Digital Implementation of a Synchronous First-in First-out (FIFO) using CAD tools

A Thesis presented

by

Aseem Gupta

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Master of Science

in

Electrical & Computer Engineering

Stony Brook University

August 2015

Stony Brook University

The Graduate School

Aseem Gupta

We, the thesis committee for the above candidate for the

Master of Science degree, hereby recommend

acceptance of this thesis

Prof Gianluigi De Geronimo
Adjunct Faculty, Stony Brook University
Scientist, Brookhaven National Laboratory

Prof Milutin Stanacevic
Associate Professor, Department of Electrical & Computer Engineering

This thesis is accepted by the Graduate School

Charles Taber
Dean of the Graduate School

Abstract of the Thesis

Digital Implementation of a Synchronous First-in First-out (FIFO) using CAD tools

by

Aseem Gupta

Master of Science

in

Electrical & Computer Engineering

Stony Brook University, New York

2015

Over the past couple of decades, digital design has prospered many miles. The availability of advanced EDA tools help cut design times, debugging issues and time-to-market (TTM). Understanding the digital design flow has now become crucial and many IP cores are being built in no time with use of advanced tools. The tools for front-end, back-end simulations have given industry freedom to experiment new complex designs.

The productivity of a digital designer is also a function of his ability to step up of a ramp and use the EDA tools to achieve faster designs and get the job done. Tools such as for synthesis, place and route, and timing verification have evolved over times with many variations and different command tools. Thus, the designer has to adjust with different and multiple UI's (user interfaces). In general industrial design, large CAD (Computer-Aided Design) teams are needed to provide such smoother flow control and results gathering capabilities via extensive scripting[5]. The thesis is based on learning and getting hands-on experience of new cutting-edge tools for faster designs.

First-in First-out (FIFO) design is crucial where the data has to be passed across different clock domains. Synchronous FIFO is used for first-in first-out read/write operation through a single clock port. This thesis describes the digital implementation of a synchronous FIFO with front-end and back-end flow using mostly Cadence Design Systems (CDS) and Mentor Graphics Corporation (MGC) EDA tools. Front-end involves logic design and simulation, logic synthesis and functional verification. Floorplanning, automatic placing and routing, clock-tree synthesis, timing closure and physical verification were performed as back-end design steps.

The ionizing radiations on a semiconductor device can cause bits to flip thereby changing the functionality of the digital IC causing a phenomenon called Single Event Upset (SEU). Thus, digital integrated circuits used for applications which expose them to radiations need to be SEU-tolerant. Many radiation-hardened-by-design (RHBD) techniques have been developed and one such technique is discussed in this thesis. This technique is called DICE (Dual-Interlocked Cell Storage).

As part of my work, drawing layouts and getting familiarized with the Process Design Kit (PDK) was necessary to complete a SEU-tolerant Flip Flop layout. In this process, TSMC 65nm, 130nm and IBM 130nm PDK's were studied and standard cells layouts were drawn.

Dedicated to my professor Gianluigi and my parents . . .

Contents

List of Figures

Acknowledgements

I, Aseem Gupta, hereby thank my advisor Prof. Gianluigi De Geronimo for his consistent efforts in progress of the project. He has guided me at each step on-course the duration of my thesis and his intellect has inspired me to do hard work which has reaped results. I also thank with immense love my parents who have continuously supported me, motivated and encouraged me to pursue my interests and live my dreams. This thesis would not have been completed without contributions from all these people. Last but not the least, I thank God for giving me the mental and physical strengths to do my work and making me able to complete the project....

Chapter 1

Digital VLSI Design Flow

1.1 Introduction: RTL-GDSII Design Flow

The gadget world has grown tremendously over last 15 years. Thanks to the ever-growing silicon industry, we are now living in a much more advanced and safer world.

The development of a silicon chip used in electronic devices starts with sand (silica). This silicon, a semiconductor element, along with its compounds primarily make the transistor we use in integrated circuits.

It is so remarkable that we have been able to use sand to reach to the moon. Electronic design has been growing with development of automated EDA (Electronic Design Automation) tools.

With feature size shrinking as per Moore's Law, the design has grown in functionality and complexity. Number of transistors have been increasing giving designers to play with trade-offs such as power, performance and area.

Digital design continues to evolve. The beauty of the digital design is its levels of abstraction. Today, we can design and simulate functionality at much higher level of abstraction which gives us more freedom to incorporate complex functionalities.

Figure 1.1 describes the VLSI Design Flow-

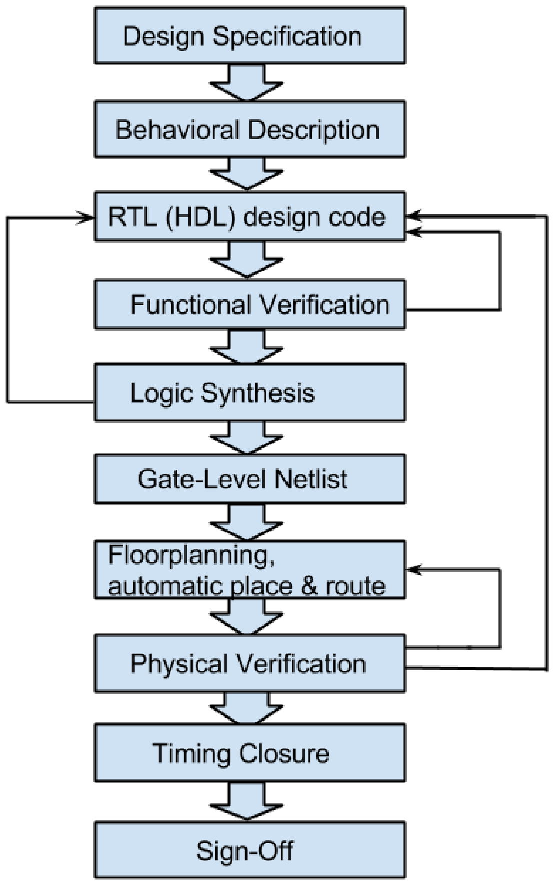


FIGURE 1.1: VLSI Design Flow

The flow shown in figure can be divided into two broad categories-

1. Front-end Design
2. Back-end Design

Standard-Cell design methodology has been chosen. This methodology uses EDA tools to automate the design flow.

Standard Cells are logic cells such as inverter, NAND, Flip Flops etc that we want to treat as black boxes during synthesis (memory blocks, custom body-bias circuits etc) and IO pad cells. All standard cells are of same height but different widths.

1.2 Front-end Steps

Front-end design includes-

A. System Specification & Architecture

B. HDL coding & Behavioral Simulation

C. Synthesis & Gate-level Simulation

Step1 - In any design, specifications of the product are written first. Specifications describe abstractly the functionality, interface, and overall architecture of the digital circuit to be designed.

Step2 - A behavioral description is then created to analyze the design in terms of functionality, performance, compliance to standards, and other high-level issues. Behavioral descriptions can be written in any Hardware Description Language (HDL)[6]. **High-level synthesis** (HLS) design using HLS tools allows us to build design in C, C++ at much higher level of abstraction.

Step3 - Logic Design

RTL (Register-Transfer Level) design starts now. A digital RTL design engineer would look at the behavioral description and manually write the RTL code in HDL. From this point onwards, the flow is fully automated using CAD tools[6].

For specific cases and complex designs, a verification engineer would simultaneously look at the behavioral description and come up with his own verification methodology independent of the RTL design created by the RTL designer.

Now, there are various HDL's to choose from -

1. VHDL (Very High Speed IC Hardware Description Language)
2. Verilog
3. System Verilog

It depends on RTL (Register-transfer level) code writer whichsoever HDL is chosen for design. Each language has its pros and cons. In this thesis, SystemVerilog has been chosen in the design of FIFO. The code could be written in any editor window such as textedit or gedit.

Any code that is synthesizable is the RTL code. Designs using the RTL specify the characteristics of a circuit by operations and the transfer of data between the registers. An explicit clock is used.
RTL design contains exact timing bounds:operations are scheduled to occur at certain times.

The code has to be simulated to verify the correct functionality of the design. There are many logic simulation tools to see the waveforms of the signals and simulate the design for which the code has been written such as -

1. ModelSim by Mentor Graphics
2. Incisive Enterprise Simulator (SimVision) by Cadence Design Systems
3. Vivado Design IC Suite by Xilinx

4. Synopsys VCS

Step4 - Logic Synthesis

Once the design is successfully simulated and verified (Functional Verification) for its correct functionality, it is ready to be synthesized. This process is called **Logic Synthesis**.

Logic Synthesis tools convert RTL description to a gate-level netlist. Some synthesis tools are -

1. Cadence RTL RC Compiler

2. Synopsys Design Compiler

A standard cell design methodology makes use of standard cell library of a particular process technology and synthesizes the design. Synthesizing the design involves use of standard cells, constraint file, timing files of the technology used and produces a gate-level netlist made out of logic gates such as- AND, INV, Flip-Flop, OR etc.

The gate-level netlist is obtained after running the synthesis. A gate-level netlist is a description of the circuit in terms of gates and connections between them. This gate-level netlist is again simulated using simulation softwares to verify the correct functionality of the design. This sub-step is called **post-synthesis gate-level simulation**.

All the above steps are discussed in detail in Chapter2.

1.3 Back-end Steps

Once the gate-level netlist is obtained, we are ready to layout (physical representation) the design.

All the steps of Back-end design are performed. These steps include -

1. Design Import - The necessary files such as the gate-level netlist (.v), timing information files (.lib), library exchange format (.lef) files, signal integrity (.cdb files) etc are imported into the back-end tool.

2. Floorplanning - The designer is required to determine die-size by arrangement of the IO's on pad frame and padframe to core. This step is a preparation step for power supplies; placement of IO pads, hard macros (RAM, PLL) and standard cells.

3. Power Planning and Routing - Power supplies to the core rows and Macros are specified. Designs vary in the physical location and the width of the supply lines. The tool will create connections between the core rows and core rings and the core rows to the core supply pads.

4. Cell Placement - Gate-level netlist is used to place the standard cells on the floorplan.

5. Clock-Tree Synthesis - After cell placement, placement of registers is known and the clock tree can be synthesized. Constraints like skew, slew, rise time and fall time can be specified in the clock-tree specification file (.ctstch). Clock Buffers and DECAPS are placed in the gaps of the floorplan[7].

6. Signal Routing - Signals as specified in the gate-level netlist need to be routed to complete physical placement of the design.

7. Timing Verification- The tool will try to optimize the timing by optimal routing i.e. the signals will be assembled by wires that span over several layers.

8. Physical Verification - The design is physically verified for DRC, ERC, antenna and LVS violations.

9. GDS Export - The final design is exported in GDSII format (Graphic Design Systems) format. This file contains the layout design in form of polygons and is sent out to the foundry to fabricate the design.

For a fully developed flow, it is recommended to include -

1. RC Extraction
2. Signal Integrity Check
3. Cross-talk
4. IR Drop Analysis

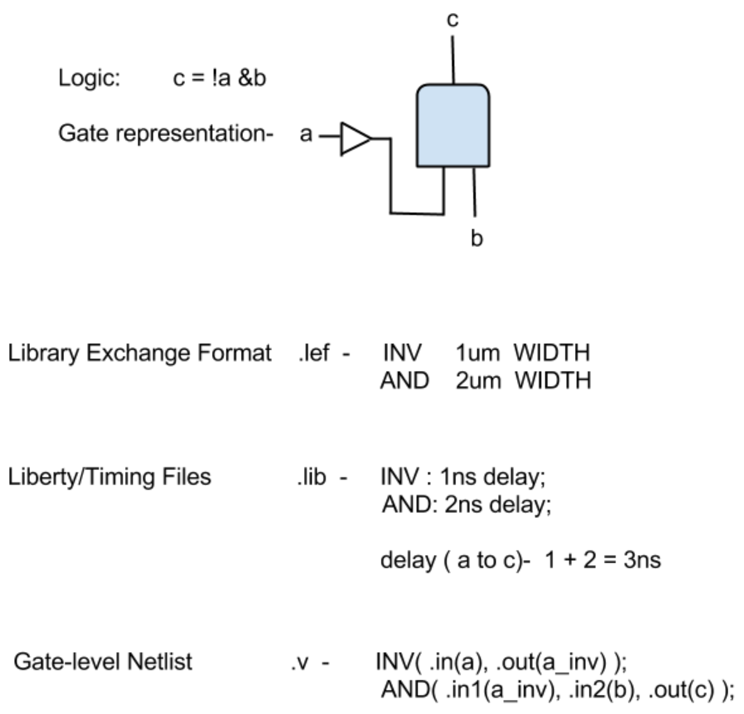
All these above steps are discussed in detail in Chapter 3.

Figure 1.2 shows a simple example of a combinational logic circuit giving an idea of sample .v, .lef and .lib files.

.v (gate-level netlist) - circuit in terms of gates and connections between them

.lib files - contains the timing information (delay of the gates)

.lef files - contains the geometry of the cells



```
Library Exchange Format   .lef -   INV    1um  WIDTH
                                   AND    2um  WIDTH

Liberty/Timing Files      .lib -   INV : 1ns delay;
                                   AND: 2ns delay;

                                   delay ( a to c)-  1 + 2 = 3ns

Gate-level Netlist        .v  -    INV( .in(a), .out(a_inv) );
                                   AND( .in1(a_inv), .in2(b), .out(c) );
```

Figure 1.2: Different sample file representations of a simple logic

Chapter 2

Front-end Design

2.1 Synchronous FIFO

A synchronous FIFO is a First-In-First-Out queue consisting of a storage array with control logic that manages the read and write of data and generates status flags. The number of rows of the array is called the DEPTH of the FIFO, and the bit length of each row is called the WIDTH of the FIFO. In this thesis design, Depth is 64-bit and and Width is 38-bit.

A synchronous FIFO has a single clock port for both data-read and data-write operations. These FIFO's are the ideal choice for high-performance systems due to high-operating speed. They also offer many other advantages that improve system performance and reduce complexity. These include status flags: synchronous flags, half-full, programmable almost-empty and almost-full flags. These FIFOs include features such as width expansion, depth expansion, and retransmit. Synchronous FIFOs are easier to use at high speeds because they use free-running clocks to time internal operations whereas asynchronous FIFOs require read and write pulses to be generated without an external clock reference[8].

2.2 FIFO: Front-end Design

After the design specifications and the behavioral description is written, the design is ready to be converted to RTL code.

Front-end design broadly includes following steps-

A. Logic Design

B. Logic Synthesis

C. Formal Verification

After synthesis, we perform gate-level simulation. In digital design, we do not go deep into transistor-level simulations unlike the analog design.

A. Logic Design

As the word says, this step deals with writing the RTL logic of the functionality we want in our chip. In digital design, a HDL (Hardware Description Language) code is written while analog designers make transistor-level schematics. Both are logic representations.

As previously explained, common hardware languages are - VHDL, Verilog and SystemVerilog. The written source code is simulated for correct functionality. This is called **logic simulation**.

For simulation, just as we draw testbenches for schematic design, we need a testbench to simulate the written HDL code. The testbench can be written in any language (software or hardware).

Logic Simulators available to simulate the written HDL and see waveforms are -

1. ModelSim by Mentor Graphics
2. Incisive Enterprise Simulator (SimVision) by Cadence Design Systems
3. Vivado Design IC Suite by Xilinx
4. Synopsys VCS

B. Logic Synthesis

After verifying the correct functionality of the design, it is ready to be synthesized if a synthesizable code has been written. The code has to be written very efficiently so that it does not have inferred latches while it is being synthesized.

The common EDA tools used for logic synthesis are -

1. Synopsys Design Compiler
2. Cadence RTL (RC) Compiler

The logic synthesis tool (RTL RC Compiler in this case) will take in the timing information (obtained from the characterization of the library technology) and synthesize our design with certain area, frequency and power. We can put our own constraints into synthesis step as per our requirement.

Input files to a logic synthesis tool -

1. .v (verified source verilog code) or .vhd (vhdl) or .sv (systemverilog) - This file is analogous to the schematic in analog design

2. .lib models (timing library information) - Slow, typical or fast (whatever mode we want the design to run). This step is analogous to choosing model libraries while performing simulation in ADE (Analog Design Environment)

3. Technology file - This file is very crucial as it specifies what technology (or process) the design has to build/synthesized with. We did not say for what technology we wrote the code. And this is exactly the beauty of digital design. If we want the same functionality for a different technology, we just need to change the path to new technology and the new technology timing library files. There is no need to write the HDL code again. On the contrary, schematics have to be redrawn for the new technology.

4. Constraint File (.sdc) - This is called the Synopsys Design Constraint file. Any constraint on clock, input delay, output delay, area, frequency etc can be included in this file. This file will be read by the logic synthesis tool and it will synthesize the design keeping in mind all the constraints specified in this file.

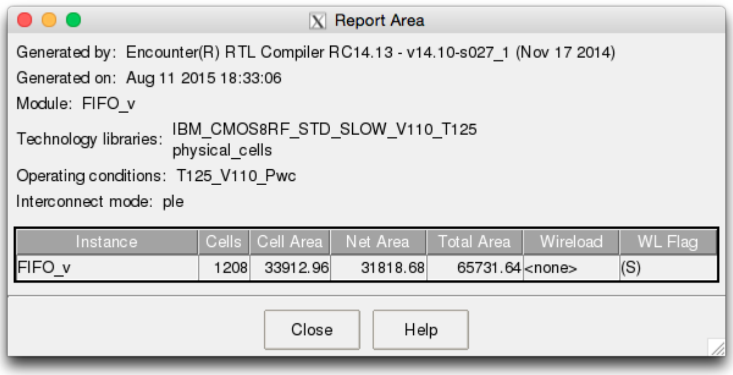
Output Files to a logic synthesis tool-

1. .v (Gate-level Netlist) - This file contains the synthesized circuit only in terms of gates such as AND, OR etc and the connection between them.

2. .sdc - The tool again generates a constraint file which is used in the back-end flow.

3. Power and area report - The tool reports the power and the approax area taken by the design (area of the cells only not of the wires as we have not routed the physical wires yet).

The results of logic synthesis steps have been shown in the figures below.



Report Area

Generated by: Encounter(R) RTL Compiler RC14.13 - v14.10-s027_1 (Nov 17 2014)
Generated on: Aug 11 2015 18:33:06
Module: FIFO_v
Technology libraries: IBM_CMOS8RF_STD_SLOW_V110_T125
physical_cells
Operating conditions: T125_V110_Pwc
Interconnect mode: ple

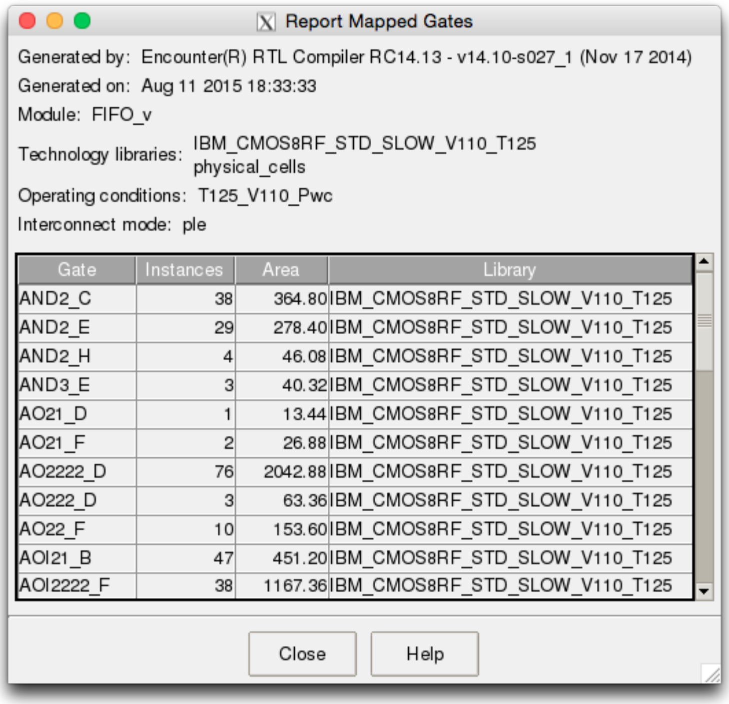
Instance	Cells	Cell Area	Net Area	Total Area	Wireload	WL Flag
FIFO_v	1208	33912.96	31818.68	65731.64	<none>	(S)

Close Help

FIGURE 2.1: Area after synthesis

The tool reports area of the design. The total cell area and the nets area approax 65000 sq um.

We obtain the report of the full list of the mapped gates in our design.



Report Mapped Gates

Generated by: Encounter(R) RTL Compiler RC14.13 - v14.10-s027_1 (Nov 17 2014)
Generated on: Aug 11 2015 18:33:33
Module: FIFO_v
Technology libraries: IBM_CMOS8RF_STD_SLOW_V110_T125
physical_cells
Operating conditions: T125_V110_Pwc
Interconnect mode: ple

Gate	Instances	Area	Library
AND2_C	38	364.80	IBM_CMOS8RF_STD_SLOW_V110_T125
AND2_E	29	278.40	IBM_CMOS8RF_STD_SLOW_V110_T125
AND2_H	4	46.08	IBM_CMOS8RF_STD_SLOW_V110_T125
AND3_E	3	40.32	IBM_CMOS8RF_STD_SLOW_V110_T125
AO21_D	1	13.44	IBM_CMOS8RF_STD_SLOW_V110_T125
AO21_F	2	26.88	IBM_CMOS8RF_STD_SLOW_V110_T125
AO2222_D	76	2042.88	IBM_CMOS8RF_STD_SLOW_V110_T125
AO222_D	3	63.36	IBM_CMOS8RF_STD_SLOW_V110_T125
AO22_F	10	153.60	IBM_CMOS8RF_STD_SLOW_V110_T125
AOI21_B	47	451.20	IBM_CMOS8RF_STD_SLOW_V110_T125
AOI2222_F	38	1167.36	IBM_CMOS8RF_STD_SLOW_V110_T125

Close Help

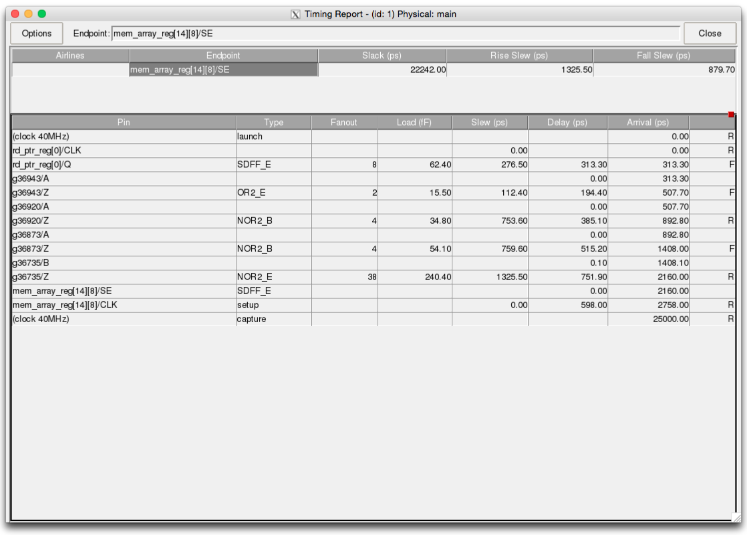
FIGURE 2.2: Mapped Gates

We have to make sure the **Slack** is not negative in Static Timing Analysis (STA). Slack is defined as the difference of the required time and the arrival time of the timing path.

At the frequency we provided in the constraint file, if the slack is positive, it is good as it means the signal was able to reach/arrive on time traversing the combinational path on time. The arrival time was less than the required (as specified in the slow/fast timing files).

If the slack comes out to be negative, we have to increase the time period (or reduce frequency) to get the timing meet in STA.

After the logic synthesis, the gate-level netlist obtained is again simulated for correct functionality of design. This is called **post-synthesis simulation**.



Timing Report - (id: 1) Physical: main

Endpoint: mem_array_reg[14][8]/SE

Airlines	Endpoint	Slack (ps)	Rise Slew (ps)	Fall Slew (ps)
	mem_array_reg[14][8]/SE	22242.00	1325.50	879.70

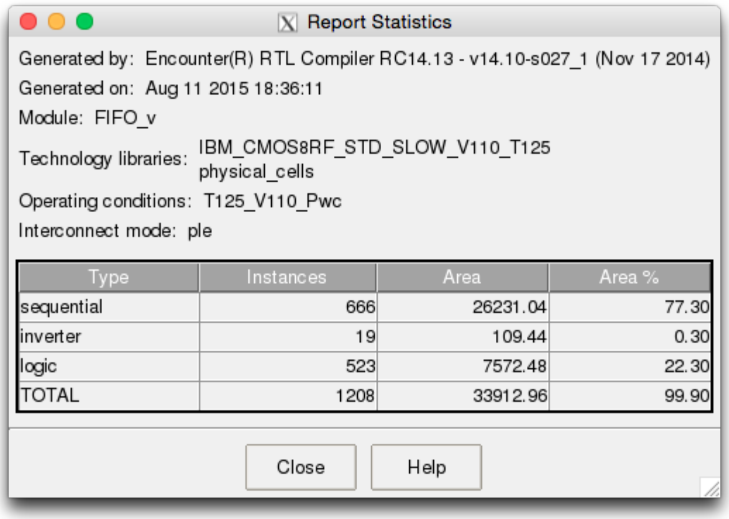
Pin	Type	Fanout	Load (fF)	Slew (ps)	Delay (ps)	Arrival (ps)	
(clock 40MHz)	launch					0.00	R
rd_ptr_reg[0]/CLK				0.00		0.00	R
rd_ptr_reg[0]/Q	SDFF_E	8	62.40	276.50	313.30	313.30	F
g36943/A					0.00	313.30	
g36943/Z	OR2_E	2	15.50	112.40	194.40	507.70	F
g36920/A					0.00	507.70	
g36920/Z	NOR2_B	4	34.80	753.60	385.10	892.80	R
g36873/A					0.00	892.80	
g36873/Z	NOR2_B	4	54.10	759.60	515.20	1408.00	F
g36735/B					0.10	1408.10	
g36735/Z	NOR2_E	38	240.40	1325.50	751.90	2160.00	R
mem_array_reg[14][8]/SE	SDFF_E				0.00	2160.00	
mem_array_reg[14][8]/CLK	setup			0.00	598.00	2758.00	R
(clock 40MHz)	capture					25000.00	R

FIGURE 2.3: Timing Report

And we see that the Slack is positive.

We also see the statistics giving clear picture of number and area of logic and sequential cells given by the statistics report.

C. Formal Verification



Report Statistics

Generated by: Encounter(R) RTL Compiler RC14.13 - v14.10-s027_1 (Nov 17 2014)
Generated on: Aug 11 2015 18:36:11
Module: FIFO_v
Technology libraries: IBM_CMOS8RF_STD_SLOW_V110_T125
physical_cells
Operating conditions: T125_V110_Pwc
Interconnect mode: ple

Type	Instances	Area	Area %
sequential	666	26231.04	77.30
inverter	19	109.44	0.30
logic	523	7572.48	22.30
TOTAL	1208	33912.96	99.90

FIGURE 2.4: Statistics Report

A verification step is performed called Formal verification. The step is for logical equivalence checking through a tool. Gate-level netlist obtained after synthesis is checked against the RTL code.

This gate-level netlist is again simulated to verify functionality. This is called **post-synthesis gate-level simulation**.

This completes the Front-end part of the design.

Chapter 3

Back-end Design

3.1 FIFO: Back-End Physical Design

Tool- Cadence Encounter EDI141

Broadly, the following steps were performed-

Netlist Import

Floorplanning

Power Routing

Placement of standard cells

Clock-Tree synthesis (CTS)

Routing of standard cells

Timing verification

Physical DRC/LVS verification

Sign-off/GDSII Export

3.1.1 A short Introduction about the tool

Cadence Encounter EDI is a complete tool for back-end design by Cadence Design Systems,Inc (CDS). It is an automated tool for placing & routing, clock-tree synthesis (CTS) and timing verification.

Input Files to Cadence Encounter -

1. Gate-level netlist(.v)
2. Library Exchange Format(.lef)
3. Timing Files/Liberty Files (.lib)
4. Synopsys Design Constraint (.sdc)
5. I/O pad file (.io)
6. Power and Ground Nets (VDD/VSS)

Output Files out of Cadence Encounter-

1. strmOutMap (GDSII) file - This file has design in polygons and is understood by the foundry.
2. Design Exchange Format (DEF) File - The file is to import the design to virtuoso for a mixed-signal design or DRC/LVS physical verification.

The design in Cadence Encounter can be seen in three views-

1. Floorplan view
2. Amoeba view

3. Placement view

Figure 3.1 below shows a basic Encounter EDI flow.

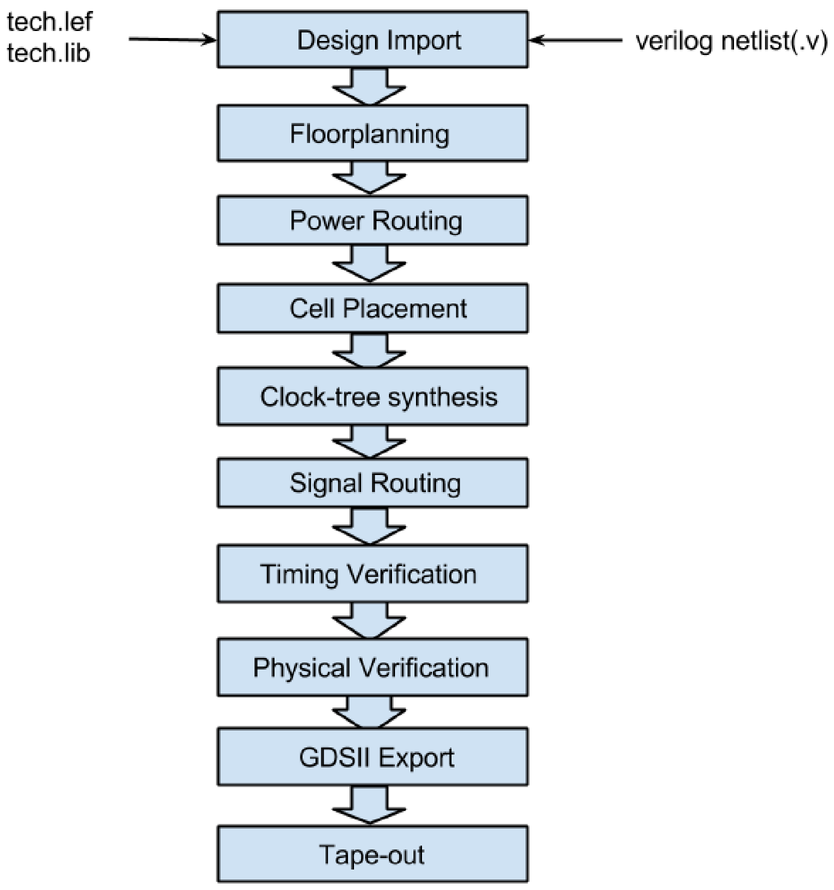


Figure 3.1: Basic Encounter EDI Flow

3.1.2 Importing Files

1. Gate-level netlist of the design (.v file) - The gate-level netlist obtained from logic synthesis tools (RC compiler) containing the circuit in terms of logic-gates and connections between them.

2. Library Exchange Format (.lef file) - These are the files which contains the technology information of a particular process technology. These files include the design rules for routing and abstract of the cells. These files do not contain any information about the internal netlist of the design.

Abstract- An abstract is a high-level representation of a layout view. An abstract contains information about the type and size of the cells, position of pins or terminals and the overall sizes of blockages. Standard cells are often saved in abstract format.
The abstracts are used in place of full layouts to improve the performance of place and route tools. After place and route is complete, the abstracts are replaced back with the layouts.
The information contained in the LEF file is the text version of the Virtuoso cell view abstract and includes layer names, layer widths and layer usage. LEF file can be obtained using abstract generator tool in Virtuoso.

LEF file has two different categories -

A. Technology/Tech LEF - contains physical information about routing layers, design rules, metal vias definitions, width, pitch, sheet resistance and sheet metal capacitance of the process technology
B. MACROS/Cell LEF - cells descriptions, cells dimensions, layout of pins and blockages, routing information of the each circuit block in the design library.

Note - Always Tech LEF is imported first and then the Macro LEF.

3. Timing Library Files/ Liberty Files (.lib file) - These files are very important and specify the timing characteristics of each standard cell drawn in the technology. Each cell is separately wire-modelled against different load capacitances and PVT (Process, voltage and temperature) conditions. All these calculated experimental values are stored in timing files (.lib files). The bc(best case) and wc (worst case) conditions during design are tested by these max and min timing libraries. Liberate Altos is a tool for Library Characterization.

4. Pad file- This is the IO file which places the pad all around the die. This file is not included here as this digitally implemented block would become a part of a bigger layout design.

5. Power Information - This relates to the power nets to use in the layout. Power net names are those defined in the LEF technology file.

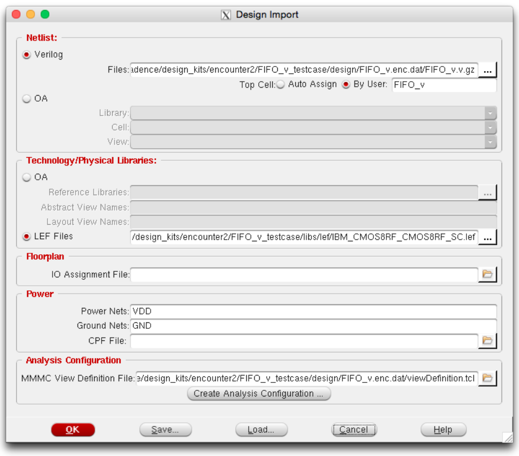
File - Import Design, this inputs all the files to the Encounter. The first one is the gate-level netlist obtained from RTL Compiler which is imported into Encounter EDI tool. Encounter only supports verilog-netlist(.v) .

Also, our MMMC (Multi-Mode Multi Corner) configuration is setup by importing all the relevant subsidiary files.

6. Captable File - The capacitance value at different process parameters for 130nm technology.

7. Extraction File (.qrc) - he file is used to extract parasitic capacitances and resistances.

8. Signal Integrity (.cdb) -CDB (CeltIC dB) noise files are to prevent crosstalk.



Netlist:

[✓ Verilog]

Files: adence/design_kits/encounter2/FIFO_v_testcase/design/FIFO_v.enc.dat/FIFO_v.v.gz

Top Cell: Auto Assign [✓ By User]: FIFO_v

OA

Library:
Cell:
View:

Technology/Physical Libraries:

OA

Reference Libraries:
Abstract View Names:
Layout View Names:

[✓ LEF Files] /design_kits/encounter2/FIFO_v_testcase/libs/lef/IBM_CMOS8RF_CMOS8RF_SC.lef

Floorplan

IO Assignment File:

Power

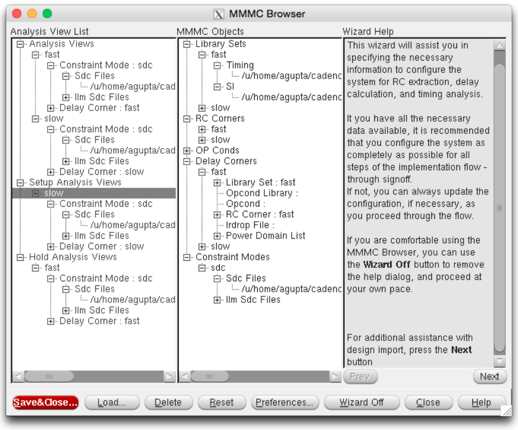
Power Nets: VDD
Ground Nets: GND
CPF File:

Analysis Configuration

MMMC View Definition File: e/design_kits/encounter2/FIFO_v_testcase/design/FIFO_v.enc.dat/viewDefinition.tcl

FIGURE 3.2: Importing files

9. Design Constraint File (.sdc) - A constraint file outputted by RTL RC Compiler.



```
Analysis Views
  fast
    Constraint Mode : sdc
      Sdc Files
        /u/home/agupta/cad
      Ilm Sdc Files
    Delay Corner : fast
  slow
    Constraint Mode : sdc
      Sdc Files
        /u/home/agupta/cad
      Ilm Sdc Files
    Delay Corner : slow
Setup Analysis Views
  slow
    Constraint Mode : sdc
      Sdc Files
        /u/home/agupta/cad
      Ilm Sdc Files
    Delay Corner : slow
Hold Analysis Views
  fast
    Constraint Mode : sdc
      Sdc Files
        /u/home/agupta/cad
      Ilm Sdc Files
    Delay Corner : fast
```

```
Library Sets
  fast
    Timing
      /u/home/agupta/cadenc
    SI
      /u/home/agupta/cadenc
  slow
RC Corners
  fast
  slow
OP Conds
Delay Corners
  fast
    Library Set : fast
    Opcond Library :
    Opcond :
    RC Corner : fast
    Irdrop File :
    Power Domain List
  slow
Constraint Modes
  sdc
    Sdc Files
      /u/home/agupta/cadenc
    Ilm Sdc Files
```

This wizard will assist you in specifying the necessary information to configure the system for RC extraction, delay calculation, and timing analysis.

It you have all the necessary data available, it is recommended that you configure the system as completely as possible for all steps of the implementation flow - through signoff.
If not, you can always update the configuration, if necessary, as you proceed through the flow.

If you are comfortable using the MMMC Browser, you can use the **Wizard Off** button to remove the help dialog, and proceed at your own pace.

For additional assistance with design import, press the **Next** button

FIGURE 3.3: Creating Configuration Files

3.1.3 Specifying Floorplan

After the configuration files are created, the original loaded floorplan looks like in the figure. The die area has been appended all around the core.

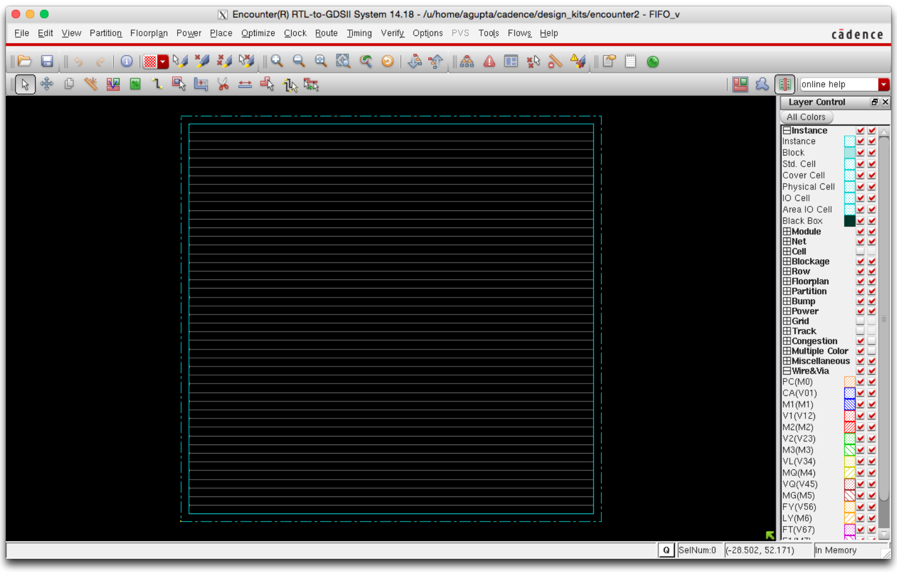

Encounter(R) RTL-to-GDSII System 14.18 - /u/home/agupta/cadence/design_kits/encounter2 - FIFO_v
File Edit View Partition Floorplan Power Place Optimize Clock Route Timing Verify Options PVS Tools Flows Help
cādence
online help
Layer Control
All Colors
Instance
Module
Net
Cell
Blockage
Row
Floorplan
Partition
Bump
Power
Grid
Track
Congestion
Multiple Color
Miscellaneous
Wire&Via
SelNum:0
(-28.502, 52.171)
In Memory

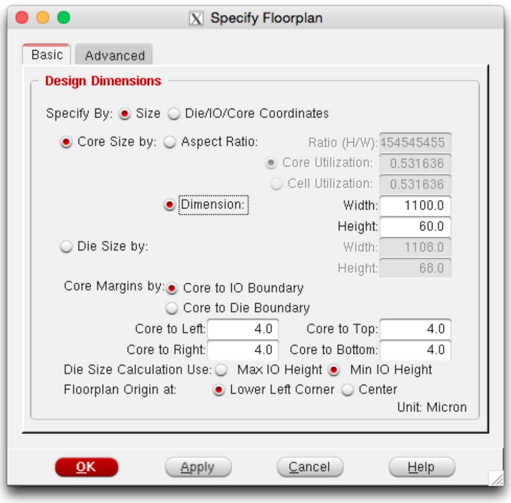
FIGURE 3.4: Initial Floorplan

Specify customized Floorplan

We specify the Aspect Ratio = 1 , Core utilization = 70. Die Size (Width, Height) and Core Size (Width, Height) as per the area value of the design given out by the logic synthesis tool RTL Compiler. We can also change the die area around the core area of the chip.

Size the Floorplan - 60 micron X 1090 micron, this dimension is a requirement for the project.

The initial floorplan was sized-customized as per the area report given by the RTL compiler in the logic synthesis step. The Encounter EDI tool automatically size the floorplan seeing the area but we could apply our own constraints.



Specify Floorplan

Basic | Advanced

Design Dimensions

Specify By: [✓ Size] Die/IO/Core Coordinates

[✓ Core Size by:] Aspect Ratio: Ratio (H/W): 454545455

Core Utilization: 0.531636

Cell Utilization: 0.531636

[✓ Dimension:] Width: 1100.0

Height: 60.0

Die Size by: Width: 1108.0

Height: 68.0

Core Margins by: [✓ Core to IO Boundary] Core to Die Boundary

Core to Left: 4.0 Core to Top: 4.0

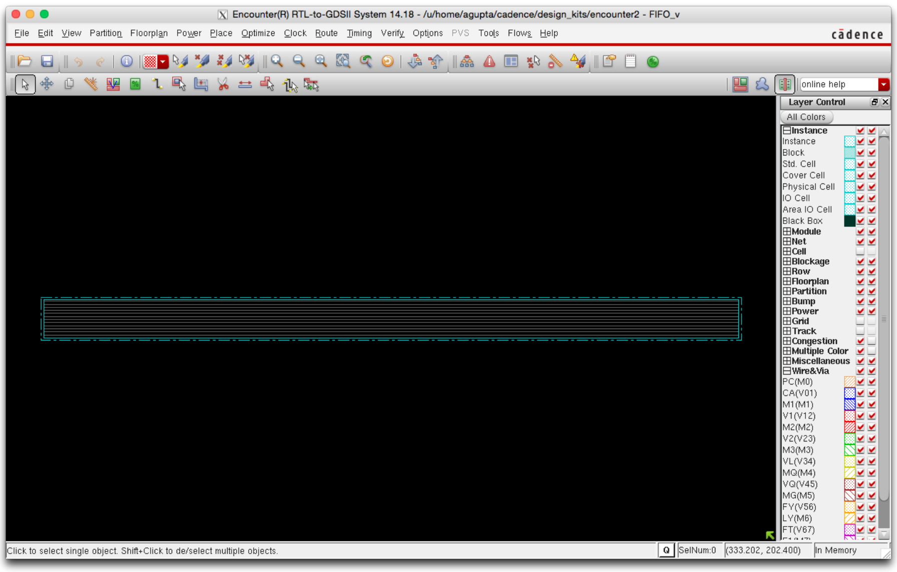
Core to Right: 4.0 Core to Bottom: 4.0

Die Size Calculation Use: Max IO Height [✓ Min IO Height]

Floorplan Origin at: [✓ Lower Left Corner] Center

Unit: Micron

FIGURE 3.5: Loading Floorplan



Encounter(R) RTL-to-GDSII System 14.18 - /u/home/agupta/cadence/design_kits/encounter2 - FIFO_v

Click to select single object. Shift+Click to de/select multiple objects. SelNum:0 (333.202, 202.400) In Memory

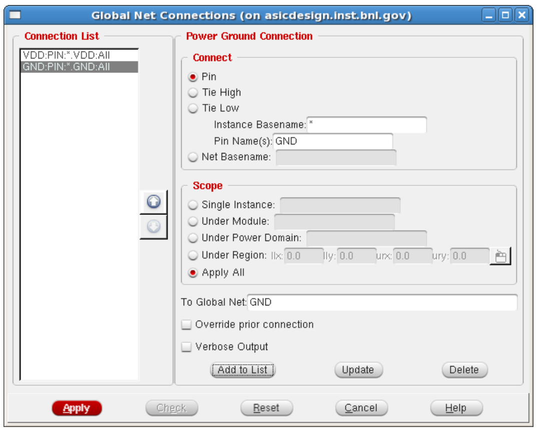
FIGURE 3.6: Sized (customized) Floorplan

3.1.4 Power Rails

Its time to specify the power rails. We have to add the Global Net Connection. We will add two rails. VDD and GND.

Adding Power Rings all around the design.

By browsing **Power-Add Power Rings**, we can add VDD and GND all around the Floorplan Core. Make sure you route low metal layers for power rails. The



Connection List
VDD:PIN:*.VDD:All
GND:PIN:*.GND:All

Power Ground Connection

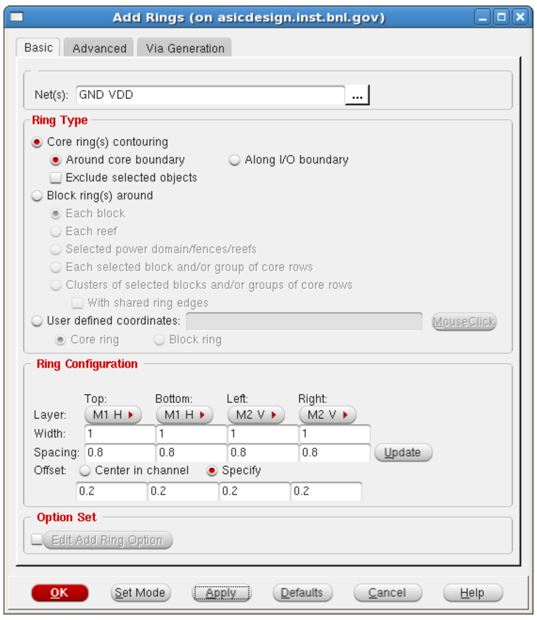
Connect
[✓ Pin]
Tie High
Tie Low
Instance Basename: *
Pin Name(s): GND
Net Basename:

Scope
Single Instance:
Under Module:
Under Power Domain:
Under Region: llx: 0.0 lly: 0.0 urx: 0.0 ury: 0.0
[✓ Apply All]

To Global Net: GND
Override prior connection
Verbose Output

FIGURE 3.7: Adding Power Rails

higher metal layers are wider and thus have more resistance resulting in greater IR drop.



Net(s): GND VDD

Ring Type
[✓ Core ring(s) contouring]
[✓ Around core boundary] Along I/O boundary
Exclude selected objects
Block ring(s) around
[✓ Each block]
Each reef
Selected power domain/fences/reefs
Each selected block and/or group of core rows
Clusters of selected blocks and/or groups of core rows
With shared ring edges
User defined coordinates:
[✓ Core ring] Block ring

Ring Configuration

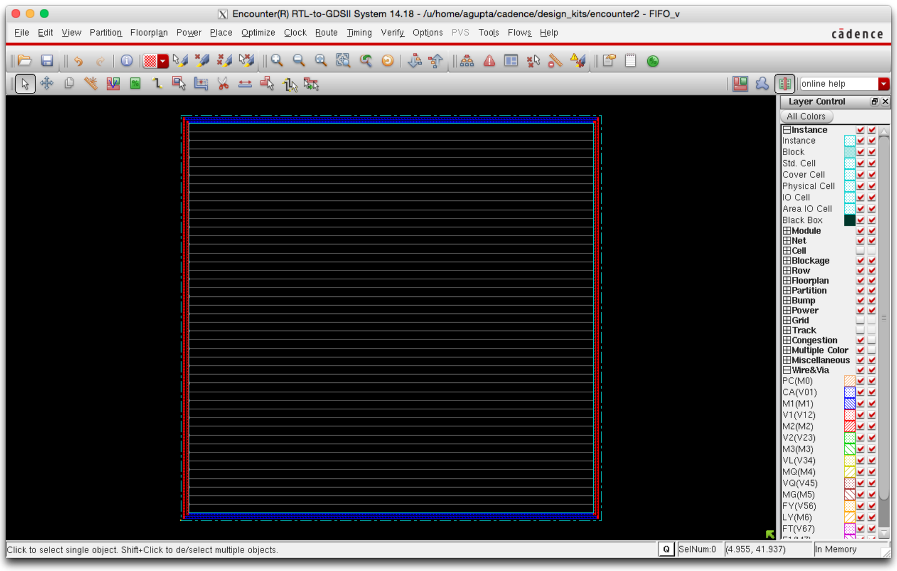
	Top:	Bottom:	Left:	Right:
Layer:	M1 H	M1 H	M2 V	M2 V
Width:	1	1	1	1
Spacing:	0.8	0.8	0.8	0.8
Offset: Center in channel [✓ Specify]	0.2	0.2	0.2	0.2

Option Set

FIGURE 3.8: Power Rings

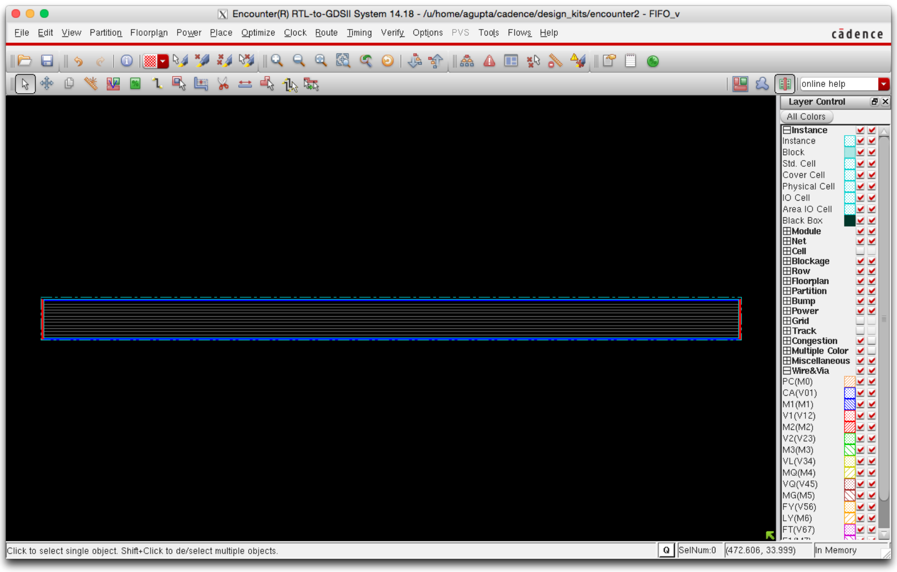
We see power rails on all 4 sides of our floorplan.

We see power rails on all 4 sides of our customized floorplan in Figure 3.10



Click to select single object. Shift+Click to de/select multiple objects.

FIGURE 3.9: Power Rings on design

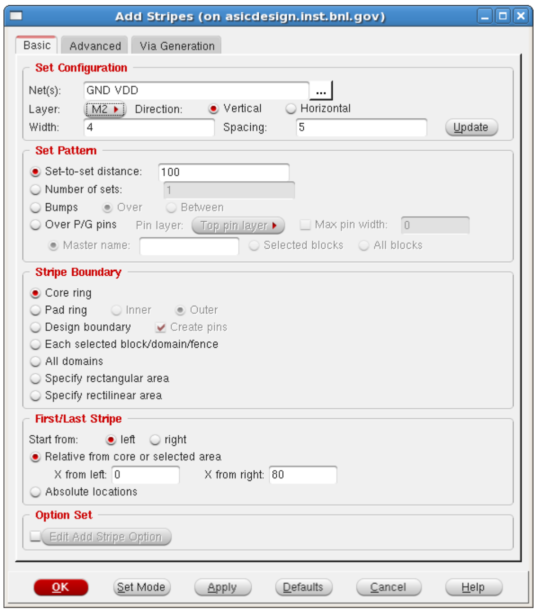


Click to select single object. Shift+Click to de/select multiple objects.

FIGURE 3.10: Power Rings- customized floorplan

3.1.5 Power Stripes

Adding Power Stripes By browsing to **Power -Add Power Stripes** we could run VDD and GND lines through the floorplan of the core.


Add Stripes (on asicdesign.inst.bnl.gov)

Basic | Advanced | Via Generation

Set Configuration

Net(s): GND VDD

Layer: M2 Direction: [✓ Vertical] Horizontal

Width: 4 Spacing: 5 Update

Set Pattern

[✓ Set-to-set distance:] 100

Number of sets: 1

Bumps [✓ Over] Between

Over P/G pins Pin layer: Top pin layer Max pin width: 0

[✓ Master name:] Selected blocks All blocks

Stripe Boundary

[✓ Core ring]

Pad ring Inner [✓ Outer]

Design boundary [✓ Create pins]

Each selected block/domain/fence

All domains

Specify rectangular area

Specify rectilinear area

First/Last Stripe

Start from: [✓ left] right

[✓ Relative from core or selected area]

X from left: 0 X from right: 80

Absolute locations

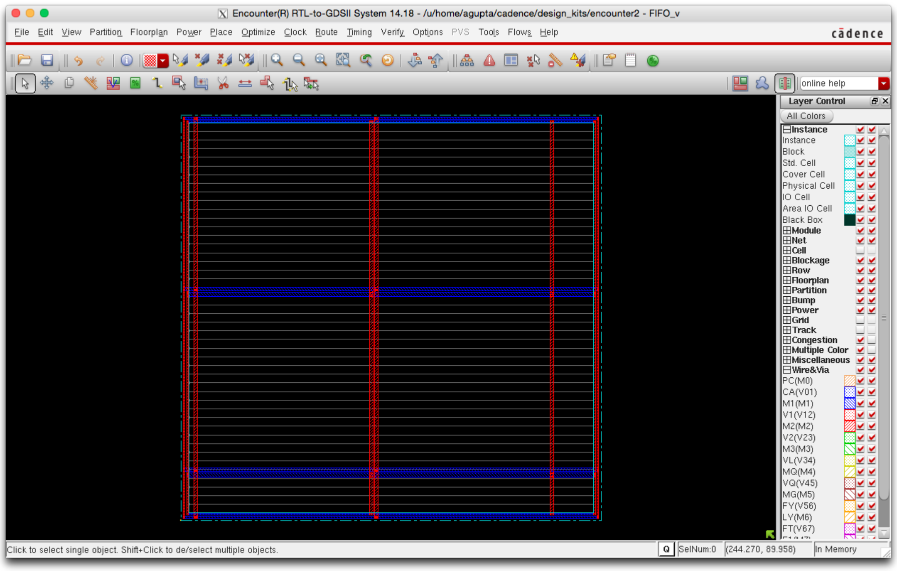
Option Set

Edit Add Stripe Option

OK | Set Mode | Apply | Defaults | Cancel | Help

FIGURE 3.11: Power Stripes

We see a M2 (Vertical) and M1(Horizontal) power stripes in the Floorplan design.



Click to select single object. Shift+Click to de/select multiple objects. | SelNum:0 | (244.270, 89.958) | In Memory

FIGURE 3.12: Power Stripes on design

The thickness and the placement of these power rings and power stripes are in designers' hand and can be controlled through the tool.

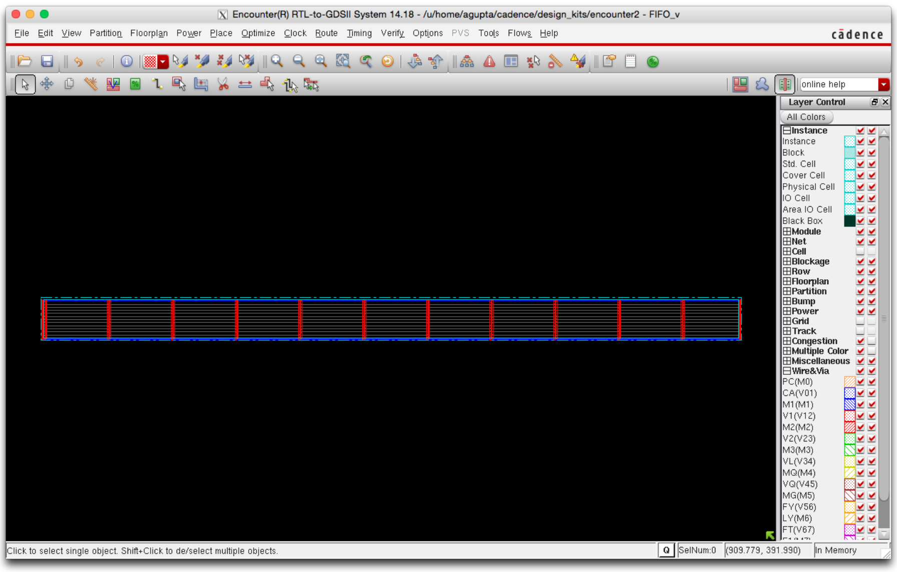
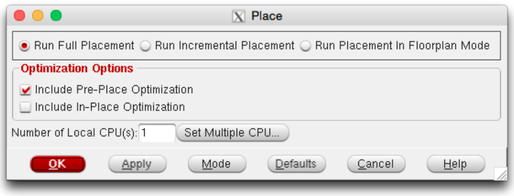


FIGURE 3.13: Power Stripes- customized floorplan

3.1.6 Placing Cells

The Encounter reads in the Tech LEF and the MACRO LEF file and places the standard cells in the rows of the floorplan as per the rules information contained in the LEF. All cells have the same height and but are different in width.


Place

[● Run Full Placement] ○ Run Incremental Placement ○ Run Placement In Floorplan Mode

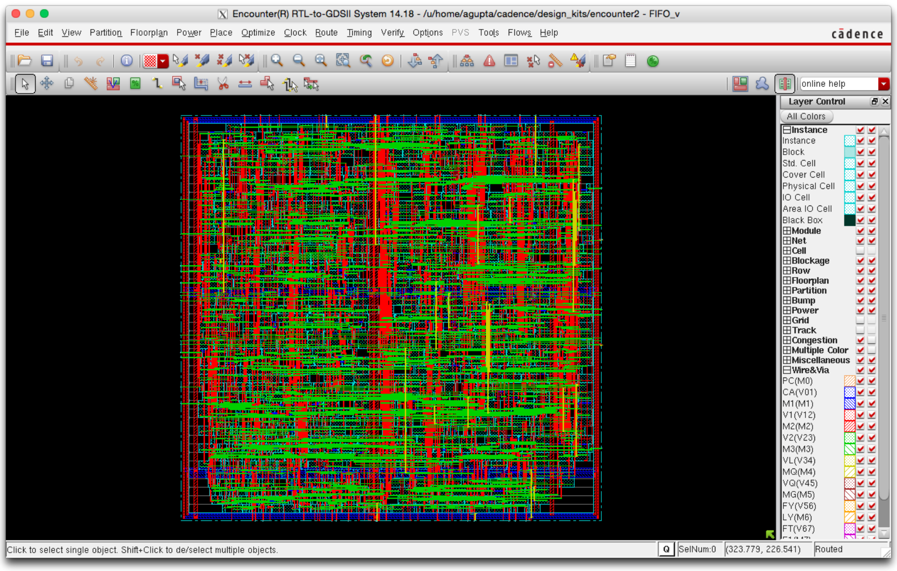
Optimization Options

[✓] Include Pre-Place Optimization
[ ] Include In-Place Optimization

Number of Local CPU(s): 1

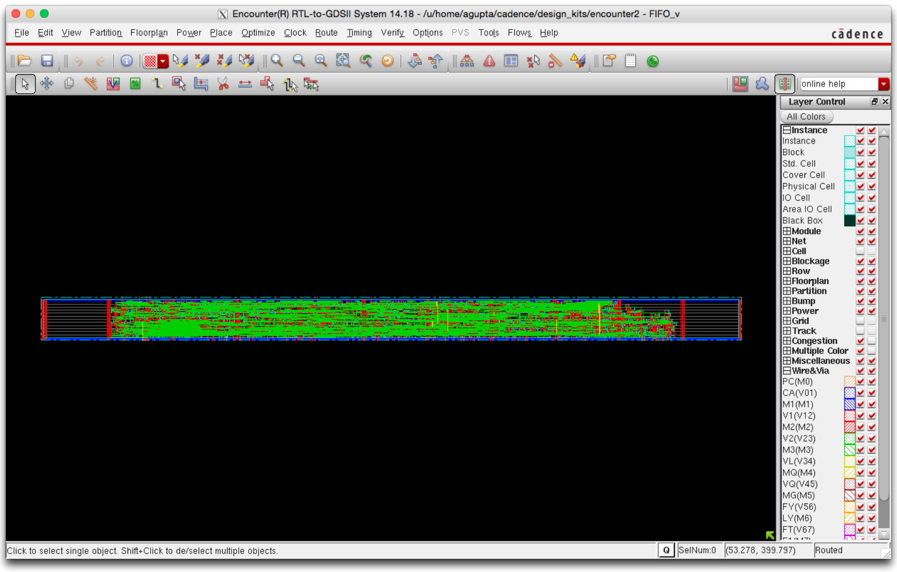
FIGURE 3.14: Placing Standard Cells

Placement of standard cells in the rows of the floorplan shown in the Figure 3.15


Encounter(R) RTL-to-GDSII System 14.18 - /u/home/agupta/cadence/design_kits/encounter2 - FIFO_v

Click to select single object. Shift+Click to de/select multiple objects. SelNum:0 (323.779, 226.541) Routed

Figure 3.15: Placement of Standard Cells


Encounter(R) RTL-to-GDSII System 14.18 - /u/home/agupta/cadence/design_kits/encounter2 - FIFO_v

Click to select single object. Shift+Click to de/select multiple objects. SelNum:0 (53.278, 399.797) Routed

Figure 3.16: Placement of Standard Cells- customized floorplan

Turn off the visibility of the nets in the design and just have a look at the placed standard cells (Figure 3.17).

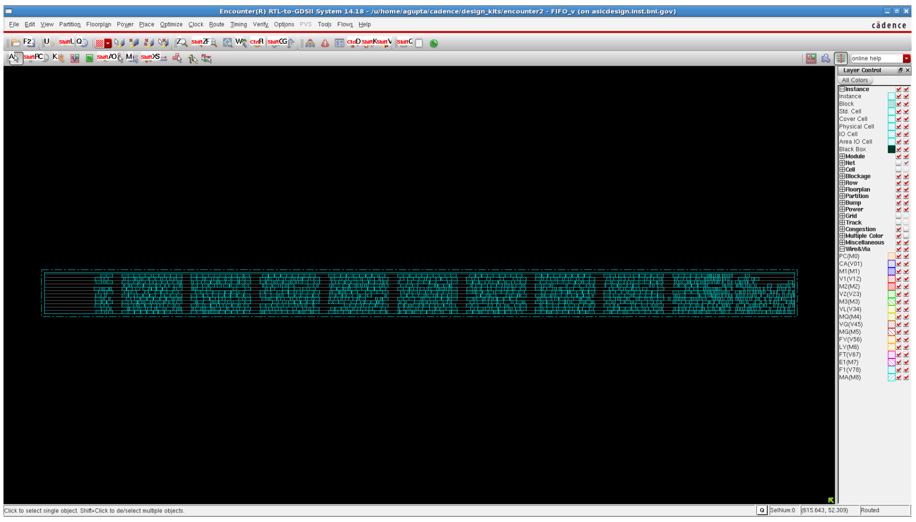


FIGURE 3.17: Placed cells without Nets

Now we will tell the tool what process technology we are using.

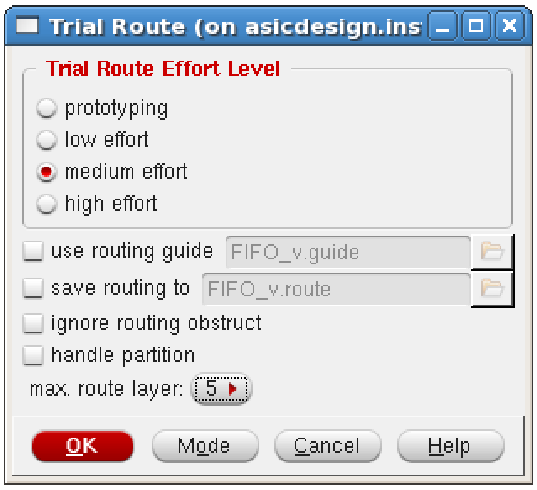
setDesignMode -process 130

The above command is to specify the process technology because it sets capacitive filters and extraction effort level based on the process node. This is done through command line.

3.1.7 Trial Route

Trial Route is run to check the congestion. We can specify how many maximum number of routing metal layers we want to use in our design. After running trial route, we look for congestion in the design through congestion markers. There might be numbers in the form of **H: -top/bottom** or **V: -top/bottom**. H stands for horizontal congestion and V stands for vertical congestion. The - top is for the required number of routing tracks used in this area and - bottom is the available routing tracks.

Note- This is only a trial route for Global Assignments. The signals are routed by Nano Route engine later.



Trial Route (on asicdesign.ins

Trial Route Effort Level

- ( ) prototyping
- ( ) low effort
- (●) medium effort
- ( ) high effort

- [ ] use routing guide FIFO_v.guide
- [ ] save routing to FIFO_v.route
- [ ] ignore routing obstruct
- [ ] handle partition

max. route layer: 5

FIGURE 3.18: Trial Route Run

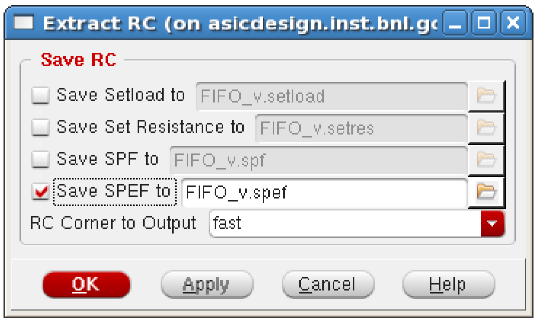
3.1.8 RC Extraction

The capacitance and resistance values for all the nets in the design are extracted by the extract RC form.

Extraction is run in pre-route mode prior to signal routing and in post-route mode after the signals are routed with the NanoRoute license of the tool.

The RC extraction mode can be changed by the **Options-Set Mode- Specify Extraction Mode**

Select **Timing-Extract RC**



Save RC

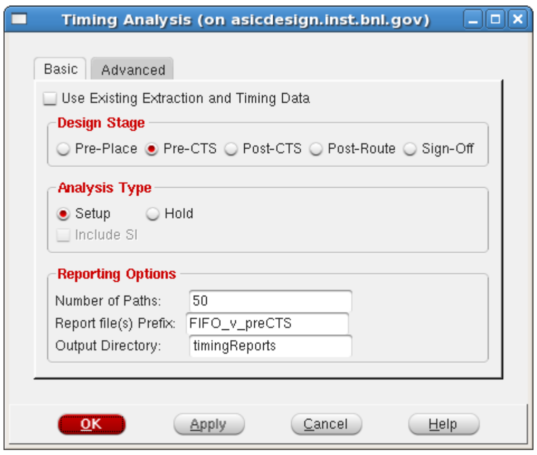
- [ ] Save Setload to FIFO_v.setload
- [ ] Save Set Resistance to FIFO_v.setres
- [ ] Save SPF to FIFO_v.spf
- [✓] Save SPEF to FIFO_v.spef

RC Corner to Output: fast

FIGURE 3.19: Extracting RC

3.1.9 Timing Analysis

One thing to ponder here, how are we running Timing Analysis without having a clock in the design. The tool is smart enough to assume an ideal clock with transition delay value of 0.1 ps (pico second).



- [ ] Use Existing Extraction and Timing Data

Design Stage: Pre-Place [✓ Pre-CTS] Post-CTS Post-Route Sign-Off

Analysis Type: [✓ Setup] Hold

- [ ] Include SI

Reporting Options

Number of Paths:	50
Report file(s) Prefix:	FIFO_v_preCTS
Output Directory:	timingReports

FIGURE 3.20: Timing Analysis

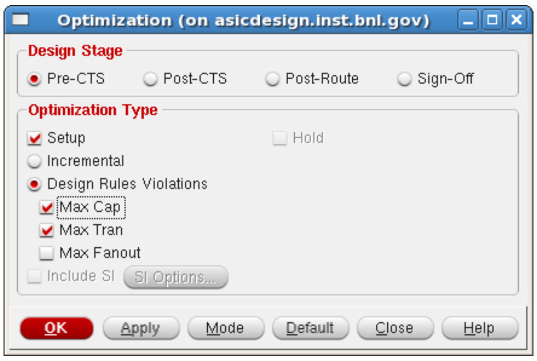
3.1.10 Pre-CTS Optimization

Before we do clock tree synthesis (CTS), it is better to optimize the design to fix all the timing violations. There may be violations in the design such as -

1. Setup and Hold Violations
2. DRV (Design Rule Violations - max transitions and max capacitance violations)

.

Therefore, timing optimization is run several times during the implementation flow to fix the above issues.


Optimization (on asicdesign.inst.bnl.gov)

Design Stage

[✓ Pre-CTS] Post-CTS Post-Route Sign-Off

Optimization Type

[✓ Setup] Hold

Incremental

[✓ Design Rules Violations]

[✓ Max Cap]

[✓ Max Tran]

Max Fanout

Include SI

FIGURE 3.21: Timing Optimization

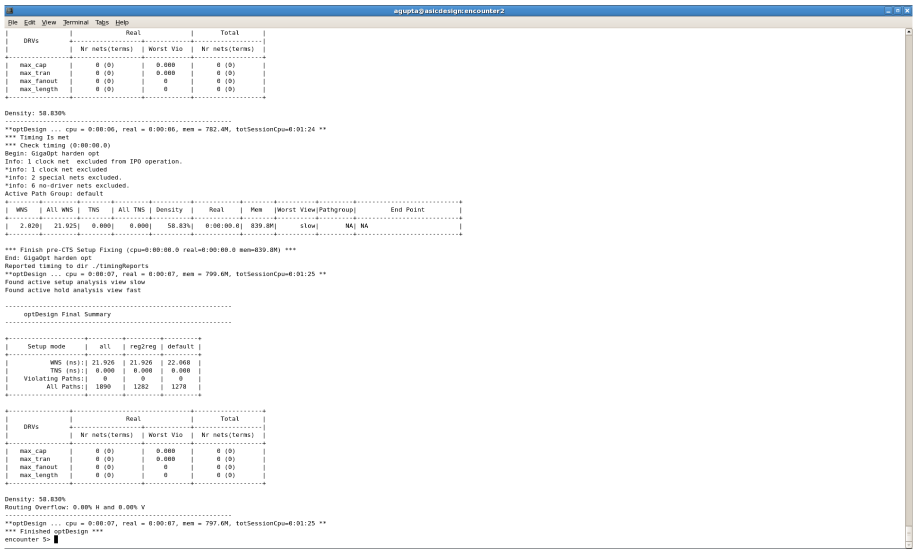
After running this, all the violating paths would go away as the tool runs optimization many times till we the design is free from any kind of violation.

Here, its a screenshot from the command terminal window of the design-

3.1.11 Clock Tree Synthesis (CTS)

Its time to route the clock in the design. For this, we have to generate clock specification file(.ctstch). There are two ways to generate a clock specification file -

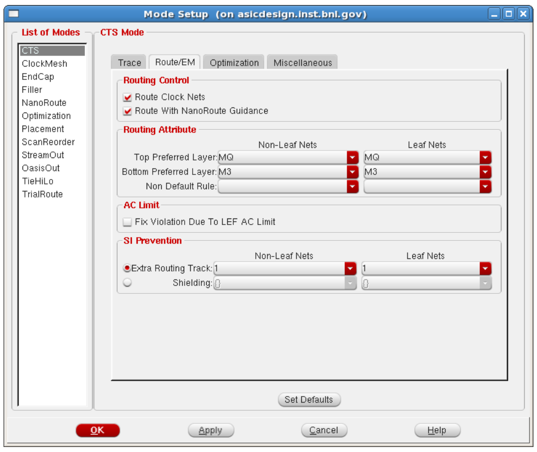
1. Hand-written clock specification file
2. Tool-generated clock specification file


agupta@asicdesign:encounter2
File Edit View Terminal Tabs Help

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

Density: 58.830%
------------------------------------------------------------
**optDesign ... cpu = 0:00:06, real = 0:00:06, mem = 782.4M, totSessionCpu=0:01:24 **
*** Timing Is met
*** Check timing (0:00:00.0)
Begin: GigaOpt harden opt
Info: 1 clock net  excluded from IPO operation.
*info: 1 clock net excluded
*info: 2 special nets excluded.
*info: 6 no-driver nets excluded.
Active Path Group: default
+--------+---------+--------+---------+----------+------------+--------+----------+---------+---------------------------+
|  WNS   | All WNS |  TNS   | All TNS | Density  |    Real    |  Mem   |Worst View|Pathgroup|         End Point         |
+--------+---------+--------+---------+----------+------------+--------+----------+---------+---------------------------+
|   2.020|   21.925|   0.000|    0.000|    58.83%|  0:00:00.0|  839.8M|      slow|       NA| NA                        |
+--------+---------+--------+---------+----------+------------+--------+----------+---------+---------------------------+

*** Finish pre-CTS Setup Fixing (cpu=0:00:00.0 real=0:00:00.0 mem=839.8M) ***
End: GigaOpt harden opt
Reported timing to dir ./timingReports
**optDesign ... cpu = 0:00:07, real = 0:00:07, mem = 799.6M, totSessionCpu=0:01:25 **
Found active setup analysis view slow
Found active hold analysis view fast

------------------------------------------------------------
     optDesign Final Summary
------------------------------------------------------------

+--------------------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | default |
+--------------------+---------+---------+---------+
|           WNS (ns):| 21.926  | 21.926  | 22.068  |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |    0    |
|          All Paths:|  1890   |  1282   |  1278   |
+--------------------+---------+---------+---------+

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

Density: 58.830%
Routing Overflow: 0.00% H and 0.00% V
------------------------------------------------------------
**optDesign ... cpu = 0:00:07, real = 0:00:07, mem = 797.6M, totSessionCpu=0:01:25 **
*** Finished optDesign ***
encounter 5> 
```

Figure 3.22: Clearing Violating Paths

We could browse to **Clock- Synthesize Clock Tree - Mode- Mode Setup - Route EM tab**, here we can specify on which metal layers we want the clock signals to get routed. Select Metal layers for Top and Bottom Preferred routing layers for Non-Leaf Nets and Leaf Nets.



Routing Control
[✓] Route Clock Nets
[✓] Route With NanoRoute Guidance

Routing Attribute

	Non-Leaf Nets	Leaf Nets
Top Preferred Layer:	MQ	MQ
Bottom Preferred Layer:	M3	M3
Non Default Rule:		

AC Limit
[ ] Fix Violation Due To LEF AC Limit

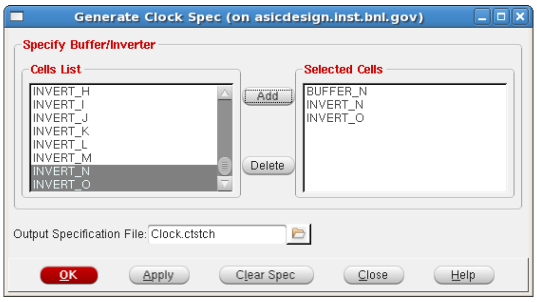
SI Prevention

	Non-Leaf Nets	Leaf Nets
[●] Extra Routing Track:	1	1
[○] Shielding:	0	0

Figure 3.23: Selecting Metal Layers to route Clock Signal

It is advisable to let EDI generate the Clock Specification(.ctstch) file. Generation of this file depends on the constraint file (.sdc) file imported while creating the MMMC configuration file in Step1.

SDC file (Constraint file) contains the information about the clock ports, time period, pulse width, duty cycle , rise time , fall time of the clock signal. We manually put those constraints to the design while generating the .sdc (constraint file) from the logic synthesis tool.



Cells List

```
INVERT_H
INVERT_I
INVERT_J
INVERT_K
INVERT_L
INVERT_M
INVERT_N
INVERT_O
```

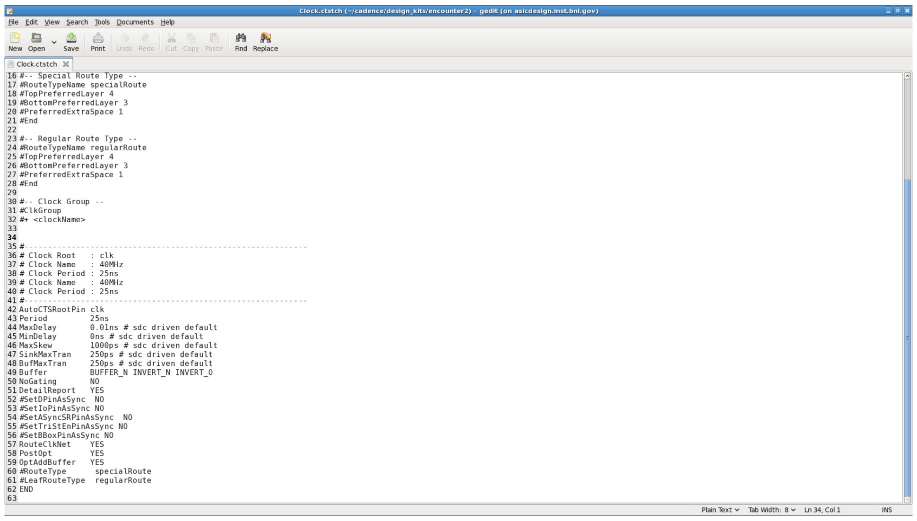
Selected Cells

```
BUFFER_N
INVERT_N
INVERT_O
```

Output Specification File: Clock.ctstch

FIGURE 3.24: Generating Clock Specification File

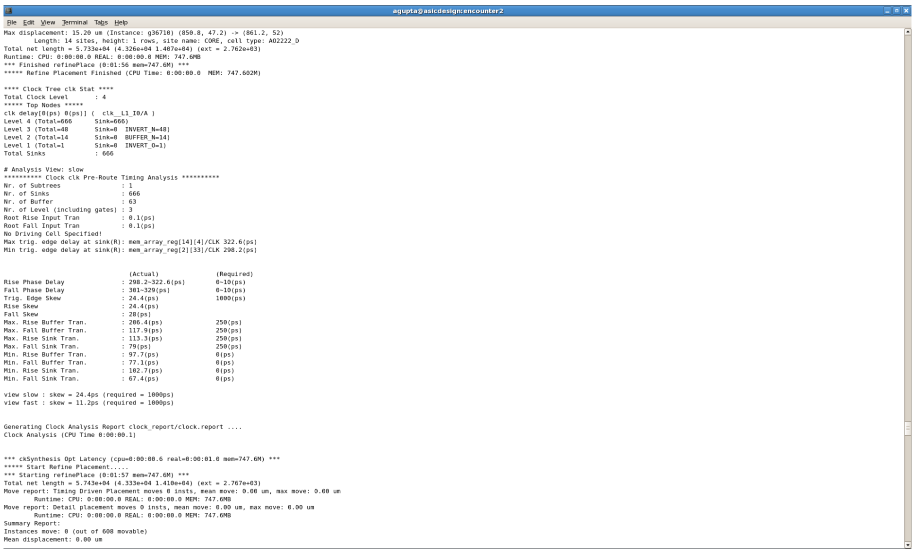
After running the above command, we browse through our design directory and have a look at the generated Clock Specification (.ctstch) file. This is how it looks like-


Clock.ctstch (~/cadence/design_kits/encounter2) - gedit (on asicdesign.inst.bnl.gov)
File Edit View Search Tools Documents Help
New Open Save Print Undo Redo Cut Copy Paste Find Replace
Clock.ctstch
Plain Text Tab Width: 8 Ln 34, Col 1 INS

```
#-- Special Route Type --
#RouteTypeName specialRoute
#TopPreferredLayer 4
#BottomPreferredLayer 3
#PreferredExtraSpace 1
#End

#-- Regular Route Type --
#RouteTypeName regularRoute
#TopPreferredLayer 4
#BottomPreferredLayer 3
#PreferredExtraSpace 1
#End

#-- Clock Group --
#ClkGroup
#+ <clockName>


#--------------------------------------------------------------
# Clock Root   : clk
# Clock Name   : 40MHz
# Clock Period : 25ns
# Clock Name   : 40MHz
# Clock Period : 25ns
#--------------------------------------------------------------
AutoCTSRootPin clk
Period         25ns
MaxDelay       0.01ns # sdc driven default
MinDelay       0ns # sdc driven default
MaxSkew        1000ps # sdc driven default
SinkMaxTran    250ps # sdc driven default
BufMaxTran     250ps # sdc driven default
Buffer         BUFFER_N INVERT_N INVERT_O
NoGating       NO
DetailReport   YES
#SetDPinAsSync  NO
#SetIoPinAsSync NO
#SetASyncSRPinAsSync  NO
#SetTriStEnPinAsSync NO
#SetBBoxPinAsSync NO
RouteClkNet    YES
PostOpt        YES
OptAddBuffer   YES
#RouteType      specialRoute
#LeafRouteType  regularRoute
END

```

FIGURE 3.25: Clock Specification File

We see the all the clock parameters- Slew, Skew, Rise time, Fall time in the file.

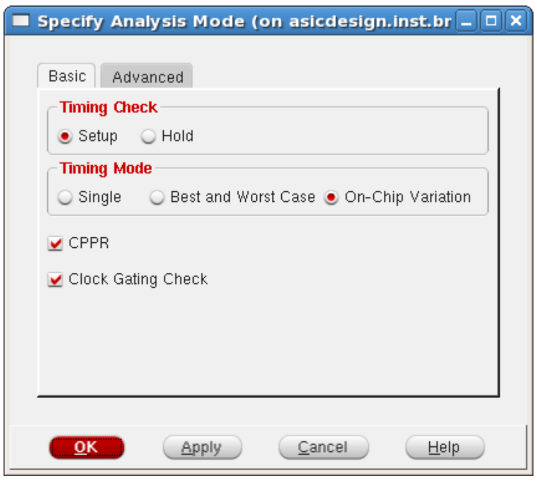

agupta@asicdesign:encounter2
File Edit View Terminal Tabs Help

```
Max displacement: 15.20 um (Instance: g36710) (850.8, 47.2) -> (861.2, 52)
        Length: 14 sites, height: 1 rows, site name: CORE, cell type: AO2222_D
Total net length = 5.733e+04 (4.326e+04 1.407e+04) (ext = 2.762e+03)
Runtime: CPU: 0:00:00.0 REAL: 0:00:00.0 MEM: 747.6MB
*** Finished refinePlace (0:01:56 mem=747.6M) ***
***** Refine Placement Finished (CPU Time: 0:00:00.0  MEM: 747.602M)

**** Clock Tree clk Stat ****
Total Clock Level       : 4
***** Top Nodes *****
clk delay[0(ps) 0(ps)] (  clk__L1_I0/A )
Level 4 (Total=666      Sink=666)
Level 3 (Total=48       Sink=0  INVERT_N=48)
Level 2 (Total=14       Sink=0  BUFFER_N=14)
Level 1 (Total=1        Sink=0  INVERT_O=1)
Total Sinks             : 666

# Analysis View: slow
********** Clock clk Pre-Route Timing Analysis **********
Nr. of Subtrees                : 1
Nr. of Sinks                   : 666
Nr. of Buffer                  : 63
Nr. of Level (including gates) : 3
Root Rise Input Tran           : 0.1(ps)
Root Fall Input Tran           : 0.1(ps)
No Driving Cell Specified!
Max trig. edge delay at sink(R): mem_array_reg[14][4]/CLK 322.6(ps)
Min trig. edge delay at sink(R): mem_array_reg[2][33]/CLK 298.2(ps)


                                (Actual)               (Required)
Rise Phase Delay               : 298.2~322.6(ps)       0~10(ps)
Fall Phase Delay               : 301~329(ps)           0~10(ps)
Trig. Edge Skew                : 24.4(ps)              1000(ps)
Rise Skew                      : 24.4(ps)
Fall Skew                      : 28(ps)
Max. Rise Buffer Tran.         : 206.4(ps)             250(ps)
Max. Fall Buffer Tran.         : 117.9(ps)             250(ps)
Max. Rise Sink Tran.           : 113.3(ps)             250(ps)
Max. Fall Sink Tran.           : 79(ps)                250(ps)
Min. Rise Buffer Tran.         : 97.7(ps)              0(ps)
Min. Fall Buffer Tran.         : 77.1(ps)              0(ps)
Min. Rise Sink Tran.           : 102.7(ps)             0(ps)
Min. Fall Sink Tran.           : 67.4(ps)              0(ps)

view slow : skew = 24.4ps (required = 1000ps)
view fast : skew = 11.2ps (required = 1000ps)


Generating Clock Analysis Report clock_report/clock.report ....
Clock Analysis (CPU Time 0:00:00.1)


*** ckSynthesis Opt Latency (cpu=0:00:00.6 real=0:00:01.0 mem=747.6M) ***
***** Start Refine Placement.....
*** Starting refinePlace (0:01:57 mem=747.6M) ***
Total net length = 5.743e+04 (4.333e+04 1.410e+04) (ext = 2.767e+03)
Move report: Timing Driven Placement moves 0 insts, mean move: 0.00 um, max move: 0.00 um
        Runtime: CPU: 0:00:00.0 REAL: 0:00:00.0 MEM: 747.6MB
Move report: Detail placement moves 0 insts, mean move: 0.00 um, max move: 0.00 um
        Runtime: CPU: 0:00:00.0 REAL: 0:00:00.0 MEM: 747.6MB
Summary Report:
Instances move: 0 (out of 608 movable)
Mean displacement: 0.00 um
```

FIGURE 3.26: Clock Synthesis Report

3.1.12 Post-CTS Optimization

Select the mode On-Chip Variation (OCV) and Common Path Pessimism Removal (CPPR).



Timing Check
[✓ Setup] Hold

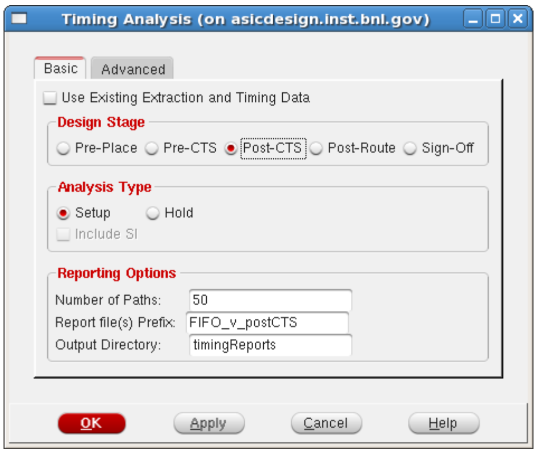
Timing Mode
Single Best and Worst Case [✓ On-Chip Variation]

[✓ CPPR]

[✓ Clock Gating Check]

FIGURE 3.27: Analysis Mode

Running the timing analysis post-CTS. After routing the clock, we once again do the timing analysis.



Timing Analysis (on asicdesign.inst.bnl.gov)

Basic | Advanced

☐ Use Existing Extraction and Timing Data

Design Stage
Pre-Place Pre-CTS [✓ Post-CTS] Post-Route Sign-Off

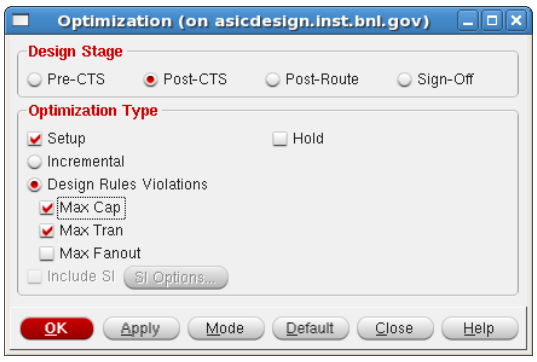
Analysis Type
[✓ Setup] Hold
☐ Include SI

Reporting Options

Number of Paths:	50
Report file(s) Prefix:	FIFO_v_postCTS
Output Directory:	timingReports

FIGURE 3.28: Post-CTS Timing Analysis

Post-CTS optimization



Optimization (on asicdesign.inst.bnl.gov)

Design Stage
Pre-CTS [✓ Post-CTS] Post-Route Sign-Off

Optimization Type
☑ Setup ☐ Hold
Incremental
[✓ Design Rules Violations]
☑ Max Cap
☑ Max Tran
☐ Max Fanout
☐ Include SI

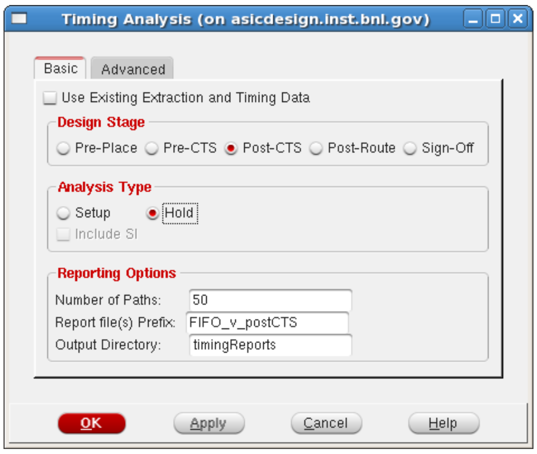
FIGURE 3.29: Post-CTS optimization

Post-CTS Optimization - Hold

Change the configuration to HOLD time from SETUP and again do the post-CTS Timing analysis.

3.1.13 Nano Route

To complete the timing closure on the design, we need to prevent crosstalk. Thus, the design is run with global and detail routing using NanoRoute[4].


Timing Analysis (on asicdesign.inst.bnl.gov)

Basic | Advanced

☐ Use Existing Extraction and Timing Data

Design Stage

○ Pre-Place ○ Pre-CTS [✓ Post-CTS] ○ Post-Route ○ Sign-Off

Analysis Type

○ Setup [✓ Hold]

☐ Include SI

Reporting Options

Number of Paths:	50
Report file(s) Prefix:	FIFO_v_postCTS
Output Directory:	timingReports

Figure 3.30: Post-CTS Hold Timing Analysis

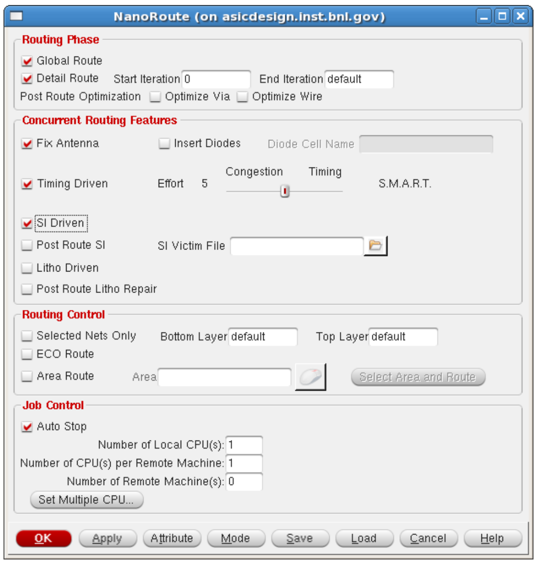
To close timing and prevent crosstalk, we enable two options in NanoRoute-

1. Timing Driven
2. SI (Signal Integrity) Driven

Note- To check the crosstalk noise through SI driven option, we have to include capacitance table (or QRC technology) files while setting up the configuration files in the MMMC browser.

The empty space in the floorplan (where standard cells are not placed) is filled up by the filler cells. These are spare filler cells which have no connection to the standard cells and are just used to fill up the gaps in the floorplan.

After placing the filler cells (generally the Clock Buffers)-



NanoRoute (on asicdesign.inst.bnl.gov)

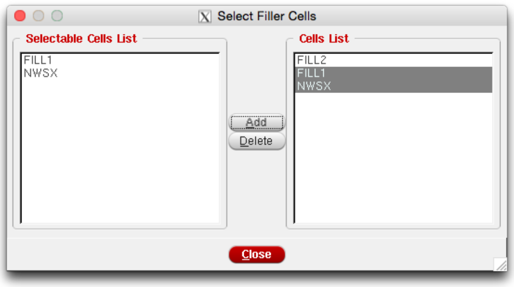
Routing Phase
- [✓] Global Route
- [✓] Detail Route   Start Iteration 0   End Iteration default
- Post Route Optimization [ ] Optimize Via [ ] Optimize Wire

Concurrent Routing Features
- [✓] Fix Antenna   [ ] Insert Diodes   Diode Cell Name
- [✓] Timing Driven   Effort 5   Congestion   Timing   S.M.A.R.T.
- [✓] SI Driven
- [ ] Post Route SI   SI Victim File
- [ ] Litho Driven
- [ ] Post Route Litho Repair

Routing Control
- [ ] Selected Nets Only   Bottom Layer default   Top Layer default
- [ ] ECO Route
- [ ] Area Route   Area   Select Area and Route

Job Control
- [✓] Auto Stop
- Number of Local CPU(s): 1
- Number of CPU(s) per Remote Machine: 1
- Number of Remote Machine(s): 0
- Set Multiple CPU...

Figure 3.31: Nano Route



Select Filler Cells

Selectable Cells List	Cells List
FILL1	FILL2
NWSX	FILL1
	NWSX

Figure 3.32: Adding Spare Cells: Filler

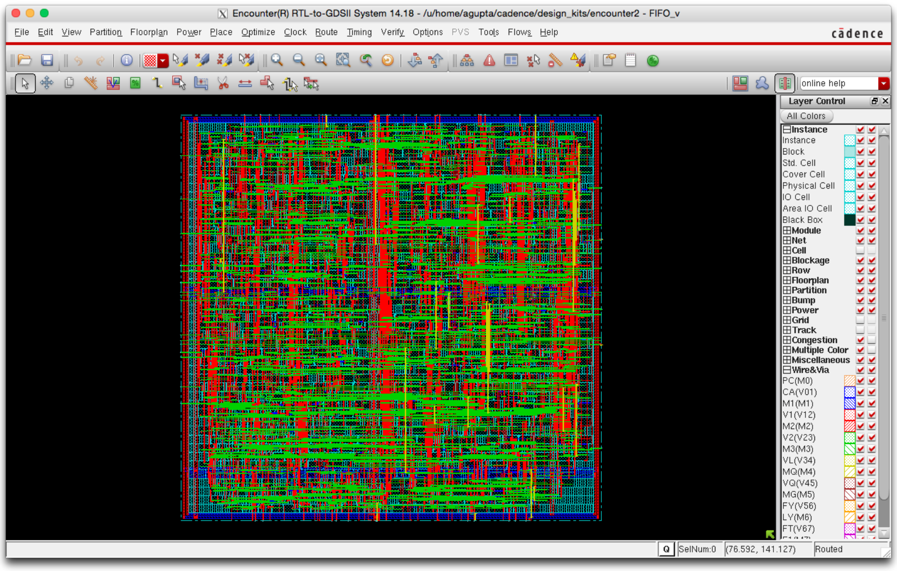


Figure 3.33: Post-placement of Fillers Cells

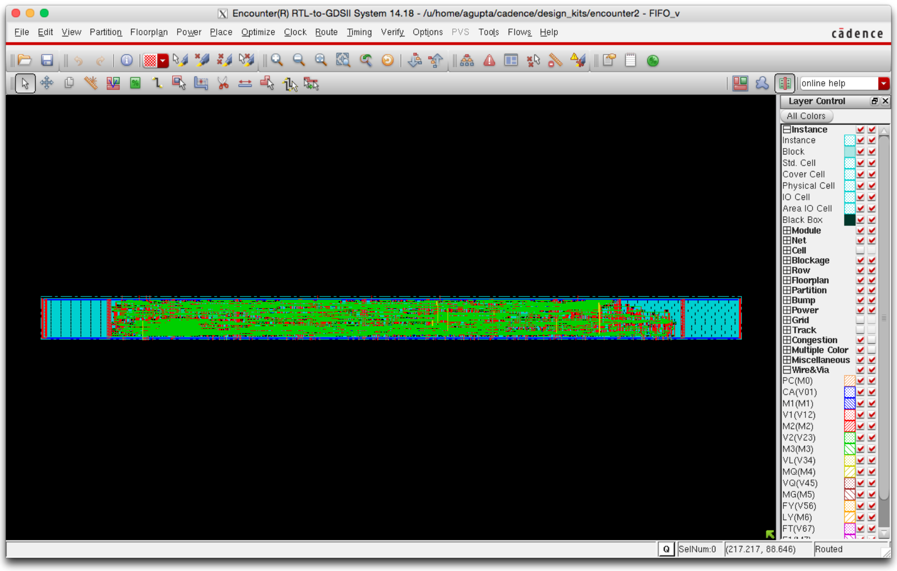
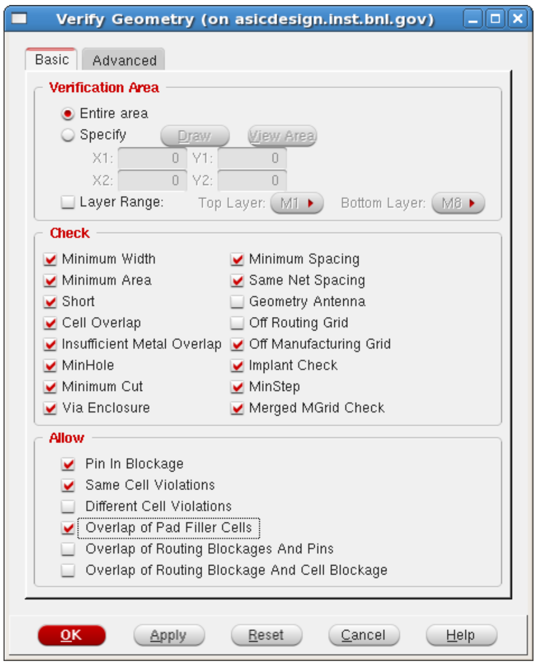


FIGURE 3.34: Post-placement of Fillers Cells -customized floorplan

3.1.14 Verify Geometry

This step is done to verify the geometry. If the technology/PDK has rules written compatible with PVS (Physical Verification Systems) tool, we can do DRC/LVS in EDI only.



Verify Geometry (on asicdesign.inst.bnl.gov)

Basic | Advanced

Verification Area

[● Entire area] ○ Specify   Draw   View Area

X1: 0   Y1: 0

X2: 0   Y2: 0

☐ Layer Range: Top Layer: M1   Bottom Layer: M8

Check

✓ Minimum Width	✓ Minimum Spacing
✓ Minimum Area	✓ Same Net Spacing
✓ Short	☐ Geometry Antenna
✓ Cell Overlap	☐ Off Routing Grid
✓ Insufficient Metal Overlap	✓ Off Manufacturing Grid
✓ MinHole	✓ Implant Check
✓ Minimum Cut	✓ MinStep
✓ Via Enclosure	✓ Merged MGrid Check

Allow

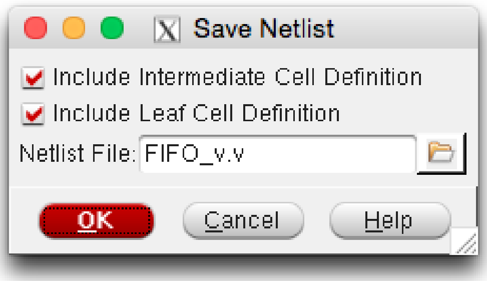
✓ Pin In Blockage
✓ Same Cell Violations
☐ Different Cell Violations
✓ Overlap of Pad Filler Cells
☐ Overlap of Routing Blockages And Pins
☐ Overlap of Routing Blockage And Cell Blockage

OK   Apply   Reset   Cancel   Help

FIGURE 3.35: Verify Geometry

3.1.15 Netlist Export

Since, we have routed many nets (CTS), the final netlist imported in the encounter has modified a lot. At last, we export the final netlist (.v) file from the encounter. This file will serve as our schematic when we do DRC/LVS in Virtuoso.


Save Netlist

✓ Include Intermediate Cell Definition

✓ Include Leaf Cell Definition

Netlist File: FIFO_v.v

Figure 3.36: Netlist Export

3.1.16 GDS Export

After performing all the steps, our final design is shown in figure below

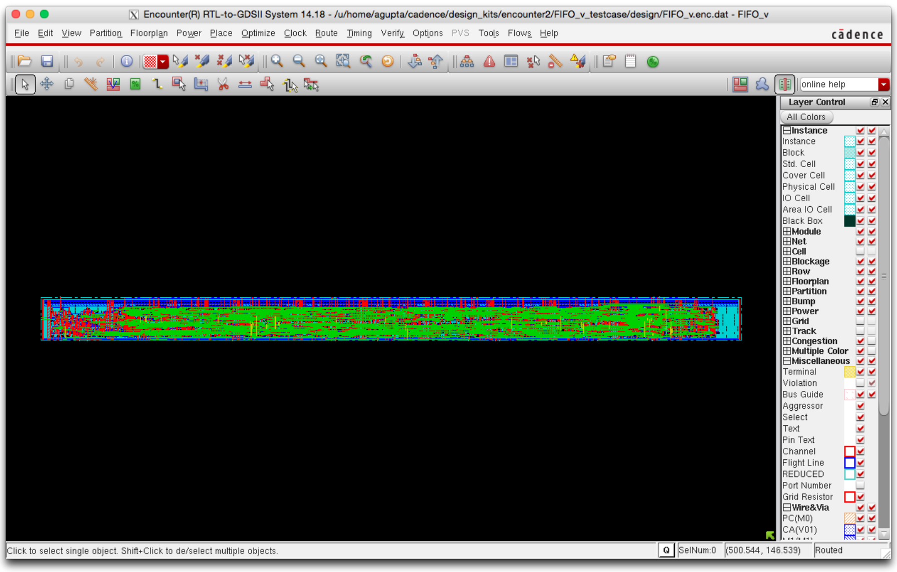
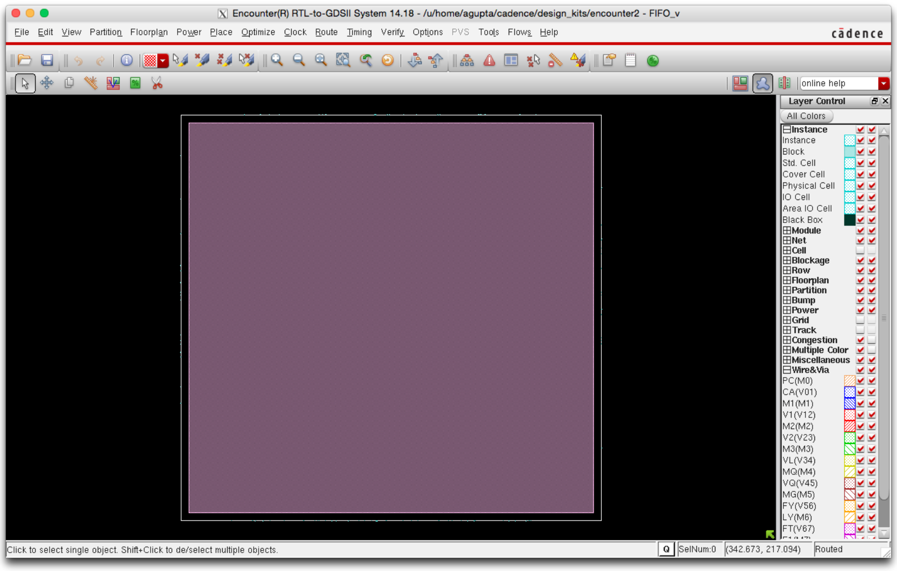

Encounter(R) RTL-to-GDSII System 14.18 - /u/home/agupta/cadence/design_kits/encounter2/FIFO_v_testcase/design/FIFO_v.enc.dat - FIFO_v
File Edit View Partition Floorplan Power Place Optimize Clock Route Timing Verify Options PVS Tools Flows Help
cadence
online help
Layer Control
All Colors
Instance
Module
Net
Cell
Blockage
Row
Floorplan
Partition
Bump
Power
Grid
Track
Congestion
Multiple Color
Miscellaneous
Wire&Via
Click to select single object. Shift+Click to de/select multiple objects.
SelNum:0
(500.544, 146.539)
Routed

Figure 3.37: Final design

Final Design in the Amoeba view looks like-

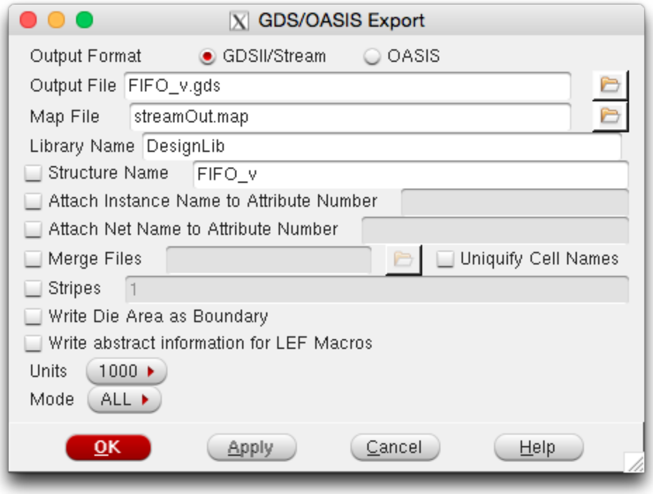


Click to select single object. Shift+Click to de/select multiple objects. SelNum:0 (342.673, 217.094) Routed

FIGURE 3.38: Amoeba View of the design

GDSII (Graphic Design Systems) Export

The design is exported in GDSII format to verify DRC/LVS in Virtuoso if not done in PVS in EDI. This export will create strMapOut file.



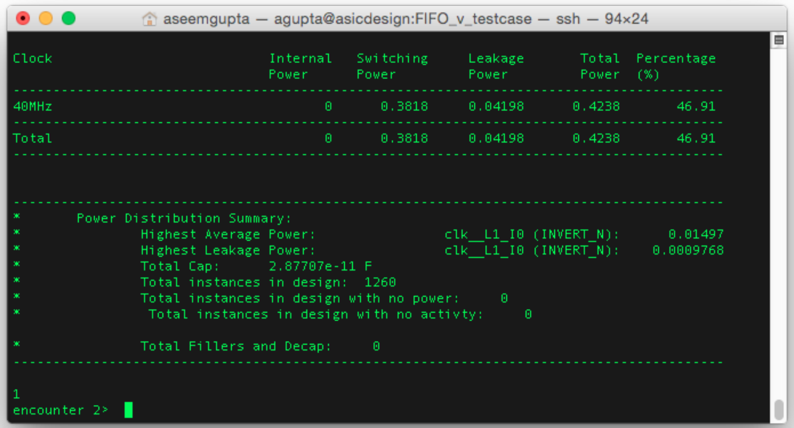
Output Format	[✓ GDSII/Stream] OASIS
Output File	FIFO_v.gds
Map File	streamOut.map
Library Name	DesignLib
☐ Structure Name	FIFO_v
☐ Attach Instance Name to Attribute Number	
☐ Attach Net Name to Attribute Number	
☐ Merge Files	☐ Uniquify Cell Names
☐ Stripes	1
☐ Write Die Area as Boundary	
☐ Write abstract information for LEF Macros	
Units	1000
Mode	ALL

FIGURE 3.39: GDSII Export

3.1.17 Power

We can use report power command to get the tool give power of the design. The command from the terminal is -

report_power



aseemgupta — agupta@asicdesign:FIFO_v_testcase — ssh — 94×24

```
Clock                           Internal   Switching     Leakage       Total  Percentage
                                Power      Power         Power         Power  (%)
------------------------------------------------------------------------------------------
40MHz                                  0      0.3818     0.04198      0.4238       46.91
------------------------------------------------------------------------------------------
Total                                  0      0.3818     0.04198      0.4238       46.91
------------------------------------------------------------------------------------------


------------------------------------------------------------------------------------------
*       Power Distribution Summary:
*               Highest Average Power:                clk__L1_I0 (INVERT_N):       0.01497
*               Highest Leakage Power:                clk__L1_I0 (INVERT_N):     0.0009768
*               Total Cap:      2.87707e-11 F
*               Total instances in design:  1260
*               Total instances in design with no power:     0
*                Total instances in design with no activty:     0

*               Total Fillers and Decap:     0
------------------------------------------------------------------------------------------

1
encounter 2> 
```

FIGURE 3.40: Power

Power of the design is as follows-

At 40MHz frequency,

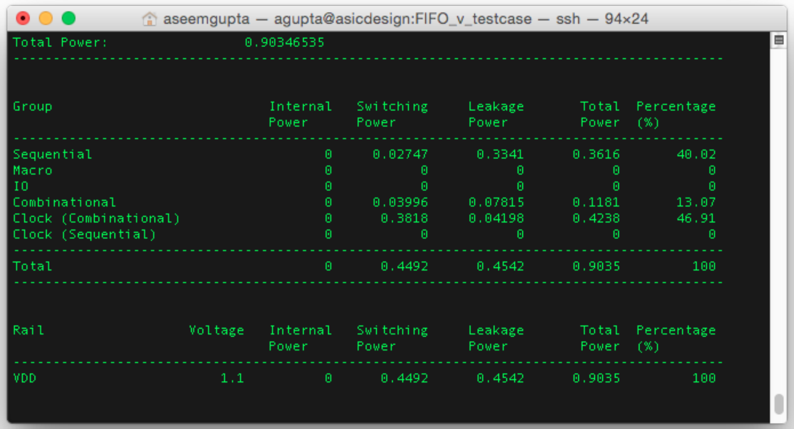
Switching power - 0.38 mW

Leakage power - 0.041 mW

3.1.18 Area

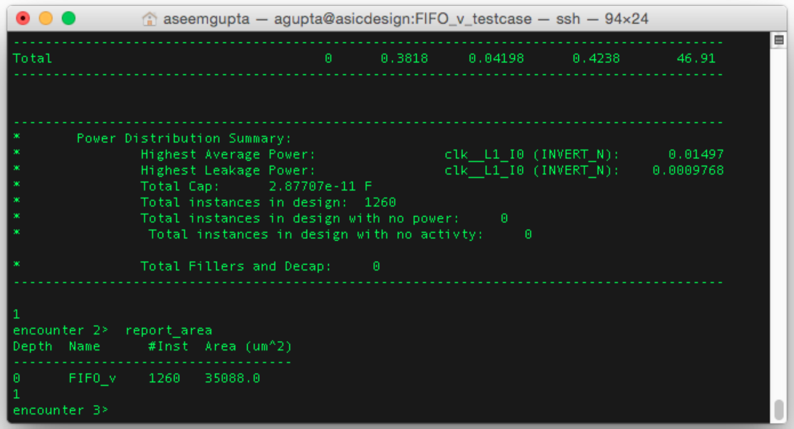
We can use report area command to get the tool give the area of the design. The area reported was approax 35000 sq. um. Total area with nets is around 65000 sq um as reported after synthesis also.

The command from the terminal is report_area.



```
Total Power:                 0.90346535
------------------------------------------------------------------------------------------


Group                           Internal    Switching     Leakage       Total  Percentage
                                Power       Power         Power         Power  (%)
------------------------------------------------------------------------------------------
Sequential                             0     0.02747      0.3341      0.3616       40.02
Macro                                  0           0           0           0           0
IO                                     0           0           0           0           0
Combinational                          0     0.03996     0.07815      0.1181       13.07
Clock (Combinational)                  0      0.3818     0.04198      0.4238       46.91
Clock (Sequential)                     0           0           0           0           0
------------------------------------------------------------------------------------------
Total                                  0      0.4492      0.4542      0.9035         100
------------------------------------------------------------------------------------------


Rail                  Voltage   Internal    Switching     Leakage       Total  Percentage
                                Power       Power         Power         Power  (%)
------------------------------------------------------------------------------------------
VDD                       1.1          0      0.4492      0.4542      0.9035         100
```

FIGURE 3.41: Power II



```
------------------------------------------------------------------------------------------
Total                                  0      0.3818     0.04198      0.4238       46.91
------------------------------------------------------------------------------------------



------------------------------------------------------------------------------------------
*       Power Distribution Summary:
*               Highest Average Power:                 clk__L1_I0 (INVERT_N):      0.01497
*               Highest Leakage Power:                 clk__L1_I0 (INVERT_N):    0.0009768
*               Total Cap:      2.87707e-11 F
*               Total instances in design:  1260
*               Total instances in design with no power:      0
*                Total instances in design with no activty:     0

*               Total Fillers and Decap:     0
------------------------------------------------------------------------------------------


1
encounter 2>  report_area
Depth  Name      #Inst  Area (um^2)
-----------------------------------
0      FIFO_v    1260   35088.0
1
encounter 3>
```

FIGURE 3.42: Area

We finish the design here. Virtuoso Export and mixed-signal integration would be performed in the upcoming project while integrating this digital implementation in a bigger top-level layout.

Chapter 4

Radiation-Hardened-by-Design (RHBD)

4.1 Single Event Upset

Silicon chips used in applications where they are exposed to ionizing radiations such as outer-space applications, high-ionized ion source (Large Hadron Collider, Heavy Ion Collider) must be hardened against the radiations in order to make them function properly. These radiations can cause undesirable effects in the devices, such as flipping the state of the memory cells or other effects caused by accumulation of trapped charges induced by radiation.

The error induced by an incident particle on a digital integrated circuit (IC) is called a Single Event Upset (SEU). A large number of electron-hole pairs is generated, for an equivalent charge in excess to 1pC, by the incident particle which follows its way to the circuit node making it a sensitive node. This perturbation may propagate through the circuit resulting in operational errors[1]. For instance, if a circuit node with voltage interruptions due to incident particles belongs to a memory state (latch), it is highly likely to flip the state of the memory thereby completely reversing the logic causing a SEU. These kind of errors/flaws in the design can be very hazardous for the functioning of the chip. Imagine a SEU causing RESET bit to flip, thereby causing the entire functionality to go in reset mode.

With feature size getting smaller and smaller, the problem of SEU has increased manifold. Smaller feature sizes translate to smaller capacitances and held charges, allowing radiation of even smaller energies to upset vulnerable/sensitive nodes[2].

SEU mitigation is necessary to ensure data integrity. Semiconductor Associations have realized the need of building radiation-tolerant ASIC's and have come up with many Radiation-Hardened-by-Design (RHBD) techniques[2].

One such technique is the Dual Interlocked Cell Storage (DICE). Before that, lets see the functioning of a basic Flip-Flop Cell.

The D-Flip Flop operates in two mode-

1. Master
2. Slave

Both configurations have back-to-back inverters as shown in the figure 4.1

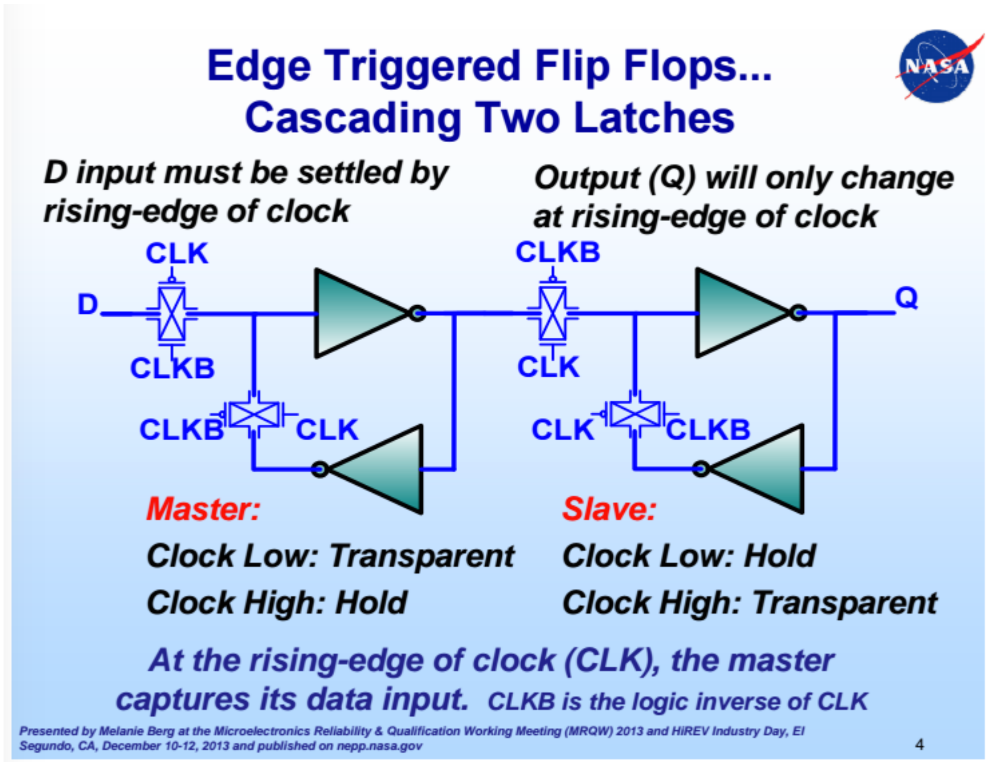
Master - Clk Low - Transparent Mode Clk High- Hold mode

At the rising edge of the clock (positive-edge triggered), the output Q of a flip-flop follows input data D. Master configuration of FF passes the data (Transparent) when the clock is low and holds the information when the clock is high. This is achieved by back-to-back inverters topology, thus making a memory.

Slave - Clk Low - Hold Mode Clk High - Transparent mode

In slave configuration, the data is in hold mode when clock is low and is passed when clock is high. Transmission gate is one topology to use as a switch.

With this Master-Slave configuration, we achieve proper functioning of Flip-flop at clock edges.



Edge Triggered Flip Flops...
Cascading Two Latches

D input must be settled by rising-edge of clock

Output (Q) will only change at rising-edge of clock

Master:
Clock Low: Transparent
Clock High: Hold

Slave:
Clock Low: Hold
Clock High: Transparent

At the rising-edge of clock (CLK), the master captures its data input. *CLKB is the logic inverse of CLK*

Presented by Melanie Berg at the Microelectronics Reliability & Qualification Working Meeting (MRQW) 2013 and HiREV Industry Day, El Segundo, CA, December 10-12, 2013 and published on nepp.nasa.gov 4

Figure 4.1: Master-Slave DFF

The DICE-DFF configuration shown in Figure 4.2 is able to behave as radiation-tolerant configuration for milliCoulombs of charge. The charge pulse of 50mA current and 1ns time period (50pC charge) was injected to see if the configuration was able to sustain the charge. The result was found out to be positive. On the contrary, normal Flip Flop configuration can sustain only femtoCoulombs of charge.

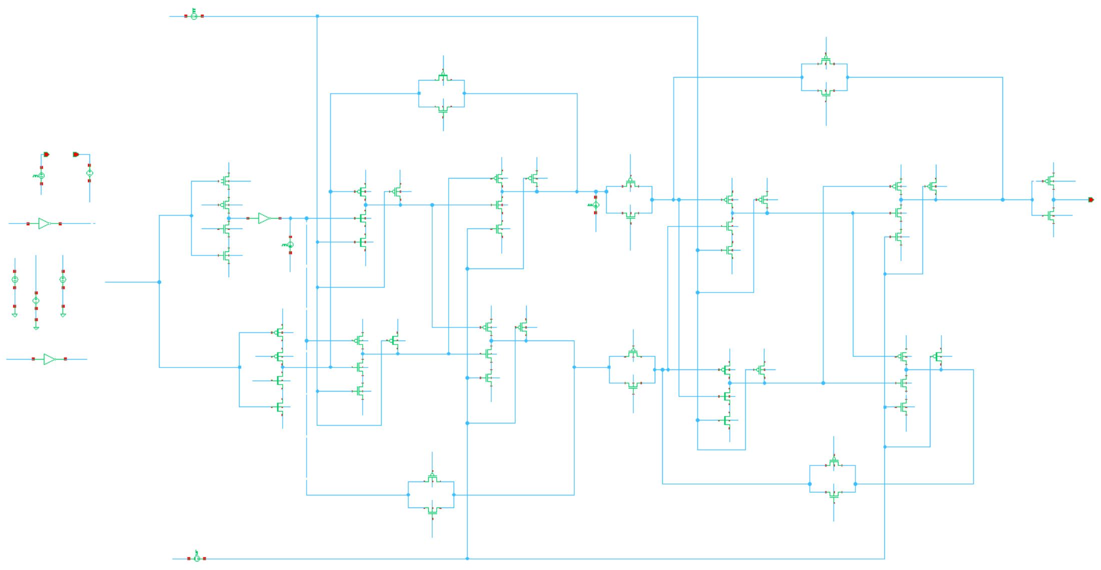


FIGURE 4.2: Dual Interlocked Storage Cell Flip Flop

4.2 Standard Cells Layout

Technology - TSMC 130nm Cell - DFF

The digital ground was separated from the analog substrate. The D flip-flop is only a reset Flip Flop.

Technology - TSMC 130nm Cell - DFFSR

This configuration of DFF is the set-reset configuration.

All these design steps have helped me complete my thesis. Through this thesis, I have learnt to draw efficient layouts, study and analyze SEU designs and developed strong capabilities working on RTL-to-GDSII design flow using CAD tools.

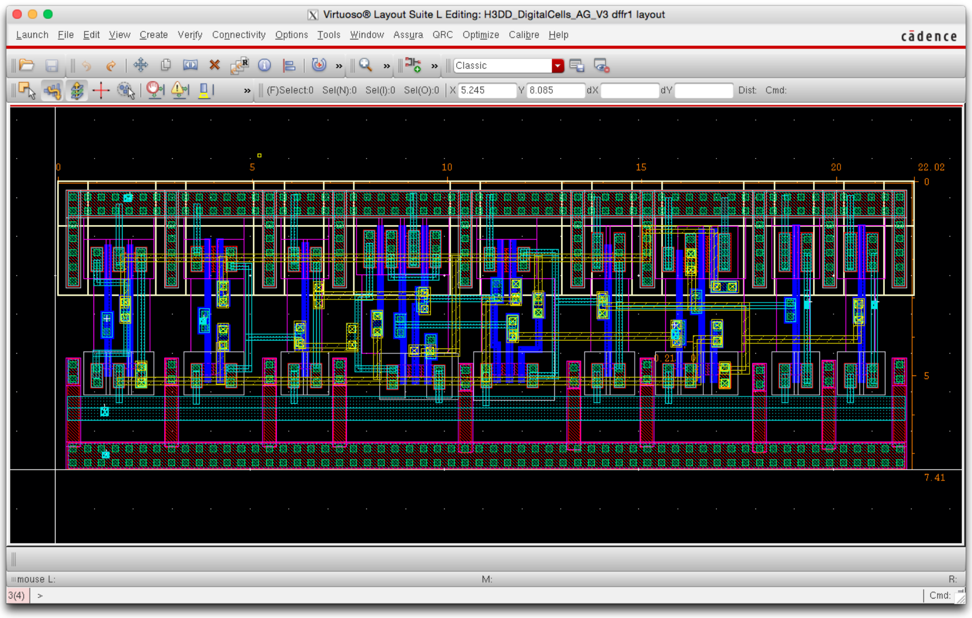

Virtuoso® Layout Suite L Editing: H3DD_DigitalCells_AG_V3 dffr1 layout
Launch File Edit View Create Verify Connectivity Options Tools Window Assura QRC Optimize Calibre Help
cādence
Classic
(F)Select:0 Sel(N):0 Sel(I):0 Sel(O):0 X 5.245 Y 8.085 dX dY Dist: Cmd:
0
5
10
15
20
22.02
5
7.41
mouse L:
M:
R:
3(4) >
Cmd:

FIGURE 4.3: Layout -D Flip Flop(reset) - TSMC 130nm

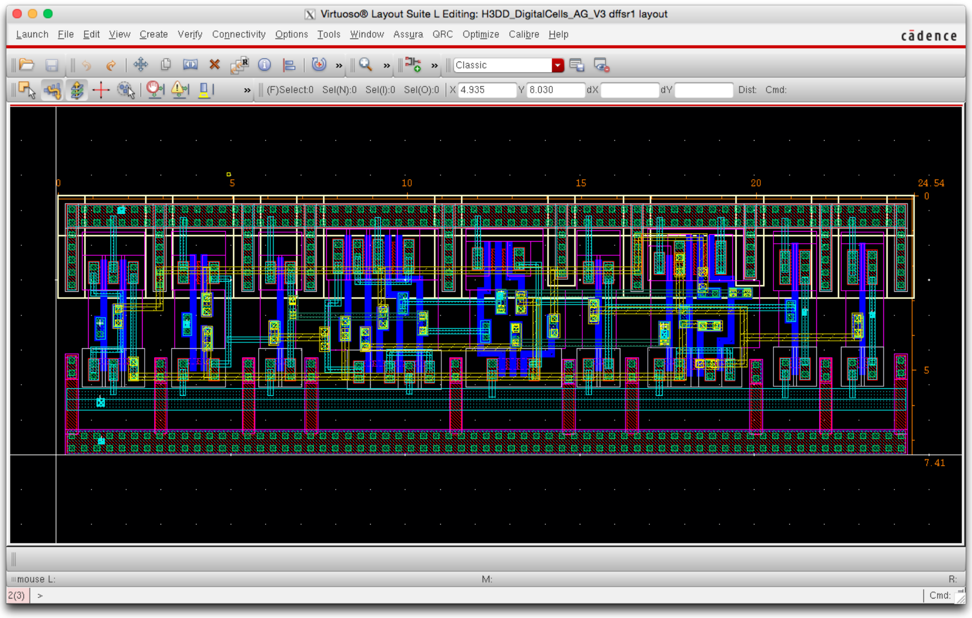

Virtuoso® Layout Suite L Editing: H3DD_DigitalCells_AG_V3 dffsr1 layout
Launch File Edit View Create Verify Connectivity Options Tools Window Assura QRC Optimize Calibre Help
cādence
Classic
(F)Select:0 Sel(N):0 Sel(I):0 Sel(O):0 X 4.935 Y 8.030 dX dY Dist: Cmd:
0
5
10
15
20
24.54
5
7.41
mouse L:
M:
R:
2(3) >
Cmd:

FIGURE 4.4: Layout- D Flip Flop (set/reset) - TSMC 130nm

Abbreviations

FIFO	**F**irst-**i**n **F**irst-**o**ut
CAD	**C**omputer-**A**ided **D**esign
VLSI	**V**ery **L**arge **S**cale **I**ntegration
CDS	**C**adence **D**esign **S**ystems
MGC	**M**entor **G**raphics **C**orporation
EDA	**E**lectronic **D**esign **A**utomation
ASIC	**A**pplication **S**pecific **I**ntegrated **C**ircuit
FPGA	**F**ield **P**rogrammable **G**ate **A**rray
SoC	**S**ystem-on-**C**hip
RTL	**R**egister **T**ransfer **L**evel
HDL	**H**ardware **D**escription **L**anguage
VHDL	**V**ery High Speed IC **H**ardware **D**escription **L**anguage
PPA	**P**ower **P**erformance **A**rea
ECO	**E**ngineering **C**hange **O**rder
Q-o-S	**Q**uality-of-**S**ilicon
Q-o-R	**Q**uality-of-**R**esults
TTM	**T**ime-to-**M**arket
TAT	**T**urn**A**round **T**ime
EDI	**E**ncounter **D**igital **I**mplementation
CPF	**C**ommon **P**ower **F**ile
SDC	**S**ynopsys **D**esign **C**onstraint
SDF	**S**tandard **D**elay **F**ormat
LEC	**L**ogic **E**quivalence **C**hecking
CCD	**C**onformal **C**onstraint **D**esigner

CLP	**C**onformal **L**ow **P**ower
LEF	**L**ibrary **E**xchange **F**ormat
DEF	**D**esign **E**xchange **F**ormat
MMMC	**M**ulti **M**ode **M**ulti **C**orner
STA	**S**tatic **T**iming **A**nalysis
CTS	**C**lock **T**ree **S**ynthesis
GTD	**G**lobal **T**iming **D**ebug
OCV	**O**n **C**hip **V**ariation
CPPR	**C**ommon **P**ath **P**essimism **R**emoval
AAE	**A**dvanced **A**nalysis **E**ngine
SI	**S**ignal **I**ntegrity
GDSII	**G**raphic **D**esign **S**ystems II
LIB	**L**iberty Files
ILM	**I**nterface **L**ogic **M**odels
TLM	**T**ransaction **L**evel **M**odelling
DRC	**D**esign **R**ule **C**heck
ERC	**E**lectrical **R**ule **C**hecking
DRV	**D**esign **R**ule **V**iolations
LVS	**L**ayout vs **S**chematic
PVS	**P**hysical **V**erification **S**ystem
FHM	**F**ast **H**ardware **M**odels
ELS	**E**mbedded **L**ogic **S**ynthesis
BST	**B**ehavior **S**tructure **T**iming
CFS	**C**onstraint **F**unctionality **S**eparation
CTOS	**C**-to-**S**ilicon
HLS	**H**igh **L**evel **S**ynthesis
BIST	**B**uilt-**I**n **S**elf **T**est
ATPG	**A**utomated **T**est **P**attern **G**eneration
JTAG	**J**oint **T**est **A**ction **G**roup
DFT	**D**esign **F**or **T**estability
DFM	**D**esign **F**or **M**anufacturability

Bibliography

[1] An SEU-Tolerant DICE Latch Design With Feedback Transistors, IEEE Transactions on Nuclear Science, Vol. 62, No.2, April 2015

[2] The 90nm Double-DICE Storage Element To Reduce Single-Event Upsets, IEEE 978-1-4244-4480-9/09(c)2009 IEEE

[3] Revisiting Dual Interlocked Cell Storage (DICE) Single Event Upset (SEU) Sensitivity, NASA at Microelectronic Reliability & Qualification Working (MRQW) meeting 2013 and HIREV Industry Day, published on nepp.nasa.gov

[4] Cadence Online Support: Rapid Adoption Kit (RAK), Introduction to Encounter Digital Implementation (EDI) System 13.2 and Block Implementation Flow, Cadence Design Systems Inc.

[5] Cadence CDNLive Boston 2014 Proceedings: Digital & Sign-off

[6] Verilog HDL: A guide to Digital Design & Synthesis by Samir Palnitkar

[7] Top-Down Digital Design Flow by Alain Vachoux, Microelectronic Systems Lab, STI-IEL-LSM

[8] Understanding Synchronous FIFOs, Cypress, AN1042