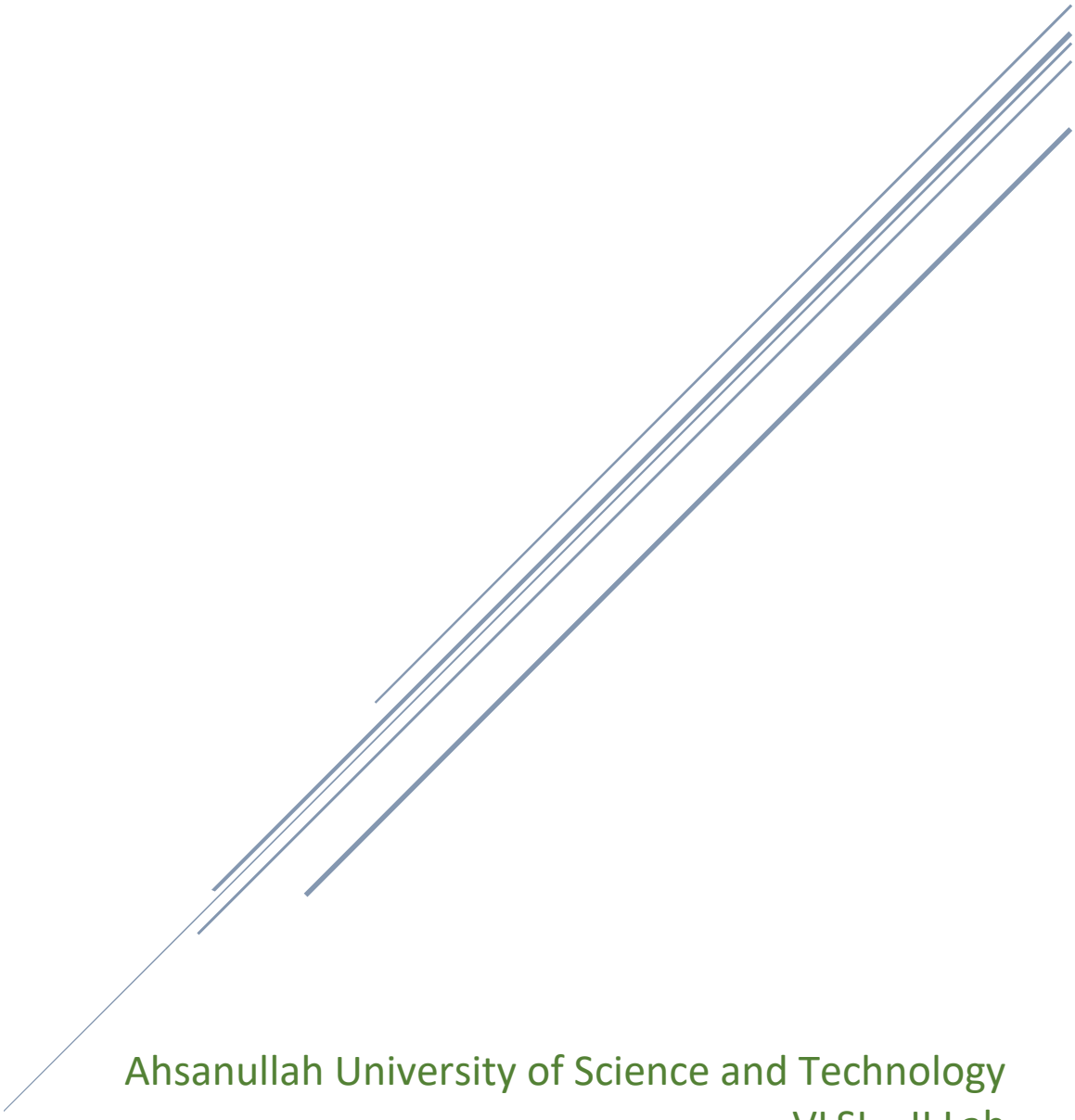
SYNTHESIS, PLACE AND ROUTING OF ARITHMETIC LOGIC UNIT

[Cadence Tools (Modelsim, Genus, Encounter)]



Ahsanullah University of Science and Technology
VLSI – II Lab

Chapter - I

Introduction

Recent changes in the technology market demand faster turnaround of the design, and as a result, designers struggle to meet performance requirements under prohibitively expensive nonrecurring engineering (NRE) costs. Increasing costs for the design, validation, and time to market are some of the most pressing issues for next generation microprocessor systems. Custom versus synthesis VLSI circuit design has been a lively debate for the last decade. However, the ever-increasing cost of designs in large-scale projects is becoming a bottleneck to the industry, thus, the wind is blowing strongly in favor of synthesis, especially when the modern synthesis engines are becoming more and more sophisticated in closing timing, completing routings, avoiding congestions and better interface with all the backend tools. Custom design advocates will argue that the last pico-second and milli-watts cannot be left on the table, but the reality is, they need to get over this mentality in favor of cost, time to market, and ability to adopt the last-minute change of the design.

Integrated Circuit (IC) design complexity has increased radically since the first" hand-made" designs in the late 50s, with a few transistors. Nowadays, Very Large-Scale Integration (VLSI) designs contains hundred thousand, million or even billion transistors and not only the experience of the designer, but also Electronic Design Automation (EDA) tools and some methodology is needed.

In this project I have explored some important tool of Cadence Design Suits, such as, Cadence Genus, Cadence Encounter (Innovus). This is a project which is to build different module of processor chip, including arithmetic logic unit (ALU).

1.1 Project Goals

Therefore, there are two goals to be achieved in this project:

- The main purpose is to complete a VLSI design flow in 45 nm technology with up-to date EDA tools (Design Compiler from Synopsys and SoC Encounter from Cadence) in order to set up a design environment for more complex and specific designs.
- Learn about the synthesis process briefly with Cadence Genus tool and learn about the constrains, library files, RTL, captable and many more file systems.
- Place and Routing (PnR) is a very important and crucial part of VLSI design flow. With the help of SoC Encounter or also known as innovus tool PnR process was explored and verifying and fixing connectivity, geometry, design rules check (DRC), and learning static timing analysis (STA) was goal of this project.

1.2 Project Specifications

The specification of project has given below.

Register Transfer Level (RTL):

RTL file has given by the instructor. Link is:

test_pro.v - Google Drive

SDC Constraint:

1. Clock frequency “4<My last two digits of roll number> MHz”. My ID is: 160105170. So, for this case last two digit is 70. Hence, Clock frequency is 470MHz.
2. Uncertainty for setup “0.35” & hold is “0.3”.
3. Maximum transition “2” [overall design]
4. Clock transition “min” “0.3” & “max” “0.3” for both “fall” and “rise”
5. Driving cell “BUFX2”
6. Operating conditions “slow”
7. Min input & output delay is “0” and the maximum is “0.5”
8. Output load “0.5”
9. Max Fanout is “10”

Design Constraints:

1. Placement Density (after import of the design) should be greater than or equal to “70” % and the ratio should be 1:3.
2. Maximum Routing Layer up to M3.
3. Top Metal should be M4.
4. Place input pins at the “top” side and output pins at the “bottom” side.
5. Use placement blockage around the block.
6. Die boundary is not required.
7. Add Filler after clean the violations.

Chapter - II

ASIC Design Flow

An application-specific integrated circuit (ASIC) is a chip designed or customized for a special use, for example, a particular kind of transmission protocol or a palmtop computer. The diverse steps of a typical design flow are shown in Figure 1[1].

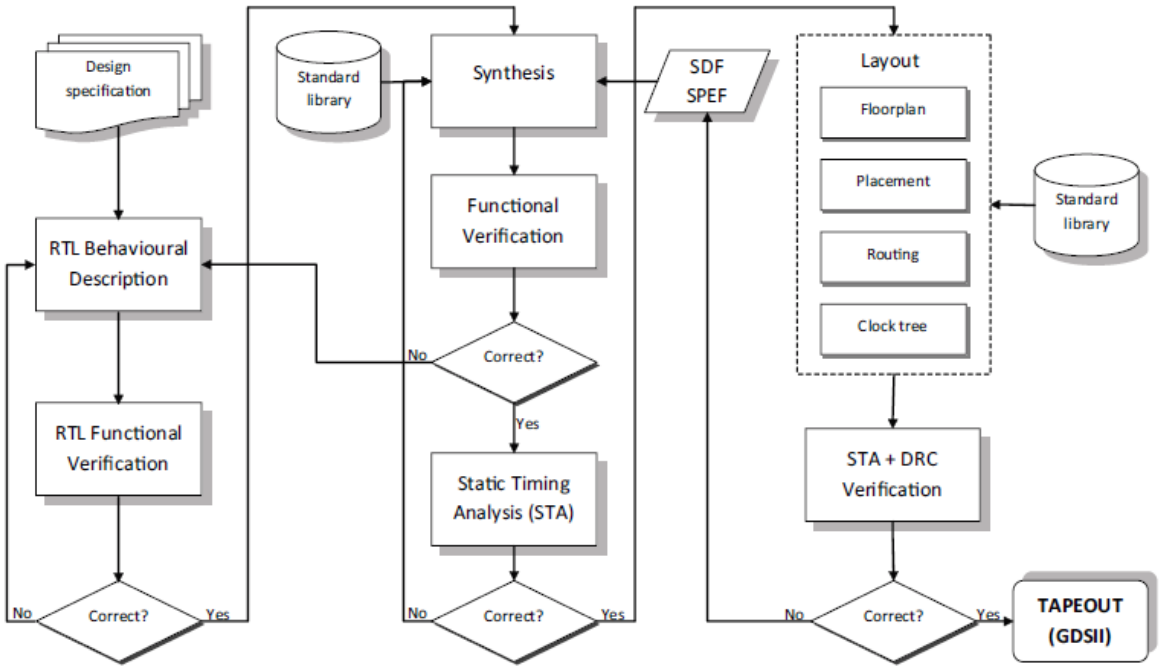


Figure 1: General VLSI Design Flow example

Hardware description language codes are designed in the EDA tools like ModelSim and simulated to check functionality. This model works as reference for observing the behavior of the design. By looking at the reference code, the RTL code is written and then verified whether it is meeting the desired functionality or not. RTL describes the hardware in terms of registers and the combinatorial logic. The generated RTL is then synthesized using synthesis tools and then sent to backend flow.

Logic synthesis is done by RTL for performing gate-level synthesis. Floor plan is performed to identify where the functional cells should be placed in limited silicon area. Place and route are done after floor planning is over, and cells are connected by wires in an optimized manner. After this, layout verses schematic is performed. All the design tests are carried out, and if the designer confirms that specifications are met then the chip is ready to be fabricated.

2.1 Register-transfer-level (RTL)

RTL is the most important step in ASIC design. This is because bad RTL leads to bad outputs and synthesis failure. RTL is based on synchronous logic and contains three primary pieces, namely registers which hold state information, combinatorial logic which defines the next state inputs and an input clock that controls the change of states. There are two widely used RTL design approaches [2]:

- Algorithmic state machine (ASM) chart
- Datapath and control path design.

For this project a RTL code of ALU was given which was written using Verilog HDL and named test_pro.v (Here .v is the extension of the Verilog file).

Chapter III

Synthesis

Synthesis is performed on an RTL to convert the design into gate-level netlist. A gate-level netlist can be explained as interconnection of basic logic gates such as NAND, NOR, AND, XOR, XNOR, OR and NOT in a structural flow. Logic synthesis uses standard cell library which has simple cells, such as basic logic gates like AND, OR and NOR, or macro-cells, such as adder, multiplexers, memory and specific flip-flops. Figure 2 shows the synthesis flow in ASIC design[2].

Synthesis = translation + optimization + mapping

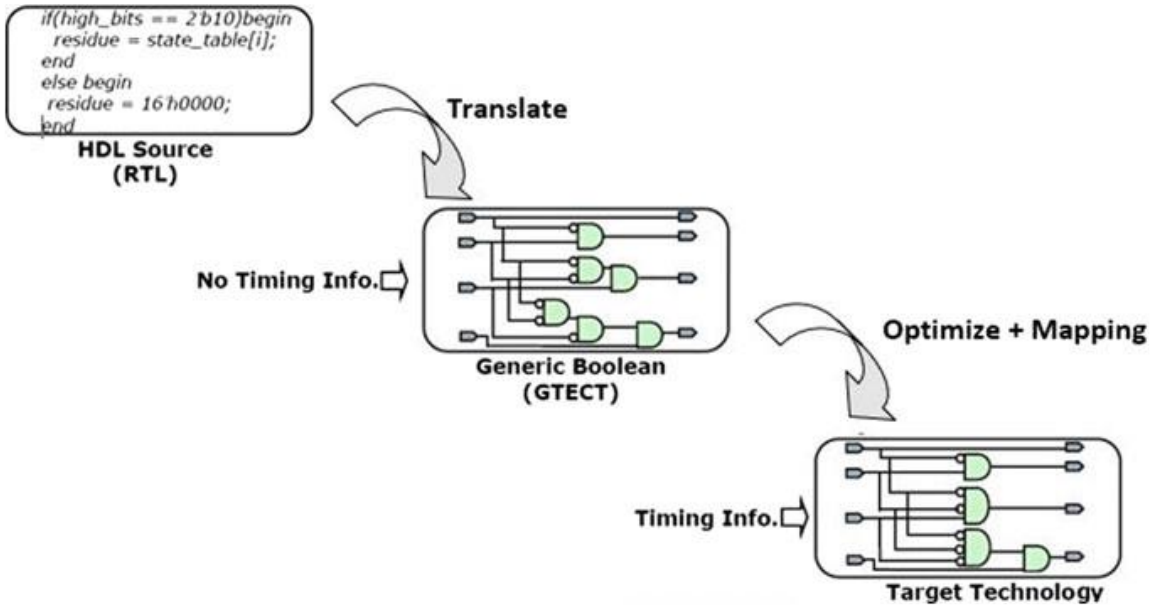


Figure 2: Generalized logic synthesis flow

Synthesis includes the following steps:

- *RTL Synthesis and Library Mapping:* The Verilog code is translated into a netlist of interconnected general gates and registers and then it is mapped to the particular gates defined in the target library. Design Compiler Cadence genus tool is a typical program used for this task.
- *Functional Verification:* At this point, it is necessary to check if the structural netlist performs the same function as the original behavioral HDL. The easiest way is to run again the test benches that have been used in the behavioral step.
- *Static Timing Analysis:* The circuit is behavioral and structural equivalent but timing requirements has to be tested. A static timing analysis can be run quickly to check if the circuit is fast enough or if we should come back to a previous design step and redesign our project.

Generic Design Flow:

The details of generic design flow with given specifications has given below [3-4, 6].

Commands	**Meaning of Commands**
Set the paths	
set_db init_lib_search_path ./timing_libs	Sets the path of timing files. For this project we used three file • Fast.lib • Slow.lib • Typical.lib
set_db hdl_search_path input_files/	Sets the input file path. In this project it is the path of test_pro.v file
Load the library	
set_db lef_library ./lef/gsclib045.lef	Set the specified lef or library file.
Load and elaborate the design	
read_hdl <your_rtl_name> *read_hdl test_pro.v*	Loads one or more HDL files in the order given into memory. Files containing macro

	definitions should be loaded before the macros are used.
elaborate	Creates a design from a Verilog module or from a VHDL entity and architecture. The command returns the directory path to the top-level design(s) that it creates.
Specify timing and design constraints	
set_top_module <rtl_top_module_name> *set_top_module proccessor.v*	Sets the name of top module
current_design <rtl_top_module_name>	
write_hdl > elaborated.v	Generates one of the following design implementations in Verilog format: • A structural netlist using generic logic • A structural netlist using mapped logic
set verilogout_no_tri true	
set verilogout_unconnected_prefix "UNCONNECTED"	
set verilogout_show_unconnected_pins true	
set verilogout_single_bit false	
set edifout_netlist_only true	
set CLK_PERIOD 2.12	Sets the Clock period. According to specification clock period is 2.12
create_clock -name clk -period $CLK_PERIOD [get_ports clk]	
set_clock_uncertainty -setup 0.35 [get_clocks clk]	Sets the uncertainty setup time. According to specification setup time is 0.35
set_clock_uncertainty -hold 0.3 [get_clocks clk]	Sets the uncertainty hold time. According to specification hold time is 0.3

set_max_transition 2 [current_design]	According to specification Maximum transition is 2.
set_clock_transition -min -fall 0.3 [get_clocks clk]	According to specification Clock transition “min” “0.3” for both “fall” and “rise”
set_clock_transition -min -rise 0.3 [get_clocks clk]	
set_clock_transition -max -rise 0.3 [get_clocks clk]	According to specification Clock transition “min” “0.3” for both “fall” and “rise”
set_clock_transition -max -fall 0.3 [get_clocks clk]	
set_clock_groups -name original -group [list [get_clocks clk]]	
set DRIVING_CELL BUFX2	Sets the Driving cell. According to specification Driving cell “BUFX2”
set DRIVE_PIN {Y}	
set_driving_cell -lib_cell $DRIVING_CELL -pin $DRIVE_PIN [all_inputs]	
set_max_fanout 10 [current_design]	Sets the maximum fanout. According to specification maximum fanout is 10
set_load 0.5 [all_outputs]	According to specification Load is 0.5
set_operating_conditions slow	Sets the operating conditions. According to specification operating condition is SLOW
set MAX_INPUT_DELAY 0.5	Maximum input and output delay is 0.5
set MAX_OUTPUT_DELAY 0.5	
set MIN_INPUT_DELAY 0	Minimum input and output delay is 0.5
set MIN_OUTPUT_DELAY 0	
set_input_delay -max $MAX_INPUT_DELAY [all_inputs]	

set_output_delay -max $MAX_OUTPUT_DELAY [all_outputs]	
remove_assign -buffer_or_inverter BUFX12 -design <design_name>	
synthesize -to_mapped	Determines the most suitable design implementation using the given design constraints (clock cycle, input delays, output delays, technology library, and so on). The synthesize command takes a list of top-level designs, synthesizes the RTL blocks, optimizes the logic, and performs technology mapping. *-to_mapped* => Maps the specified design(s) to the cells described in the supplied technology library and performs logic optimization. The aim of the optimization is to provide the smallest possible implementation of the synthesized design that still meets the supplied timing goal.
remove_assigns_without_opt -buffer_or_inverter BUFX12 -verbose	Controls the aspects of the replacement of assign statements in the design with buffers or inverters without performing incremental optimization. *-buffer_or_inverter libcell* => Specifies the buffer or inverter to be used to replace the assign statements. The specified cell must be part of a library that was loaded. *-verbose* =>Displays all messages during assign removal.

set_remove_assign_options - buffer_or_inverter BUFX12 -verbose	Controls the aspects of the replacement of assign statements in the design with buffers or inverters, which is controlled by the remove_assigns root attribute. The actual replacement happens during the next incremental optimization run.
Export Design	
write -mapped > <rtl_name>_synth.v *write-mapped testpro_synth.v*	Writes out the synthesized file of the current design
write_sdc > <rtl>.sdc *write_sdc testpro.sdc*	Writes out the current design constraints in Synopsys Design Constraint (SDC) format.
exit	Exits the Genus tool

After completing synthesis using genus tool we found two output files. These files are:

testpro_synth.v

testpro.sdc

These files are the input of Cadence Encounter (Innovus) tool, which perform the place and routing operations.

Chapter- IV

Layout Generation

Layout generation is the last step before sending the chip to fabrication. It takes the structural netlist from the previous step and generates the physical layout. The layout generation flow depicted in Figure 3. The final goal is to create the GDSII file . The next steps are comprised in the layout generation:

- *Floorplanning*: It is being more and more common to perform and initial manual floorplanning before the automatic placement. In this way, some hierarchy is given to the design in order to avoid placing a “flat” design. The benefit of this approach is to get closer modules that has to communicate with each other with the purpose of minimize the wire length.
- *Placement*: Where to place the standard cells is the first task that needs to be solved in the Layout Generation. The simple idea is to minimize the length of wires, but, for example, in a timing-driven placement the intention is to decrease as much as possible the delay on the critical paths.
- *Routing*: At this point, the cells are placed and they need to be interconnected. The routing step can be divided in two stages: global routing and detailed routing. In the first stage (global routing) the problem is abstracted to establish through which channels the connections will flow. Then, in the second stage (detailed routing) the exact position of a connection wire within a channel is determined.
- *Timing Analysis*: This is the critical step of the design flow and it can be seen as a bottleneck in the process. The static timing analysis is rerun after the place and route in order to check if the timing requirements are accomplished. It can be necessary some iterations of synthesis and P&R (place and routing) before our goal is reached. This iteration process is called back-annotated synthesis.
- *Clock tree generation*: How the clock is distributed in a design is one of the most

limiting factors in order to achieve high frequency circuits. The clock signal has to get all the clocked components at the same time in order to achieve a correct operation, and the more the frequency is increased, the more clock inaccuracy we have. The clock inaccuracy consists of two elements: the clock skew and the clock jitter. In addition to this, it is also important to take care about power consumption reduction using clock gating. This factor is extremely important because, for example in Intel chips, the clock network can be even 50% of the total power consumption[1].

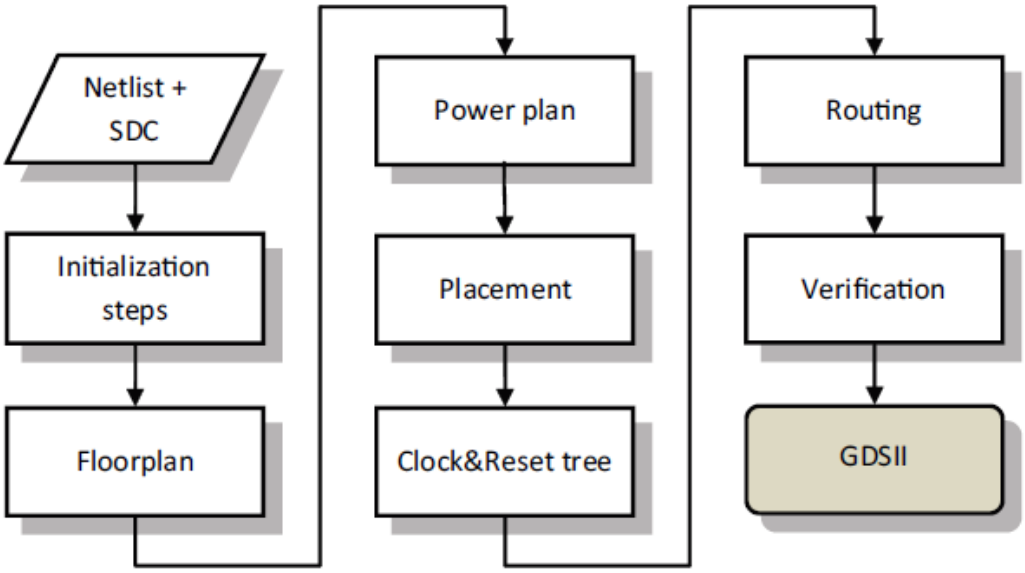


Figure 3: Layout generation flow

Chapter- V

Place and Routing (PnR)

Place and routing are part of layout generation process. Place and Routing operation was performed by Cadence Encounter tool [5]. The inputs files were:

- testpro.v
- testpro_synth.v
- testpro.sdc

The testpro_synth.v and testpro.sdc file was the output of Gate synthesis which was previously done by Cadence Genus tool. Also, for time timing analysis previously used liberty timing files (lef) or timing libraries were taken as input. Those files were:

- slow.lib
- fast.lib
- typical.lib

As technology file gsclib045.lef was taken. All these files were copied into the Encounter's working directory. These copy operations were done by bash commands or using nautilus (which provides the gui of linux).

For opening Encounter in the project working directory, we followed the instructions of lab manual [6].

From the file section by selecting import design option, all design was imported, such as technology files, library files, power rails and etc.

After that analysis configuration was created and saved. Floor plan was specified right after that process. Here, for my specifications dimensions was 71.99x70.11 (width x height). Core to IO boundary was selected. Figure 4 shows the floor plan specification information of the project. Placement blockage was placed after completing floorplanning (Figure 5).

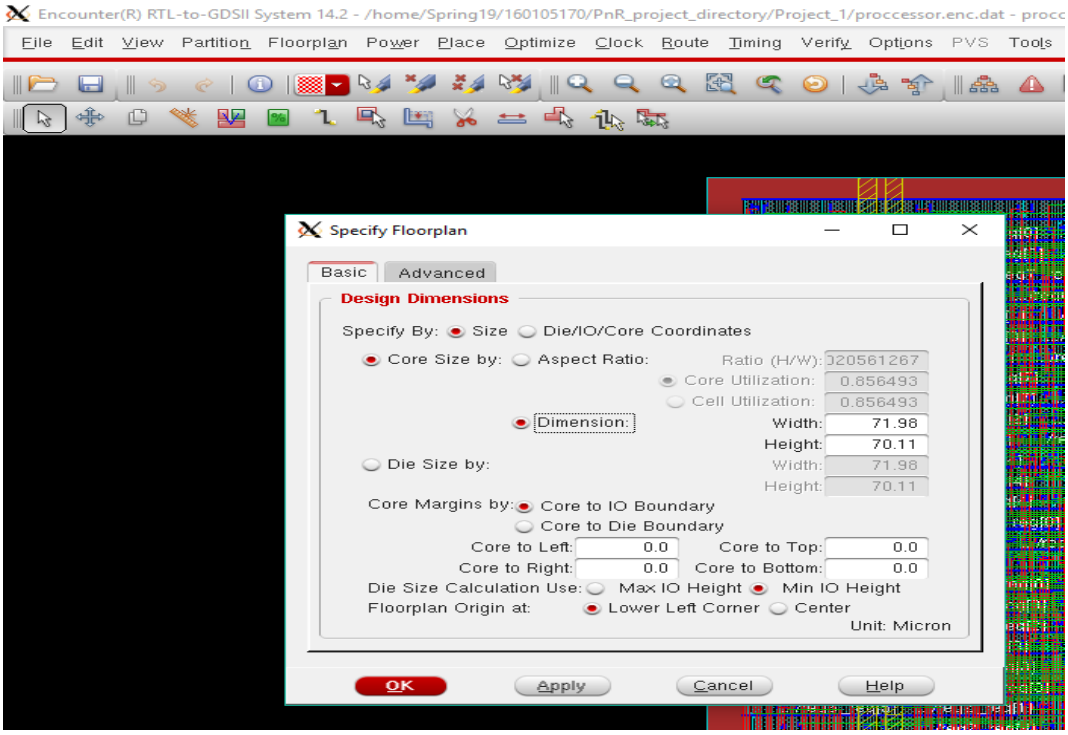


Figure 4: Floorplan specification

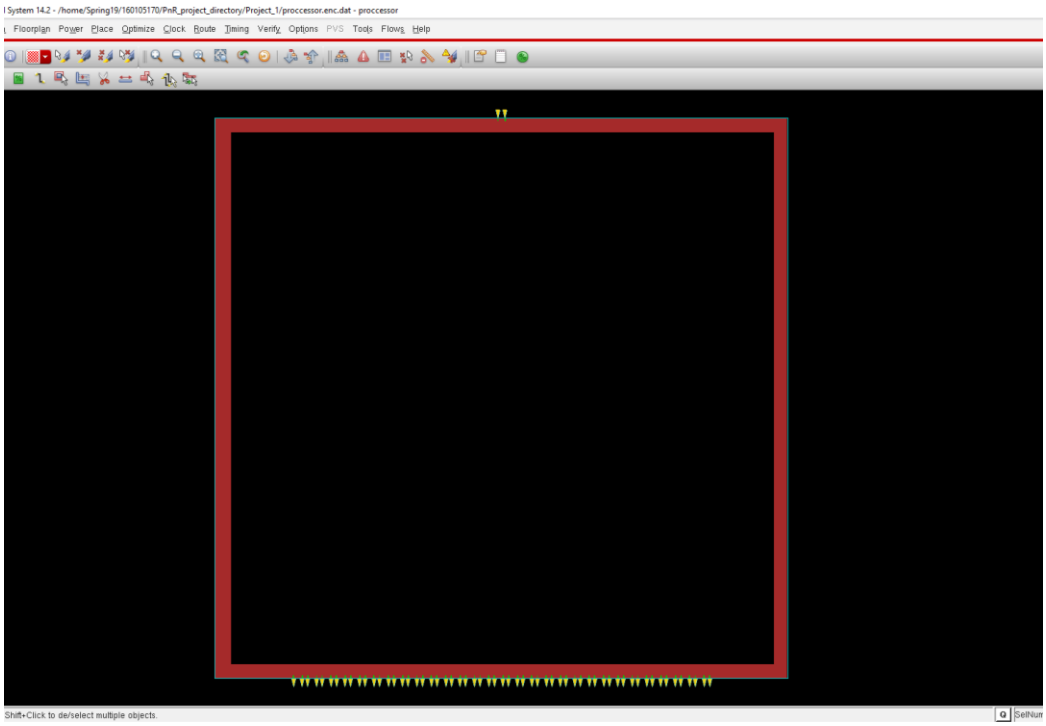


Figure 5: Placement of blockage

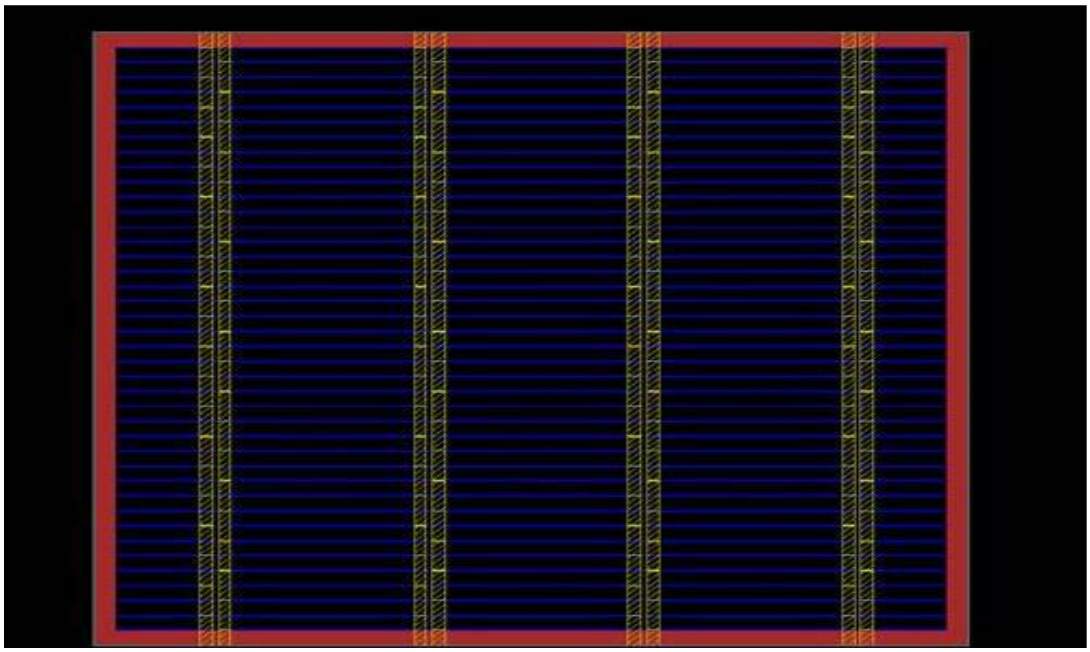


Figure 6: Adding power strips with metal 4

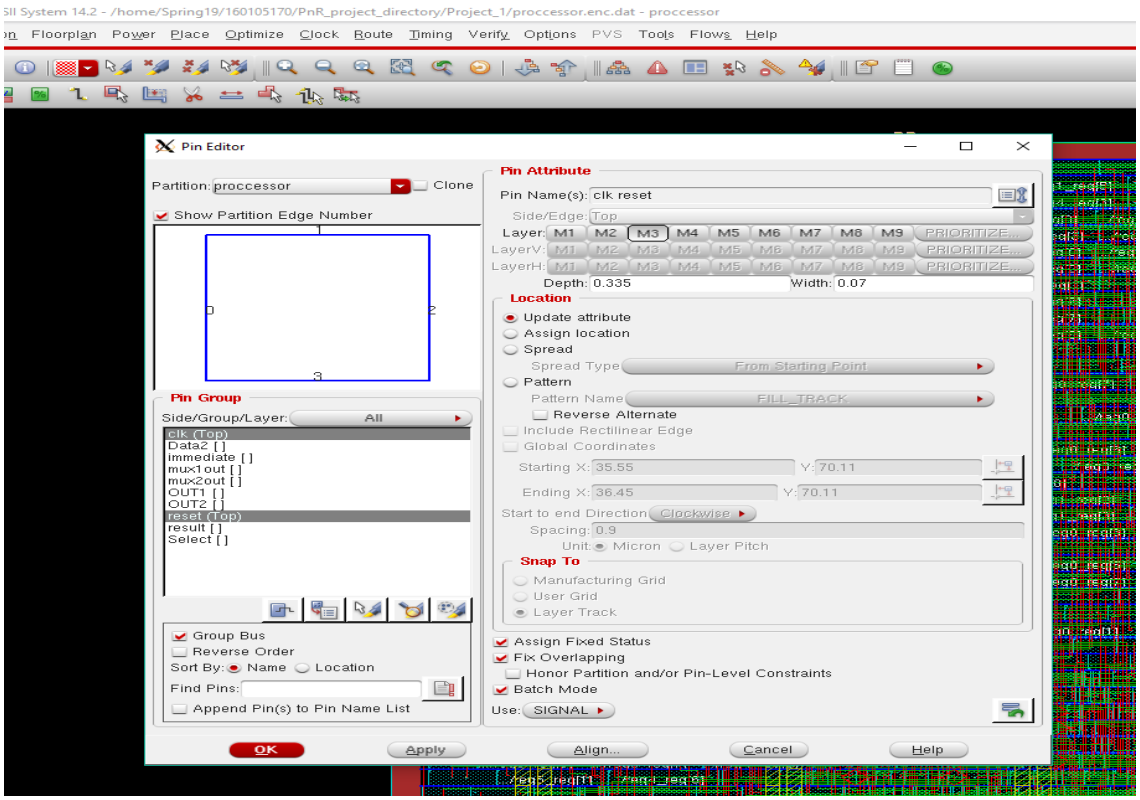


Figure 7: Input pin specification with metal 3 and placing them on the top as per specifications

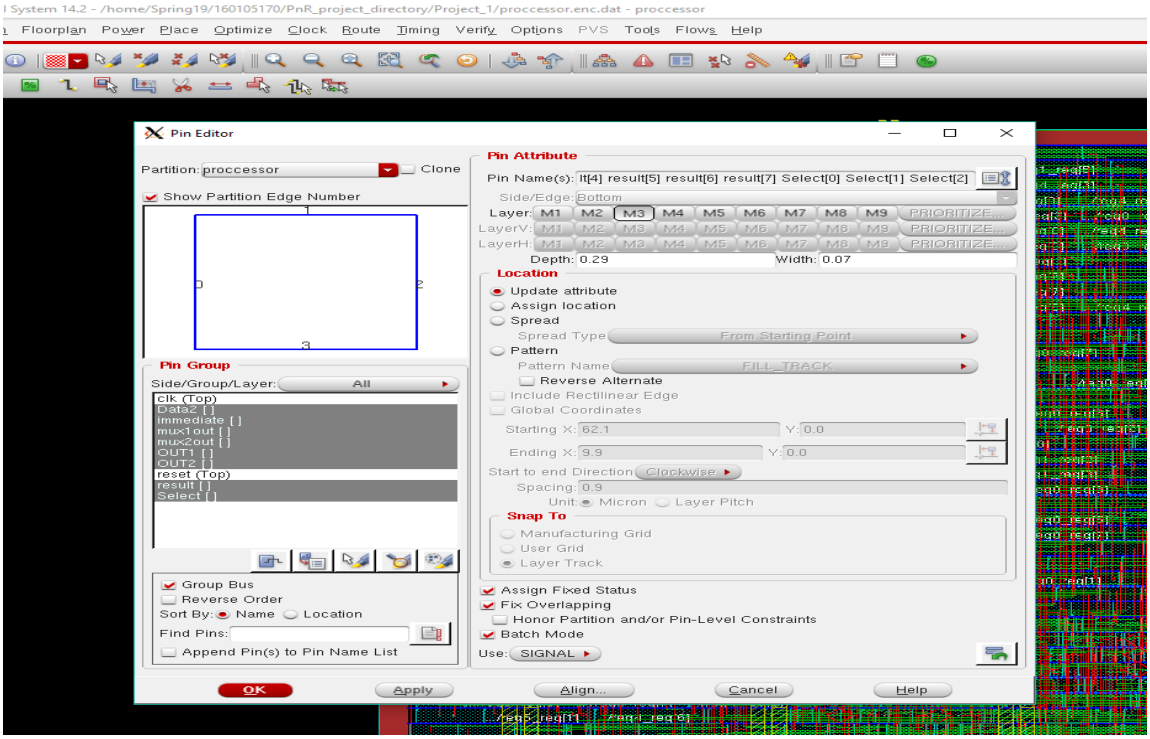


Figure 8: Output pin specification with metal 3 and placing them on the bottom as per specifications

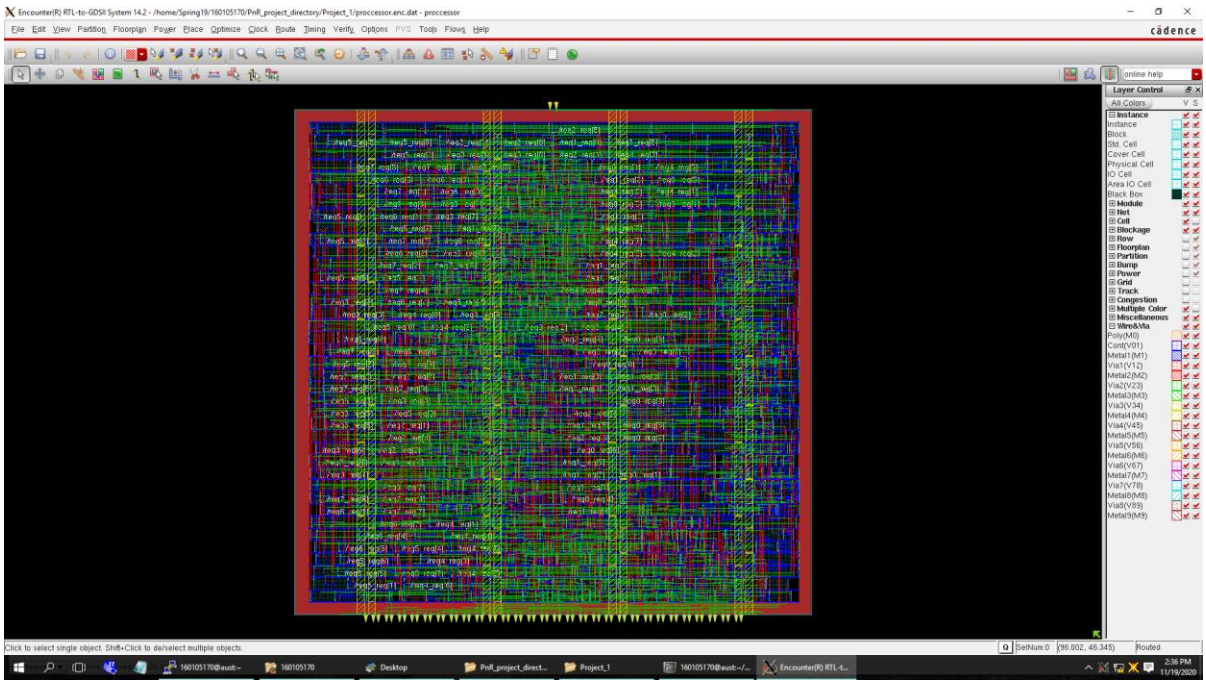


Figure 9: Layout after place and routing

```
******** Start: VERIFY CONNECTIVITY ********
Start Time: Thu Nov 19 12:15:27 2020

Design Name: proccessor
Database Units: 2000
Design Boundary: (0.0000, 0.0000) (71.9800, 70.1100)
Error Limit = 1000; Warning Limit = 50
Check all nets

Begin Summary
  Found no problems or warnings.
End Summary

End Time: Thu Nov 19 12:15:27 2020
Time Elapsed: 0:00:00.0

******** End: VERIFY CONNECTIVITY ********
  Verification Complete : 0 Viols.   0 Wrngs.
  (CPU Time: 0:00:00.1  MEM: 0.000M)

encounter 4>   *** Starting Verify DRC (MEM: 986.7) ***

  VERIFY DRC ..... Starting Verification
  VERIFY DRC ..... Initializing
  VERIFY DRC ..... Deleting Existing Violations
  VERIFY DRC ..... Creating Sub-Areas
  VERIFY DRC ..... Using new threading
  VERIFY DRC ..... Sub-Area : 1 of 1
  VERIFY DRC ..... Sub-Area : 1 complete 0 Viols.

  Verification Complete : 0 Viols.

 *** End Verify DRC (CPU: 0:00:00.3  ELAPSED TIME: 0.00  MEM: 138.5M) ***

 *** Starting Verify Geometry (MEM: 892.2) ***

  VERIFY GEOMETRY ..... Starting Verification
  VERIFY GEOMETRY ..... Initializing
  VERIFY GEOMETRY ..... Deleting Existing Violations
  VERIFY GEOMETRY ..... Creating Sub-Areas
                  ..... bin size: 1920
  VERIFY GEOMETRY ..... SubArea : 1 of 1
  VERIFY GEOMETRY ..... Cells             :   0 Viols.
  VERIFY GEOMETRY ..... SameNet           :   0 Viols.
  VERIFY GEOMETRY ..... Wiring            :   0 Viols.
  VERIFY GEOMETRY ..... Antenna           :   0 Viols.
  VERIFY GEOMETRY ..... Sub-Area : 1 complete 0 Viols.  0 Wrngs.
VG: elapsed time: 1.00
Begin Summary ...
  Cells       : 0
  SameNet     : 0
  Wiring      : 0
  Antenna     : 0
  Short       : 0
  Overlap     : 0
End Summary

  Verification Complete : 0 Viols.   0 Wrngs.

**********End: VERIFY GEOMETRY**********
 *** verify geometry (CPU: 0:00:00.5  MEM: 20.0M)
```

Figure 10: Verification of connectivity, DRC, geometry.

Using metal 4 power strips were placed within specified areas which is shown in Figure 6. On the specifications it was mentioned to set the input and output pins on top and bottom respectively. Metal 3 was used as the pin layer, Figure 7 and Figure 8 show the information of input and output pin placement. After completing all other mandatory instructions mentioned in lab manual final layout popped up (Figure 9). All connectivity, DRC (Design rule check), and geometry was verified and Figure 10 shows that all violation was cleared.

Chapter- VI

Static Timing Analysis

Static Timing Analysis (STA) is one of the techniques to verify design in terms of timing. This kind of analysis doesn't depend on any data or logic inputs, applied at the input pins.

The input to an STA tool is the routed netlist, clock definitions (or clock frequency) and external environment definitions. The STA will validate whether the design could operate at the rated clock frequency, without any timing violations.

Some of the basic timing violations are **setup violation** and **hold violation**.

The delay between Sender and Receiver is not constant, but will have range of values, as shown below (Figure 11).

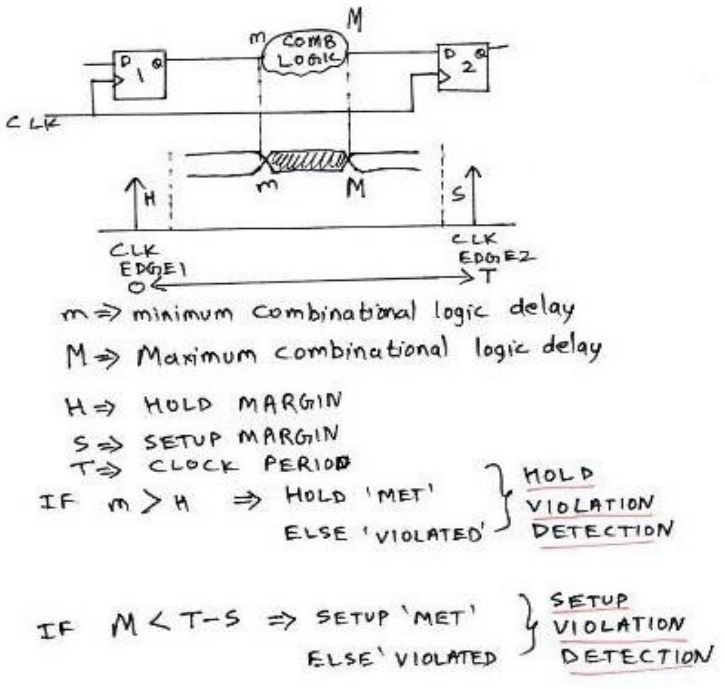


Figure 11: Setup and hold time violation

Therefore, the delay ranges from m ≤ delay ≤ M.

Where m = minimum propagation delay of combinational logic and M = Maximum propagation delay of combinational logic

The finite time periods ‘t1’ and ‘t2’ are the internal delays of a flip-flop

The data is not expected to change between hold time ‘H’ to ‘m’ and ‘M’ to (Tclk – Setup time ‘S’). Data changes somewhere between ‘m’ and ‘M’, and becomes stable after that.

Therefore, we have got certain defining equations for setup and hold time, and they are as follows:

m > H i.e. Minimum propagation delay of the combinational logic should be greater than Hold Margin.

If m < H , it results into timing violation, called as Hold violation. This means, that the combinational logic delay is very less and hence data change is very fast. To satisfy the ‘hold’ requirement, the combinational logic delay should be increased.

M < Tclk – S i.e. Maximum propagation delay of the combinational logic should be less than Clock period (Tclk) minus the Seup Margin

If M > Tclk – S , it results into timing violation, called as Setup violation. This means, that the combinational logic delay is very large and hence data change is very slow. To satisfy the ‘setup’ requirement, the combinational logic delay should be decreased.

The process of fixing timing violation, and implement the fixes back to the PNR netlist, is referrred to as *Engineering Change order (ECO).*

Workflow of static timing analysis (STA) has given below in Figure 12.

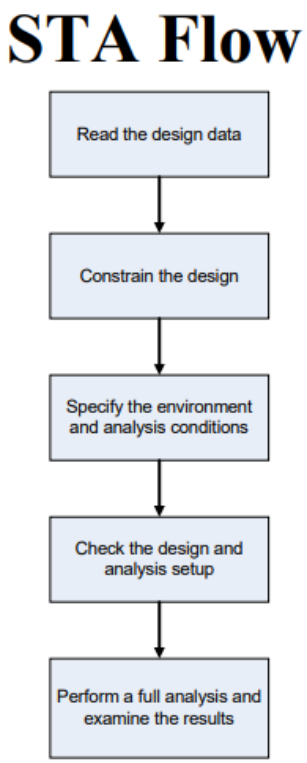


Figure 12: Static timing analysis (STA) flow

In STA the following commands were used:

- To find the summary of timing report Design:
  - To check No Load timing:

    timeDesign –prePlace
  - Check after place all the cells

    timeDesign -preCTS
  - Check after clock tree build (CTS)

    timeDesign -postCTS

    timeDesign -postCTS -hold
  - Check after complete the routing

    timeDesign -postRoute

    timeDesign -postRoute -hold

- Set max fanout fanout
  set_interactive_constraint_modes [all_constraint_modes -active]
  set_max_fanout 10 [current_design]
- Check fanout violation
  report_constraint -drv_violation_type max_fanout
- Set max transtion
  set_max_transition 2 [current_design]
- Check transition violations
  report_constraint -drv_violation_type max_transition
  setAnalysisMode -CheckType hold
  timeDesign -postRoute –hold

There were 0 setup violation on the first report of the STA but almost 188 hold violations were found. ecoPlace and ecoRoute is great command to reduce the hold violation but there are other commands that were used to reduce the hold violations on the design.

Following commands was used to reduce violations.

#remove filler cells (to add cells to remove violations by upsizing)

deleteFiller -cell *

#optimize the design optDesign –postRoute/-postCTS –hold

#check violations in netlist again

timeDesign -postRoute report_constraint

drv_violation_type max_transition (new report is updated)

ecoPlace

ecoRoute

After applying optimization commands hold violations went down to 73 (Figure 13). These could be minimized using this manual iterative method.

Setup mode	all	reg2reg	default
WNS (ns):	0.051	0.156	0.051
TNS (ns):	0.000	0.000	0.000
Violating Paths:	0	0	0
All Paths:	281	281	3

Hold mode	all	reg2reg	default
WNS (ns):	-0.173	-0.173	N/A
TNS (ns):	-7.601	-7.601	N/A
Violating Paths:	73	73	N/A
All Paths:	281	281	N/A

DRVs	Real		Total
	Nr nets(terms)	Worst Vio	Nr nets(terms)
max_cap	0 (0)	0.000	0 (0)
max_tran	0 (0)	0.000	1 (1)
max_fanout	19 (19)	-12	20 (20)
max_length	0 (0)	0	0 (0)

Figure 13: Setup and Hold time violation

Buffer Tree Synthesis

After cleaning the violation CLK using BTS, buffer tree synthesis was preformed using following command:

bufferTreeSynthesis -nets clk -maxFanout 10

Buffer tree synthesis reduce the maximum fanout at certain level(Figure 14).

Check type : max_fanout

Pin Name	Required	Actual	Slack	View
myreg/g6980/Y	10.000	22.000	-12.000	func_slow
myreg/g6978/Y	10.000	22.000	-12.000	func_slow
myreg/g6979/Y	10.000	22.000	-12.000	func_slow
myreg/g2/Y	10.000	22.000	-12.000	func_slow
myreg/g6976/Y	10.000	22.000	-12.000	func_slow
myreg/g6977/Y	10.000	22.000	-12.000	func_slow
myreg/FE_DBTC0_n_31/Y	10.000	22.000	-12.000	func_slow
myreg/g6974/Y	10.000	22.000	-12.000	func_slow
myreg/g6800/Y	10.000	17.000	-7.000	func_slow
FE_OFC16_result_4_/Y	10.000	16.000	-6.000	func_slow
FE_OFC4_result_7_/Y	10.000	16.000	-6.000	func_slow
FE_OFC26_result_1_/Y	10.000	16.000	-6.000	func_slow
FE_PHC24_INaddr_0_/Y	10.000	16.000	-6.000	func_slow
myreg/instruction_reg[10]/Q	10.000	16.000	-6.000	func_slow
FE_OFC12_result_5_/Y	10.000	15.000	-5.000	func_slow
FE_OFC20_result_3_/Y	10.000	15.000	-5.000	func_slow
myreg/g6785/Y	10.000	13.000	-3.000	func_slow
clk__L3_I0/Y	10.000	12.000	-2.000	func_slow
clk__L3_I1/Y	10.000	12.000	-2.000	func_slow
FE_OFC8_result_6_/Y	10.000	12.000	-2.000	func_slow
clk__L5_I0/Y	10.000	11.000	-1.000	func_slow
clk__L5_I2/Y	10.000	11.000	-1.000	func_slow
clk__L4_I3/Y	10.000	11.000	-1.000	func_slow

Figure 14: Maximum fanout of pins after buffer clock synthesis

Verifying connectivity, DRC and geometry after all these operations. All the violation was cleared (Figure 15)

```
******** Start: VERIFY CONNECTIVITY ********
Start Time: Thu Nov 19 12:15:27 2020

Design Name: proccessor
Database Units: 2000
Design Boundary: (0.0000, 0.0000) (71.9800, 70.1100)
Error Limit = 1000; Warning Limit = 50
Check all nets

Begin Summary
  Found no problems or warnings.
End Summary

End Time: Thu Nov 19 12:15:27 2020
Time Elapsed: 0:00:00.0

******** End: VERIFY CONNECTIVITY ********
  Verification Complete : 0 Viols.  0 Wrngs.
  (CPU Time: 0:00:00.1  MEM: 0.000M)
```

```
encounter 4>  *** Starting Verify DRC (MEM: 986.7) ***

  VERIFY DRC ..... Starting Verification
  VERIFY DRC ..... Initializing
  VERIFY DRC ..... Deleting Existing Violations
  VERIFY DRC ..... Creating Sub-Areas
  VERIFY DRC ..... Using new threading
  VERIFY DRC ..... Sub-Area : 1 of 1
  VERIFY DRC ..... Sub-Area : 1 complete 0 Viols.

  Verification Complete : 0 Viols.

 *** End Verify DRC (CPU: 0:00:00.3  ELAPSED TIME: 0.00  MEM: 138.5M)


 *** Starting Verify Geometry (MEM: 892.2) ***

  VERIFY GEOMETRY ..... Starting Verification
  VERIFY GEOMETRY ..... Initializing
  VERIFY GEOMETRY ..... Deleting Existing Violations
  VERIFY GEOMETRY ..... Creating Sub-Areas
                  ..... bin size: 1920
  VERIFY GEOMETRY ..... SubArea : 1 of 1
  VERIFY GEOMETRY ..... Cells          :  0 Viols.
  VERIFY GEOMETRY ..... SameNet        :  0 Viols.
  VERIFY GEOMETRY ..... Wiring         :  0 Viols.
  VERIFY GEOMETRY ..... Antenna        :  0 Viols.
  VERIFY GEOMETRY ..... Sub-Area : 1 complete 0 Viols. 0 Wrngs.
VG: elapsed time: 1.00
Begin Summary ...
  Cells       : 0
  SameNet     : 0
  Wiring      : 0
  Antenna     : 0
  Short       : 0
  Overlap     : 0
End Summary

  Verification Complete : 0 Viols.  0 Wrngs.

**********End: VERIFY GEOMETRY**********
 *** verify geometry (CPU: 0:00:00.5  MEM: 20.0M)
```

Figure 15: Verification after STA

Chapter VII

Final Output

The snapshot of final layout of the design has given on Figure 16.

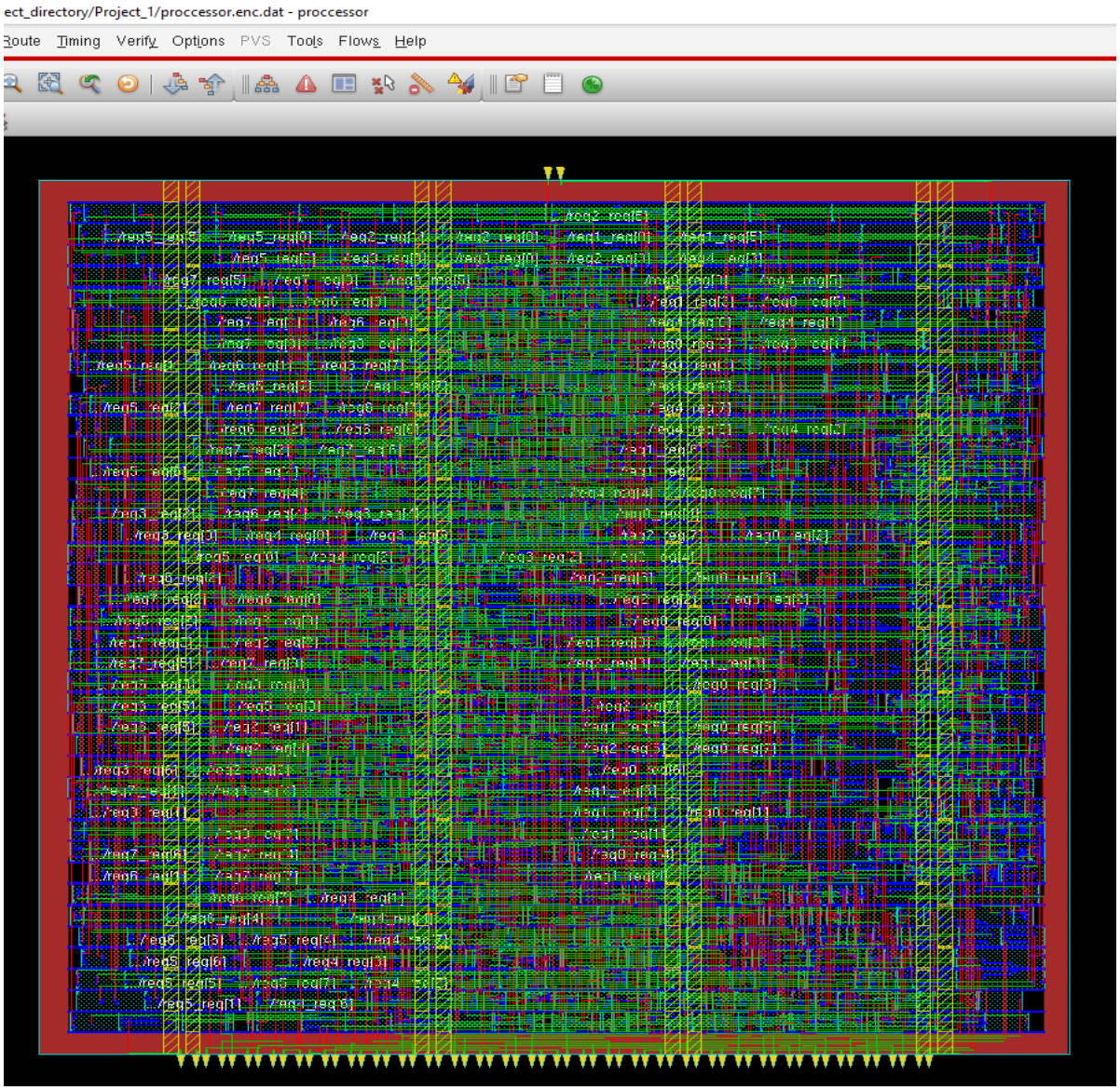


Figure 16: Final Layout of the design

Chapter VIII

Discussions

The aim of this project was to complete whole VLSI design flow with given specification using Cadence genus and encounter tool. All connectivity, DRC and geometry were checked and fixed during the project. In the static timing analysis Setup and hold violations were checked and all the setup violations were fixed successfully. Almost 188 hold violations occurred in the first place of the STA check and using *ecoplace*, *ecoroute* and other optimizing commands hold violations were reduced to 73. To cleared it to zero violations a manual iterations method can be applied. Also, at the end of the project maximum fan out was reduced by means of Buffer tree synthesis.

References

[1] Moar Gomez, J. (2009). 90 nm VLSI Design of an 8-bits Microcontroller.

[2] Singh, D., & Chandel, R. (2020). Register-Transfer-Level Design for Application-Specific Integrated Circuits. In Nanoscale VLSI (pp. 295-319). Springer, Singapore.

[3] Command Reference for Encounter RTL Compiler Product Version 9.1 July 2009.

[4] Command Reference for BuildGates Synthesis and Cadence PKS Product Version 5.0.13 December 2003

[5] Cadence (2017) Innovus user guide

[6] VLSI II lab manual, Department of Electrical and Electronic Engineering, Ahsanullah University of Science and Technology.