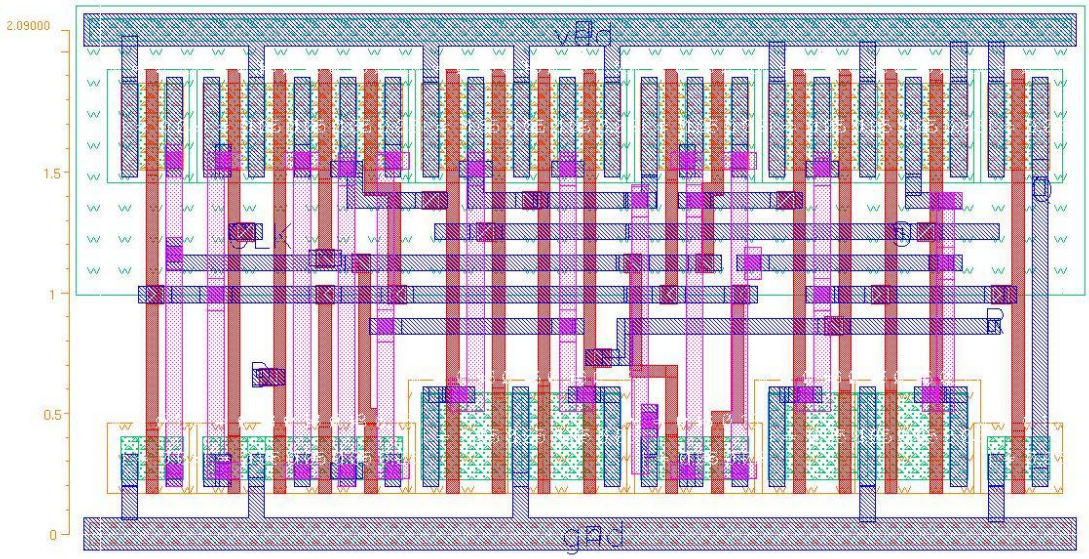


Fig 5.2 Layout done manually for DFFSR cell.

2. Reduction in height of the cells

The height of the standard cells has been reduced from *2.47 um* to *2.09 um*. The new cells are more compact and dense.

3. Improvement in timing:

The timing is expected to be improved for the new cells because of the optimization in the poly routing. But the results are found to be almost similar. This is because the resistance of the poly hasn't been modeled in the parasitic extraction rules file for this process. Hence the improvement in the delay couldn't be observed.

5.3 LVS problems for a multiple power domain ASIC when using *Calibre*[19]:

1. The goal is to be able to run LVS straight from the Verilog source obtained from the P&R tool (using *v2lvs* as the intermediary) and the SPICE library (standard cells, analog

and memory blocks). *v2lvs* by default provides only vdd and vss as .GLOBAL signals. But there are other supply names (vdd!, gnd!, vdda!, vdda) in addition to them in the SPICE library. Hence they have to be added manually after the *v2lvs* has output the SPICE netlist. The solution to make the process automated has to be found yet.

2. Also this needs to happen:

```
*.J vdd vdd!
*.J vdda vdda!
*.J gnd gnd!
```

v2lvs doesn't know that the above nets are connected up-front. Hence they have to be added manually.

The final issue is, that Verilog has modules for all the PAD* cells. As a result, v2lvs translates all of these as .SUBCKT statements in SPICE with X calls. But *Calibre* when extracting a layout doesn't create a call to a .SUBCKT statement if it doesn't have a device in it. The PADCORNER cells have no devices. So .SUBCKT statement never gets called for these four instances. When doing LVS, it results in an instance mismatch: 4 PADCORNER instances in the source, none in the layout. To overcome this, the PADCORNER X calls in the source are commented out. A possible and better solution might be creating dummy devices (devices for decoupling) in the layout so the PADCORNER sub-circuit actually gets called 4 times.

REFERENCES

[1] J. E. Stine, I. Castellanos, M. Wood, J. Henson, F. Love, W. R. Davis, P. D. Franzon, M. Bucher, S. Basavarajaiah, J. Oh, and R. Jenkal, "FreePDK: An Open-Source Variation-Aware Design Kit," in *Microelectronic Systems Education, 2007. MSE '07. IEEE International Conference on*, 2007, pp. 173-174.

[2] J. E. Stine, C. Jun, I. Castellanos, G. Sundararajan, M. Qayam, P. Kumar, J. Remington, and S. Sohoni, "FreePDK v2.0: Transitioning VLSI education towards nanometer variation-aware designs," in *Microelectronic Systems Education, 2009. MSE '09. IEEE International Conference on*, 2009, pp. 100-103.

[3] V. Jeyaraman, "Design, characterization and automation of an ultra-high-temperature standard cell library for harsh environments," in *Electrical and Computer Engineering*: Oklahoma State University, 2004, p. 124.

[4] U. Badam, "Development of a 5V digital cell library for use with the Peregrine semiconductor silicon-on-sapphire (SOS) process," in *Electrical and Computer Engineering*: Oklahoma State University, 2007, p. 88.

[5] S. Viswanathan, "Proposal for a 3.3v/5v low leakage high temperature digital cell library using stacked transistors," in *Electrical and Computer Engineering*: Oklahoma State University, 2007, p. 89.

[6] S. Viswanathan and C. Hutchens, *Design of Low Leakage High Temperature Digital Cell Libraries*: VDM Verlag, Dr.Muller Aktiengesllschaft & Co. KG, Dudweller Landstr. , 2007.

[7] "International Technology Roadmap for Semiconductors," Semiconductor Industry Association2003.

[8] C. Saint and J. Saint, *IC Mask Design, Essential Layout Techniques*, 1st ed.: McGraw-Hill, 2002.

[9] M. J. M. Pelgrom, H. P. Tuinhout, and M. Vertregt, "Transistor matching in analog CMOS applications," in *Electron Devices Meeting, 1998. IEDM '98 Technical Digest., International*, 1998, pp. 915-918.

[10] N. H. E. Weste and D. Harris, "CMOS VLSI Design: A Circuits and Systems Perspective," 3rd ed: Addison Wesley, 2004, pp. 786-800.

[11] E. Brunvand, *Digital VLSI Chip Design with Cadence and Synopsys CAD Tools*, 1st ed.: Addison Wesley, 2009.

[12] D. Stringfellow and J. Pedicone, "Decoupling Capacitance Estimation, Implementation, and Verification: *A Practical Approach for Deep Submicron SoCs*," in *Synopsys Users Group*, San Jose, 2007.

[13] W. Elmore, "The transient response of damped linear networks with particular regard to wideband amplifiers," *Applied Physics*, vol. 19, pp. 55-63, Jan 1948.

[14] M. G. Corporation, "A Quick Start to Calibre tools using an example design." vol. Software version 2010.1.

[15] R. Ahmed, "Design of 3.3v Digital standard cell library for LEON3," in *Electrical and Computer Engineering*: Oklahoma State University, 2009, p. 98.
[16] H. Shih-Hsu and W. Chu-Liao, "An effective floorplan-based power distribution network design methodology under reliability constraints," in *Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on*, 2002, pp. I-353-I-356 vol.1.
[17] M. S. Guggari, "Power Planning," 2006.
[18] A. Pathak, S. K. Mandal, R. K. Nagpal, and R. Malik, "Modelling and Analysis of Power-Ground Plane for High Speed VLSI System," in *India Conference (INDICON), 2009 Annual IEEE*, 2009, pp. 1-4.
[19] F. Dailami and D. Liddell, Private Communication ed.

APPPENDIX1

CHARACTERIZATION SETUP

Before we can start the Encounter library Characterizer we should have the following files under one directory.

1. Netlists of all the Standard cells with parasitics extracted.
2. elccfg
3. gpdk45nm.m
4. setup.ss

elccfg:

```
#Specify the environment variable settings.
EC_SIM_USE_LSF=1;
EC_SIM_LSF_CMD=" ";
EC_SIM_LSF_PARALLEL=10;
EC_SIM_TYPE="hspice";
EC_SIM_NAME="hspice";

#Specify the characterization inputs.
SUBCKT="stdcells.pex.netlist";
MODEL="gpdk45nm.m";
DESIGNS="AND2X1 AND2X2 AOI21X1 AOI22X1 BUFX2 BUFX4 CLKBUF1 CLKBUF2
CLKBUF3 DFFNEGX1 DFFPOSX1 DFFSR FAX1 HAX1 INVX1 INVX2 INVX4 INVX8 LATCH
MUX2X1 NAND2X1 NAND3X1 NOR2X1 NOR3X1 OAI21X1 OAI22X1 OR2X1 OR2X2 TBUFX1
TBUFX2 XNOR2X1 XOR2X1";
```

```
EXPAND="AND2X1 AND2X2 AOI21X1 AOI22X1 BUFX2 BUFX4 CLKBUF1 CLKBUF2
CLKBUF3 DFFNEGX1 DFFPOSX1 DFFSR FAX1 HAX1 INVX1 INVX2 INVX4 INVX8 LATCH
MUX2X1 NAND2X1 NAND3X1 NOR2X1 NOR3X1 OAI21X1 OAI22X1 OR2X1 OR2X2 TBUFX1
TBUFX2 XNOR2X1 XOR2X1";
SETUP="setup.ss";
PROCESS="typical";
```

In the environment setup we have chosen *hspice* simulator to carry out the simulations. So, it should be ensured the PATH variable in the shell includes the path to this tool. The first three lines in the *elccfg* file relates to a feature of ELC that lets us start servers on a bunch of machines and do the simulation on them during characterization. This feature is very useful to speed up the process when we have a large library.

SUBCKT: This specifies the SPICE subcircuit file. I've copied all the netlists of the standard cells into a single file "stdcells.pex.netlist".

MODEL: Specifies the model file for the transistors which is gpdk45nm.m for FREEPDK process.

DESIGNS: Specifies the name of the cells to be worked on.

EXPAND: This specifies the name of the cells that should be expanded and must be specified when characterizing hierarchical cells.

SETUP: Specifies the simulation setup file

PROCESS: Specifies the process corner of the setup file.

After having set up the above files, ELC can be started by typing *elc* at the terminal. The commands to run the tool are as follows:

db_open - opens a database.

db_prepare - installs the transistor models and the subckt descriptions into the database.

db_gsim - evaluates the transistor networks and generates the test vectors for characterization.

db_spice - simulates the design using the test vectors derived to extract the timing and power numbers using hspice.

db_output - outputs the cell library characterization results.

The output can be obtained in different formats. ALF-Advanced library format is an intermediate format to transfer data to other EDA tools. A more detailed explanation of the commands can be obtained from the Encounter Library Characterizer user guide.

Setup file:

```
Process typical{
        voltage = 1.1;
        temp = 27;
        Corner = "";
        Vtn = 0.471;
        Vtp = 0.423;
};

Signal std_cell {
        unit = REL;
        Vh=1.0 1.0;
        Vl=0.0 0.0;
        Vth=0.5 0.5;
        Vsh=0.8 0.8;
        Vsl=0.2 0.2;
};

Simulation std_cell{
```

Defines the *typical* process

REL indicates the values specified are a percentage of *logic high* and *logic low* values. *Vsh Vsl* represent the voltage levels for calculating the delays.

```
        transient = 0.1n 80n 10p;
        dc = 0.01 1.1 0.01;
        bisec = 6.0n 6.0n 0.05n;
        resistance = 10MEG;
};

Index X1{
        Slew = 0.025n 0.05n 0.075n 0.1n 0.125n;
        Load = 0.05f 1f 2f 4f 8f;
};

Index X2{
        Slew = 0.025n 0.05n 0.075n 0.1n 0.125n;
        Load = 0.004f 0.008f 0.0016f 0.0032f;
};


Index X4{
        Slew = 0.025n 0.05n 0.075n 0.1n 0.125n;
        Load = 0.008f 0.0016f 0.0032f 0.0064f;
};

Index X8{
        Slew = 0.025n 0.05n 0.075n 0.1n 0.125n;
        Load = 0.0016f 0.0032f 0.0064f 0.0128f;
};

Index Clk_Slew{
        bslew = 0.025n 0.05n 0.075n 0.1n 0.125n;
};

Group X1{
        CELL = *x1 ;
};

Group X2{
        CELL = *X2 ;
};

Group X4{
        CELL = *X4 ;
};

Group X8{
        CELL = *X8 ;
};
```

Setting up the index and slew values are explained in chapters 2,3, p11, 38.

All the cells ending in names x1, x2 etc are characterized with respect to the slew and load values specified in the INDEX1, INDEX2 etc respectively.

```
Margin m0 {
        setup   = 1.0 0.0 ;
        hold    = 1.0 0.0 ;
        release = 1.0 0.0 ;
        removal = 1.0 0.0 ;
        recovery = 1.0 0.0 ;
        width   = 1.0 0.0 ;
        delay   = 1.0 0.0 ;
        power   = 1.0 0.0 ;
        cap     = 1.0 0.0 ;
} ;
```

Specifies the correction factors. Final value = (measured value * 1.0) + 0.0

```
Nominal n0 {
        delay = 0.5 0.5 ;
        power = 0.5 0.5 ;
        cap   = 0.5 0.5 ;
} ;
```

Specifies the threshold values between which pin to pin delays, power are calculated.

```
set process(typical){
        simulation = std_cell;
        signal = std_cell;
        margin = m0;
        nominal = n0;
};

set index(typical){
        Group(X1) = X1;
        Group(X2) = X2;
        Group(X4) = X4;
        Group(X8) = X8;
        Group(Clk_Slew)  = Clk_Slew;
};
```

APPENDIX 2

CHARACTERIZATION DATA (HTML FORMAT)

AND2X1 (value: delay=typ, power=typ, check=typ, cap=typ)

Function

Y=(A&B)

Static Power:

When	Static Power [nW]
-	27.8069

Port:

Name	Direction
A	INPUT
B	INPUT
Y	OUTPUT

Name	Pin Capacitance [pF]		Internal Power [pJ]	
	Rise	Fall	Rise	Fall
A	0.000641961	0.000790923	0.000128	0.000535
B	0.00138115	0.000824827	0.0005	0.000494

Output Driving Strength

Name	Rise		Fall	
	Strength (sec/F)	Limit (pF)	Strength (sec/F)	Limit (pF)
Y	3967.54	0.230939	1766.7	0.230939

Link To Path

PATH	WHEN
(01A=>01Y)	-
(10A=>10Y)	-
(01B=>01Y)	-
(10B=>10Y)	-

(01A=>01Y)

DELAY [ns]

ts[ns] / cl[pF]	0.015	0.03	0.045	0.06	0.075
5e-05	0.022888	0.026517	0.029796	0.030854	0.031641
0.001	0.026561	0.029896	0.033344	0.034198	0.034764
0.002	0.029921	0.033325	0.036522	0.03802	0.038698
0.004	0.036389	0.03979	0.042832	0.044099	0.045069
0.008	0.049054	0.052214	0.055177	0.056138	0.056968

POWER [pW]

ts[ns] / cl[pF]	0.015	0.03	0.045	0.06	0.075
5e-05	0.002477	0.002693	0.00294	0.003082	0.003279
0.001	0.003018	0.003152	0.003441	0.003538	0.003672
0.002	0.003609	0.003702	0.004011	0.00406	0.004263
0.004	0.004756	0.004738	0.005035	0.005217	0.00539
0.008	0.007028	0.006939	0.007197	0.00734	0.007637

(10A=>10Y)

DELAY [ns]

ts[ns] / cl[pF]	0.015	0.03	0.045	0.06	0.075
5e-05	0.030989	0.035837	0.040553	0.045072	0.049647
0.001	0.034241	0.038915	0.043428	0.048427	0.052854
0.002	0.038027	0.04191	0.046197	0.051462	0.05548
0.004	0.043161	0.046696	0.051533	0.056098	0.060919
0.008	0.051245	0.055429	0.060045	0.064685	0.06969

POWER [pW]

ts[ns] / cl[pF]	0.015	0.03	0.045	0.06	0.075
5e-05	0.002936	0.00318	0.003405	0.003477	0.003641
0.001	0.00268	0.002577	0.002661	0.002737	0.003218
0.002	0.001998	0.002122	0.00224	0.002325	0.002517
0.004	0.000923	0.000951	0.001048	0.001217	0.001376
0.008	0.001501	0.00143	0.001382	0.001235	0.001028

(01B=>01Y)

DELAY [ns]

ts[ns] / cl[pF]	0.015	0.03	0.045	0.06	0.075
5e-05	0.025073	0.027251	0.029244	0.030473	0.031035
0.001	0.028938	0.031032	0.033153	0.034304	0.035223
0.002	0.032383	0.03466	0.036813	0.038027	0.038879
0.004	0.038951	0.041321	0.043382	0.044973	0.04588
0.008	0.051713	0.053873	0.056029	0.057469	0.058105

POWER [pW]

ts[ns] / cl[pF]	0.015	0.03	0.045	0.06	0.075
5e-05	0.00237	0.002417	0.002468	0.002541	0.002649
0.001	0.002932	0.002983	0.003131	0.003126	0.003179
0.002	0.003542	0.003578	0.003427	0.003765	0.003821
0.004	0.004699	0.004749	0.004858	0.005123	0.005073
0.008	0.00699	0.007074	0.007096	0.007273	0.007442

(10B=>10Y)

DELAY [ns]

ts[ns] / cl[pF]	0.015	0.03	0.045	0.06	0.075
5e-05	0.034032	0.038883	0.043784	0.048801	0.053191
0.001	0.037379	0.04228	0.046818	0.051852	0.056243
0.002	0.040245	0.045091	0.049467	0.054299	0.059145
0.004	0.045044	0.049525	0.05439	0.058967	0.064483
0.008	0.05371	0.058336	0.06262	0.067742	0.072492

POWER [pW]

ts[ns] / cl[pF]	0.015	0.03	0.045	0.06	0.075
5e-05	0.003649	0.003687	0.0041	0.004273	0.004515
0.001	0.003081	0.003131	0.00349	0.003588	0.003754
0.002	0.002421	0.002537	0.00281	0.002987	0.003
0.004	0.001445	0.001484	0.001678	0.001753	0.001943
0.008	0.000889	0.00082	0.000785	0.000612	0.000362

DFFNEGX1 (value: delay=typ, power=typ, check=typ, cap=typ)

Function

FLIPFLOP{

DATA=D

CLOCK=!CLK

Q=DS0000

QN=N_NET82_MM18_G

}

Q=DS0000

Static Power:

When	Static Power [nW]
-	57.6366

Port:

Pin	Direction	Signaltype	Polarity
CLK	INPUT	CLOCK	FALLING_EDGE
D	INPUT	DATA	-
Q	OUTPUT	-	-

Name	Pin Capacitance [pF]		Internal Power [pJ]	
	Rise	Fall	Rise	Fall
CLK	0.00748444	0.00752119	0.003939	0.008904
D	0.00124059	0.000861589	0.002572	0.003208

Output Driving Strength

Name	Rise		Fall	
	Strength (sec/F)	Limit (pF)	Strength (sec/F)	Limit (pF)
Q	1092.05	0.486759	2023.19	0.486759

Link To Path

PATH	WHEN
(10CLK=>01Q)	-
(10CLK=>10Q)	-

Link To Constraint

Type	Path
SETUP	(01D=>10CLK)
SETUP	(10D=>10CLK)
HOLD	(10CLK=>01D)
HOLD	(10CLK=>10D)
PULSEWIDTH	(01CLK=>10CLK)
PULSEWIDTH	(10CLK=>01CLK)

(10CLK=>01Q)

DELAY [ns]

ts[ns] / cl[pF]	0.06	0.24	0.48	0.9	1.2	1.8
5e-05	0.15909	0.220712	0.253671	0.330673	0.394863	0.489424
0.001	0.163508	0.225051	0.258183	0.334835	0.397757	0.493404
0.002	0.167347	0.229286	0.260706	0.338729	0.401164	0.493467
0.004	0.174225	0.236548	0.268644	0.345768	0.409201	0.501859
0.008	0.18733	0.249488	0.28359	0.353432	0.423051	0.517468

POWER [pW]

ts[ns] / cl[pF]	0.06	0.24	0.48	0.9	1.2	1.8
5e-05	0.014744	0.016327	0.016759	0.020559	0.022794	0.02676
0.001	0.015244	0.016749	0.017244	0.020904	0.023103	0.027231
0.002	0.015864	0.017367	0.017722	0.021368	0.023601	0.026916
0.004	0.016917	0.01851	0.018806	0.022323	0.024731	0.027978
0.008	0.019278	0.020874	0.020849	0.023727	0.026925	0.031242

Back To Path Index

(10CLK=>10Q)

DELAY [ns]

ts[ns] / cl[pF]	0.06	0.24	0.48	0.9	1.2	1.8
5e-05	0.065845	0.108134	0.164478	0.245956	0.279957	0.373157
0.001	0.069159	0.112977	0.167775	0.246802	0.285841	0.375903
0.002	0.072174	0.117711	0.170645	0.247716	0.291536	0.378702
0.004	0.077683	0.125355	0.175877	0.249615	0.300872	0.384251
0.008	0.087676	0.13743	0.186763	0.253738	0.306952	0.410924

POWER [pW]

ts[ns] / cl[pF]	0.06	0.24	0.48	0.9	1.2	1.8
5e-05	0.012265	0.014455	0.019401	0.023419	0.030163	0.039802
0.001	0.011696	0.014372	0.018911	0.022829	0.029822	0.038988
0.002	0.010804	0.013664	0.018138	0.022212	0.028925	0.038184
0.004	0.009836	0.012561	0.016422	0.021218	0.027485	0.036181
0.008	0.007396	0.009884	0.013846	0.018696	0.024524	0.034381

Timing Constraints

SETUP(01D=>10CLK)

re [ns] / co [ns]	0.015	0.03	0.045	0.06	0.075
0.015	0.09375	0.08125	0.06875	0.05625	0.1
0.03	0.10625	0.09375	0.08125	0.06875	0.1125
0.045	0.0625	0.10625	0.09375	0.08125	0.06875
0.06	0.075	0.0625	0.10625	0.09375	0.08125
0.075	0.0875	0.075	0.11875	0.10625	0.09375

Timing Constraints

SETUP(10D=>10CLK)

re [ns] / co [ns]	0.015	0.03	0.045	0.06	0.075
0.015	0.15	0.1375	0.125	0.1125	0.15625
0.03	0.10625	0.09375	0.1375	0.125	0.1125
0.045	0.11875	0.10625	0.15	0.1375	0.125
0.06	0.13125	0.11875	0.10625	0.15	0.1375
0.075	0.0875	0.13125	0.11875	0.10625	0.15

Timing Constraints

HOLD(10CLK=>01D)

re [ns] / co [ns]	0.015	0.03	0.045	0.06	0.075
0.015	-0.01875	-0.00625	0.00625	-0.0375	-0.025
0.03	-0.03125	-0.01875	-0.00625	0.00625	-0.0375
0.045	-0.04375	-0.03125	-0.01875	-0.00625	-0.05
0.06	-0	-0.04375	-0.03125	-0.01875	-0.00625
0.075	-0.0125	0	-0.04375	-0.03125	-0.01875

Timing Constraints

HOLD(10CLK=>10D)

re [ns] / co [ns]	0.015	0.03	0.045	0.06	0.075
0.015	0.0375	0.05	0.00625	0.01875	0.03125
0.03	0.025	0.0375	0.05	0.00625	0.01875
0.045	0.0125	0.025	0.0375	0.05	0.00625
0.06	0.05625	0.0125	0.025	0.0375	0.05
0.075	0.04375	0.05625	0.0125	0.025	0.0375

Timing Constraints

PULSEWIDTH(01CLK=>10CLK)

	6e-11	2.4e-10	4.8e-10	9e-10	1.2e-09	1.8e-09
	0.155355	0.184674	0.177585	0.176364	0.160871	0.128285

Timing Constraints

PULSEWIDTH(10CLK=>01CLK)

	6e-11	2.4e-10	4.8e-10	9e-10	1.2e-09	1.8e-09
	0.134222	0.196873	0.226511	0.298716	0.358855	0.450921

APPENDIX 3

LEF Technology header

This section includes the technology information required by the P&R tool when it places and routes the design. The information includes the design rules for different layers (metals) and vias for the FREEPDK process.

```
VERSION 5.6 ;
BUSBITCHARS "[]" ;
DIVIDERCHAR "/" ;

PROPERTYDEFINITIONS
LAYER contactResistance REAL ;
MACRO viewNameList STRING ;
END PROPERTYDEFINITIONS

UNITS
DATABASE MICRONS 2000 ;
END UNITS
MANUFACTURINGGRID 0.0025 ;
LAYER poly
  TYPE MASTERSLICE ;
END poly

LAYER contact
  TYPE CUT ;
  SPACING 0.075 ;
  PROPERTY contactResistance 10.5 ;
END contact

LAYER metal1
  TYPE ROUTING ;
  DIRECTION HORIZONTAL ;
  PITCH 0.19 ;
  WIDTH 0.065 ;
  SPACING 0.065 ;
  RESISTANCE RPERSQ 0.38 ;
END metal1

LAYER via1
  TYPE CUT ;
  SPACING 0.075 ;
  PROPERTY contactResistance 5.69 ;
END via1

LAYER metal2
```

```
  TYPE ROUTING ;
  DIRECTION VERTICAL ;
  PITCH 0.19 ;
  WIDTH 0.07 ;
  SPACING 0.075 ;
  RESISTANCE RPERSQ 0.25 ;
END metal2

LAYER via2
  TYPE CUT ;
  SPACING 0.085 ;
  PROPERTY contactResistance 11.39 ;
END via2

LAYER metal3
  TYPE ROUTING ;
  DIRECTION HORIZONTAL ;
  PITCH 0.19 ;
  WIDTH 0.07 ;
  SPACING 0.07 ;
  RESISTANCE RPERSQ 0.25 ;
END metal3

LAYER via3
  TYPE CUT ;
  SPACING 0.085 ;
  PROPERTY contactResistance 16.73 ;
END via3

LAYER metal4
  TYPE ROUTING ;
  DIRECTION VERTICAL ;
  PITCH 0.285 ;
  WIDTH 0.14 ;
  SPACING 0.14 ;
  RESISTANCE RPERSQ 0.25 ;
END metal4

LAYER via4
  TYPE CUT ;
  SPACING 0.16 ;
  PROPERTY contactResistance 21.44 ;
END via4

LAYER metal5
  TYPE ROUTING ;
  DIRECTION HORIZONTAL ;
  PITCH 0.285 ;
  WIDTH 0.14 ;
  SPACING 0.14 ;
  RESISTANCE RPERSQ 0.25 ;
```

```
END metal5

LAYER via5
  TYPE CUT ;
  SPACING 0.16 ;
  PROPERTY contactResistance 24.08 ;
END via5

LAYER metal6
  TYPE ROUTING ;
  DIRECTION VERTICAL ;
  PITCH 0.285 ;
  WIDTH 0.14 ;
  SPACING 0.14 ;
  RESISTANCE RPERSQ 0.25 ;
END metal6

LAYER via6
  TYPE CUT ;
  SPACING 0.16 ;
  PROPERTY contactResistance 11.39 ;
END via6

LAYER metal7
  TYPE ROUTING ;
  DIRECTION HORIZONTAL ;
  PITCH 0.855 ;
  WIDTH 0.4 ;
  SPACING 0.44 ;
  RESISTANCE RPERSQ 0.25 ;
END metal7

LAYER via7
  TYPE CUT ;
  SPACING 0.44 ;
  PROPERTY contactResistance 5.69 ;
END via7

LAYER metal8
  TYPE ROUTING ;
  DIRECTION VERTICAL ;
  PITCH 0.855 ;
  WIDTH 0.4 ;
  SPACING 0.44 ;
  RESISTANCE RPERSQ 0.25 ;
END metal8

LAYER via8
  TYPE CUT ;
  SPACING 0.44 ;
  PROPERTY contactResistance 16.73 ;
```

```
END via8

LAYER metal9
  TYPE ROUTING ;
  DIRECTION HORIZONTAL ;
  PITCH 1.71 ;
  WIDTH 0.8 ;
  SPACING 0.8 ;
  RESISTANCE RPERSQ 0.21 ;
END metal9

LAYER via9
  TYPE CUT ;
  SPACING 0.88 ;
  PROPERTY contactResistance 21.44 ;
END via9

LAYER metal10
  TYPE ROUTING ;
  DIRECTION VERTICAL ;
  PITCH 1.71 1.71 ;
  WIDTH 0.8 ;
  SPACING 0.8 ;
  RESISTANCE RPERSQ 0.21 ;
END metal10

LAYER OVERLAP
  TYPE OVERLAP ;
END OVERLAP

VIA M2_M1_via DEFAULT
  LAYER metal1 ;
    RECT -0.0675 -0.0325 0.0675 0.0325 ;
  LAYER via1 ;
    RECT -0.0325 -0.0325 0.0325 0.0325 ;
  LAYER metal2 ;
    RECT -0.035 -0.0675 0.035 0.0675 ;
END M2_M1_via

VIA M3_M2_via DEFAULT
  LAYER metal2 ;
    RECT -0.035 -0.07 0.035 0.07 ;
  LAYER via2 ;
    RECT -0.035 -0.035 0.035 0.035 ;
  LAYER metal3 ;
    RECT -0.07 -0.035 0.07 0.035 ;
END M3_M2_via

VIA M4_M3_via DEFAULT
  LAYER metal3 ;
    RECT -0.07 -0.035 0.07 0.035 ;
```

```
  LAYER via3 ;
    RECT -0.035 -0.035 0.035 0.035 ;
  LAYER metal4 ;
    RECT -0.07 -0.07 0.07 0.07 ;
END M4_M3_via

VIA M5_M4_via DEFAULT
  LAYER metal4 ;
    RECT -0.07 -0.07 0.07 0.07 ;
  LAYER via4 ;
    RECT -0.07 -0.07 0.07 0.07 ;
  LAYER metal5 ;
    RECT -0.07 -0.07 0.07 0.07 ;
END M5_M4_via

VIA M6_M5_via DEFAULT
  LAYER metal5 ;
    RECT -0.07 -0.07 0.07 0.07 ;
  LAYER via5 ;
    RECT -0.07 -0.07 0.07 0.07 ;
  LAYER metal6 ;
    RECT -0.07 -0.07 0.07 0.07 ;
END M6_M5_via

VIA M7_M6_via DEFAULT
  LAYER metal6 ;
    RECT -0.07 -0.07 0.07 0.07 ;
  LAYER via6 ;
    RECT -0.07 -0.07 0.07 0.07 ;
  LAYER metal7 ;
    RECT -0.2 -0.2 0.2 0.2 ;
END M7_M6_via

VIA M8_M7_via DEFAULT
  LAYER metal7 ;
    RECT -0.2 -0.2 0.2 0.2 ;
  LAYER via7 ;
    RECT -0.2 -0.2 0.2 0.2 ;
  LAYER metal8 ;
    RECT -0.2 -0.2 0.2 0.2 ;
END M8_M7_via

VIA M9_M8_via DEFAULT
  LAYER metal8 ;
    RECT -0.2 -0.2 0.2 0.2 ;
  LAYER via8 ;
    RECT -0.2 -0.2 0.2 0.2 ;
  LAYER metal9 ;
    RECT -0.4 -0.4 0.4 0.4 ;
END M9_M8_via
```

```
VIA M10_M9_via DEFAULT
  LAYER metal9 ;
    RECT -0.4 -0.4 0.4 0.4 ;
  LAYER via9 ;
    RECT -0.4 -0.4 0.4 0.4 ;
  LAYER metal10 ;
    RECT -0.4 -0.4 0.4 0.4 ;
END M10_M9_via

VIA M2_M1_viaB DEFAULT
  LAYER metal1 ;
    RECT -0.0675 -0.0325 0.0675 0.0325 ;
  LAYER via1 ;
    RECT -0.0325 -0.0325 0.0325 0.0325 ;
  LAYER metal2 ;
    RECT -0.0675 -0.035 0.0675 0.035 ;
END M2_M1_viaB

VIA M2_M1_viaC DEFAULT
  LAYER metal1 ;
    RECT -0.0325 -0.0675 0.0325 0.0675 ;
  LAYER via1 ;
    RECT -0.0325 -0.0325 0.0325 0.0325 ;
  LAYER metal2 ;
    RECT -0.035 -0.0675 0.035 0.0675 ;
END M2_M1_viaC

VIA M3_M2_viaB DEFAULT
  LAYER metal2 ;
    RECT -0.035 -0.07 0.035 0.07 ;
  LAYER via2 ;
    RECT -0.035 -0.035 0.035 0.035 ;
  LAYER metal3 ;
    RECT -0.035 -0.07 0.035 0.07 ;
END M3_M2_viaB

VIA M3_M2_viaC DEFAULT
  LAYER metal2 ;
    RECT -0.07 -0.035 0.07 0.035 ;
  LAYER via2 ;
    RECT -0.035 -0.035 0.035 0.035 ;
  LAYER metal3 ;
    RECT -0.07 -0.035 0.07 0.035 ;
END M3_M2_viaC

VIA M4_M3_viaB DEFAULT
  LAYER metal3 ;
    RECT -0.035 -0.07 0.035 0.07 ;
  LAYER via3 ;
    RECT -0.035 -0.035 0.035 0.035 ;
  LAYER metal4 ;
```

```
    RECT -0.07 -0.07 0.07 0.07 ;
END M4_M3_viaB

VIARULE M2_M1 GENERATE
  LAYER metal1 ;
    ENCLOSURE 0 0.035 ;
  LAYER metal2 ;
    ENCLOSURE 0 0.035 ;
  LAYER via1 ;
    RECT -0.0325 -0.0325 0.0325 0.0325 ;
    SPACING 0.14 BY 0.14 ;
END M2_M1

VIARULE M3_M2 GENERATE
  LAYER metal2 ;
    ENCLOSURE 0 0.035 ;
  LAYER metal3 ;
    ENCLOSURE 0 0.035 ;
  LAYER via2 ;
    RECT -0.035 -0.035 0.035 0.035 ;
    SPACING 0.155 BY 0.155 ;
END M3_M2

VIARULE M4_M3 GENERATE
  LAYER metal3 ;
    ENCLOSURE 0 0.035 ;
  LAYER metal4 ;
    ENCLOSURE 0 0 ;
  LAYER via3 ;
    RECT -0.035 -0.035 0.035 0.035 ;
    SPACING 0.155 BY 0.155 ;
END M4_M3

VIARULE M5_M4 GENERATE
  LAYER metal4 ;
    ENCLOSURE 0 0 ;
  LAYER metal5 ;
    ENCLOSURE 0 0 ;
  LAYER via4 ;
    RECT -0.07 -0.07 0.07 0.07 ;
    SPACING 0.3 BY 0.3 ;
END M5_M4

VIARULE M6_M5 GENERATE
  LAYER metal5 ;
    ENCLOSURE 0 0 ;
  LAYER metal6 ;
    ENCLOSURE 0 0 ;
  LAYER via5 ;
    RECT -0.07 -0.07 0.07 0.07 ;
    SPACING 0.3 BY 0.3 ;
```

```
END M6_M5

VIARULE M7_M6 GENERATE
  LAYER metal6 ;
    ENCLOSURE 0 0 ;
  LAYER metal7 ;
    ENCLOSURE 0.13 0.13 ;
  LAYER via6 ;
    RECT -0.07 -0.07 0.07 0.07 ;
    SPACING 0.3 BY 0.3 ;
END M7_M6

VIARULE M8_M7 GENERATE
  LAYER metal7 ;
    ENCLOSURE 0 0 ;
  LAYER metal8 ;
    ENCLOSURE 0 0 ;
  LAYER via7 ;
    RECT -0.2 -0.2 0.2 0.2 ;
    SPACING 0.8 BY 0.8 ;
END M8_M7

VIARULE M9_M8 GENERATE
  LAYER metal8 ;
    ENCLOSURE 0 0 ;
  LAYER metal9 ;
    ENCLOSURE 0.2 0.2 ;
  LAYER via8 ;
    RECT -0.2 -0.2 0.2 0.2 ;
    SPACING 0.8 BY 0.8 ;
END M9_M8

VIARULE M10_M9 GENERATE
  LAYER metal9 ;
    ENCLOSURE 0 0 ;
  LAYER metal10 ;
    ENCLOSURE 0 0 ;
  LAYER via9 ;
    RECT -0.4 -0.4 0.4 0.4 ;
    SPACING 1.6 BY 1.6 ;
END M10_M9

VIARULE M1_POLY GENERATE
  LAYER poly ;
    ENCLOSURE 0 0 ;
  LAYER metal1 ;
    ENCLOSURE 0 0.035 ;
  LAYER contact ;
    RECT -0.0325 -0.0325 0.0325 0.0325 ;
    SPACING 0.14 BY 0.14 ;
END M1_POLY
```

```
SPACING
  SAMENET metal1 metal1 0.065 ;
  SAMENET metal10 metal10 0.8 ;
  SAMENET metal2 metal2 0.07 ;
  SAMENET metal3 metal3 0.07 ;
  SAMENET metal4 metal4 0.14 ;
  SAMENET metal5 metal5 0.14 ;
  SAMENET metal6 metal6 0.14 ;
  SAMENET metal7 metal7 0.44 ;
  SAMENET metal8 metal8 0.44 ;
  SAMENET metal9 metal9 0.8 ;
END SPACING

SITE CoreSite
  CLASS CORE ;
  SIZE 0.19 BY 2.09 ;
END CoreSite
```

VITA

Naresh Anne

Candidate for the Degree of

Master of Science

Thesis: DESIGN AND CHARACTERIZATION OF A STANDARD CELL LIBRARY FOR THE FREEPDK45 PROCESS

Major Field: Electrical and Computer Engineering

Biographical:

Education:

Completed the requirements for the Master of Science in Electrical and Computer Engineering at Oklahoma State University, Stillwater, Oklahoma in December, 2010.

Completed the requirements for the Bachelor of Engineering in Electronics and Communication Engineering at Andhra University, Visakhapatnam, Andhra Pradesh, India in 2008.

Experience:

Worked as a Graduate Research Assistant in Mixed Signal VLSI Design Lab in Department of Electrical & Computer Engineering at Oklahoma State University from August, 2009 to December, 2010.

Name: Naresh, Anne Date of Degree: December, 2010

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: DESIGN AND CHARACTERIZATION OF A STANDARD CELL LIBRARY FOR THE FREEPDK45 PROCESS

Pages in Study: 91 Candidate for the Degree of Master of Science

Major Field: Electrical and Computer Engineering

Scope and Method of Study:

The scope of this thesis can be summarized to two aspects. Firstly, to develop a design methodology to do layouts for a standard cell library to ensure no electrical and functional errors occur due to poor design of the cells when an IC is built out of them using CAD tools. The cell library has been built for the FREEPDK45 process following these rules and can be used for testing the emerging architectures in VLSI and research in academic institutions. Secondly, to establish procedures for characterizing a given cell library using *Encounter library characterizer tool* from *Cadence* to yield correct results. The factors that determine the timing of the cells are studied and the setup to yield accurate results for the cell library developed has been presented. This library has been used in building a mixed signal integrated circuit, and the problems in completing the final physical verification (DRC and LVS) have been studied.

Findings and Conclusions:

The Cell library developed for the FREEPDK process has been characterized and abstracted. The library is tested for its structural correctness when input to CAD tools to construct an integrated circuit. The IC built has been verified to be DRC and LVS clean. The design rules established for building standard cell layouts can be used as a reference manual for designing any other library. The process of characterization is automated and made easy by the *Encounter library characterizer* tool. However, correctly setting up the tool is very important in yielding correct results. The factors that determine this setup have been explored and documented which can also be used as a reference for characterizing any given cell library. The problems while doing LVS for an IC with multiple power domains using *Calibre* tool and the solution to eliminate them have been documented as well.

ADVISER'S APPROVAL: Dr. Chris Hutchens