ESD 902: Reading Elective

October 25, 2018

A REPORT ON

Tool for Digital IC Design Flow Automation

Guide:
Prof. Nanditha Rao

Report by:
Amey Kulkarni (MT2017501)

Contents

List of Figures

1 Introduction

The VLSI design cycle starts with a formal specification of a VLSI chip, follows a series of steps, and eventually produces a packaged chip. A typical design cycle may be represented by the flow chart shown in figure 1. Our emphasis is on the physical design step of the VLSI design cycle. However, to gain a global perspective, we briefly outline all the steps of the VLSI design cycle.

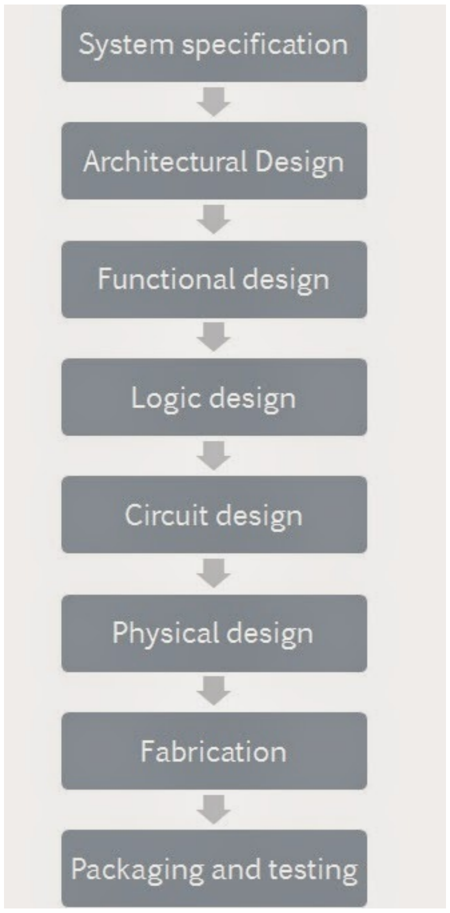


Figure 1: VLSI Design Flow

1.1 System Specification

The specification for an IC to be designed may come from a customer A who has requested a company B to design the IC for them. Else the company B may come up with its own specification for their own chip which will be marketed and sold later. The specification usually talks about the detailed functional requirements from the IC, the minimum operating speed, the maximum power and area for the IC and other robustness and reliability requirements.

1.2 Architectural Design

The basic architecture of the system is designed in this step. This includes, such decisions as RISC (Reduced Instruction Set Computer) versus CISC (Complex Instruction Set Computer), number of ALUs, Floating Point units, number and structure of pipelines, and size of caches among others.
The outcome of architectural design is a Micro-Architectural Specification (MAS). While MAS is a textual (English like) description, architects can accurately predict the performance, power and die size of the design based on such a description.

1.3 Functional Design

In this step, main functional units of the system are identified. This also identifies the interconnect requirements between the units. The area, power, and other parameters of each unit are estimated. The behavioral aspects of the system are considered without implementation specific information. For example, it may specify that a multiplication is required, but exactly in which mode such multiplication may be executed is not specified. We may use a variety of multiplication hardware depending on the speed and word size requirements. The key idea is to specify behavior, in terms of input, output and timing of each unit, without specifying its internal structure.
The outcome of functional design is usually a timing diagram or other relationships between units. This information leads to improvement of the overall design process and reduction of the complexity of subsequent phases. Functional or behavioral design provides quick emulation of the system and allows fast debugging of the full system. Behavioral design is largely a manual step with little or no automation help available.

1.4 Logic Design

Now the RTL Design team comes into the picture and the partitioned designed is given to them. The sub-blocks are designed by RTL designers by using a HDL such as Verilog/SystemVerilog. The whole design is modeled at a higher level of abstraction using mixed style of modelling (behavioral, data-flow and structural).
The functional verification happens in parallel with the RTL design where the verification team develops the verification environment using a HVL such as SystemVerilog. In simple words, the code to check whether the RTL Design code does its functionality correctly is developed by the verification engineers. Once the RTL design is complete, the design is verified for correct functionality by functional verification. In case of any errors, the design code is corrected and re-verified.

1.5 Circuit Design

The purpose of circuit design is to develop a circuit representation based on the logic design. The Boolean expressions are converted into a circuit representation by taking into consideration the speed and power requirements of the original design. Circuit Simulation is used to verify the correctness and timing of each component.
The circuit design is usually expressed in a detailed circuit diagram. This diagram shows the circuit elements (cells, macros, gates, transistors) and interconnection between these elements. This representation is also called a netlist. Tools used to manually enter such description are called schematic capture tools. In many cases, a netlist can be created automatically from logic (RTL)

description by using logic synthesis tools.

1.6 Physical Design

In this step the circuit representation (or netlist) is converted into a geometric representation. As stated earlier, this geometric representation of a circuit is called a layout. Layout is created by converting each logic component (cells, macros, gates, transistors) into a geometric representation (specific shapes in multiple layers), which perform the intended logic function of the corresponding component. Connections between different components are also expressed as geometric patterns typically lines in multiple layers.
The exact details of the layout also depend on design rules, which are guidelines based on the limitations of the fabrication process and the electrical properties of the fabrication materials. Physical design is a very complex process and therefore it is usually broken down into various sub-steps. Various verification and validation checks are performed on the layout during physical design.

1.7 Fabrication

After layout and verification, the design is ready for fabrication. Since layout data is typically sent to fabrication on a tape, the event of release of data is called Tape Out. Layout data is converted (or fractured) into photo-lithographic masks, one for each layer. Masks identify spaces on the wafer, where certain materials need to be deposited, diffused or even removed. Silicon crystals are grown and sliced to produce wafers. Extremely small dimensions of VLSI devices require that the wafers be polished to near perfection. The fabrication process consists of several steps involving deposition, and diffusion of various materials on the wafer. During each step one mask is used. Several dozen masks may be used to complete the fabrication process.
A large wafer is 20 cm (8 inch) in diameter and can be used to produce hundreds of chips, depending of the size of the chip. Before the chip is mass produced, a prototype is made and tested. Industry is rapidly moving towards a 30 cm (12 inch) wafer allowing even more chips per wafer leading to lower cost per chip.

1.8 Packaging and testing

Finally, the wafer is fabricated and diced into individual chips in a fabrication facility. Each chip is then packaged and tested to ensure that it meets all the design specifications and that it functions properly. Chips used in Printed Circuit Boards (PCBs) are packaged in Dual In-line Package (DIP), Pin Grid Array (PGA), Ball Grid Array (BGA), and Quad Flat Package (QFP). Chips used in Multi-Chip Modules (MCM) are not packaged, since MCMs use bare or naked chips.

2 Tool Flow

The flow of the tool is such that when a Verilog Code along with constraints is given at the input of the tool, a Layout of the Verilog code is generated along with the corresponding reports generated at each and every stage of the flow. The Flow is as shown in Figure 2.

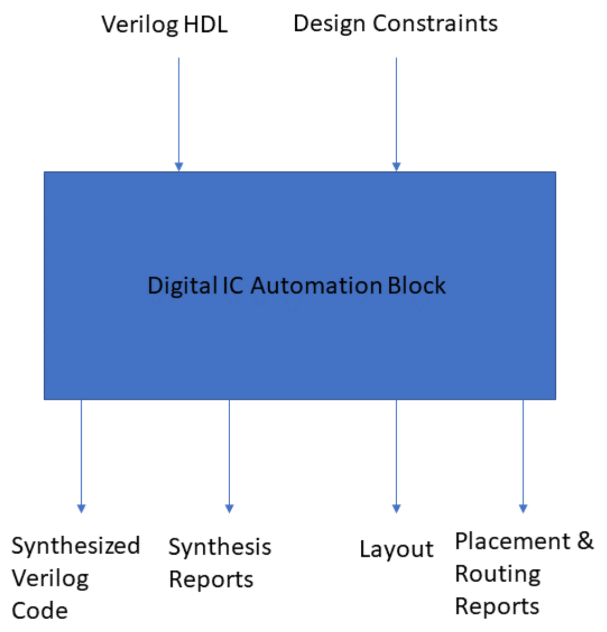


Figure 2: Digital IC Automation Tool Block

Note that in the above block diagram, all the inputs to the tool are not shown. Apart from the inputs shown, all other inputs are fixed for a particular technology node and the inputs are as shown in figure 3.
As shown in figure 3, the tool can be invoked either by command line or by using GUI. The command line script for the tool has been designed using TCL Shell and the GUI has been designed using TKinter library in Python 2. These scripts control two tools in the Cadence Design Suite namely Genus and Innovus. The tools used are as shown in the figure 4.

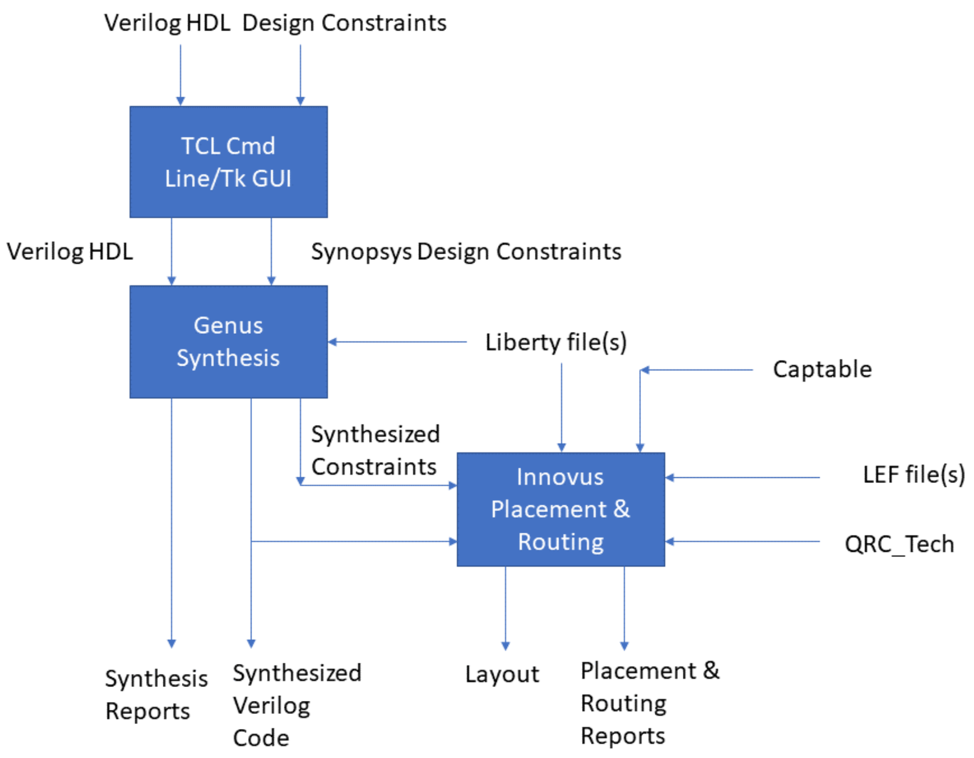


Figure 3: Digital IC Automation Tool Complete Block

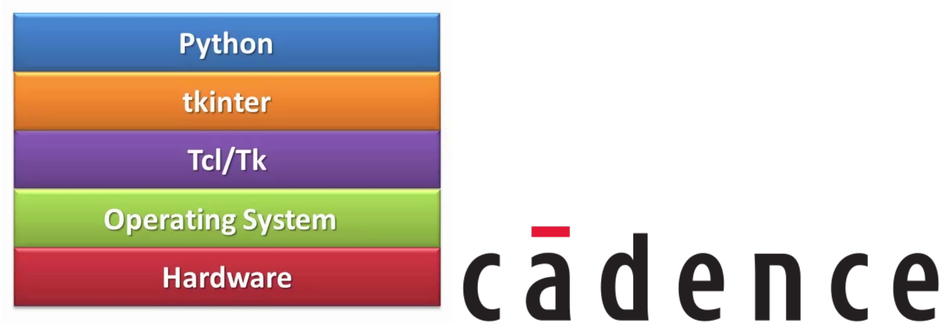
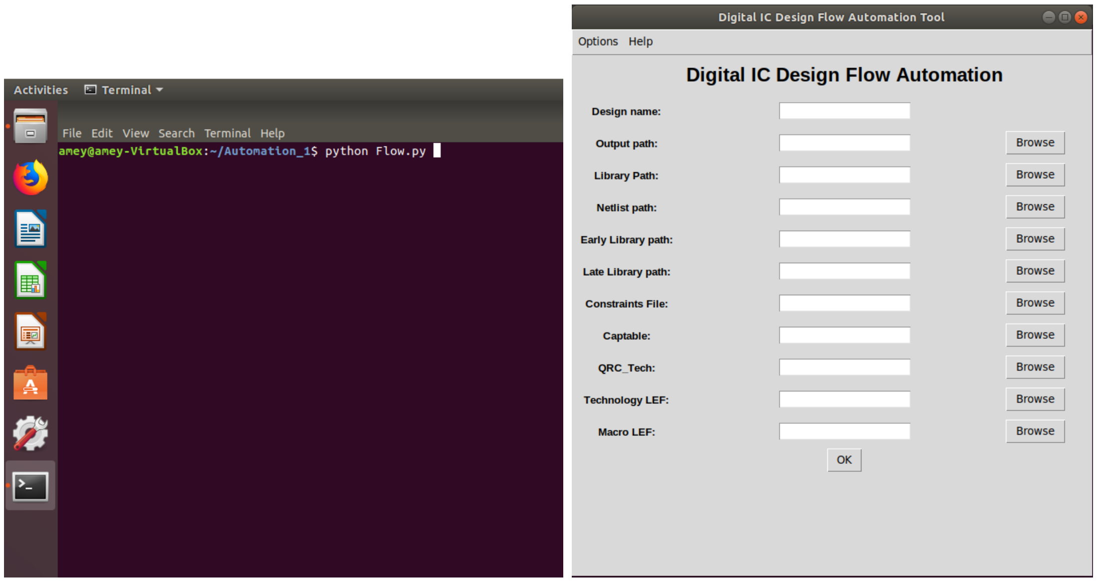


Figure 4: Software Used

3 Tool Usage

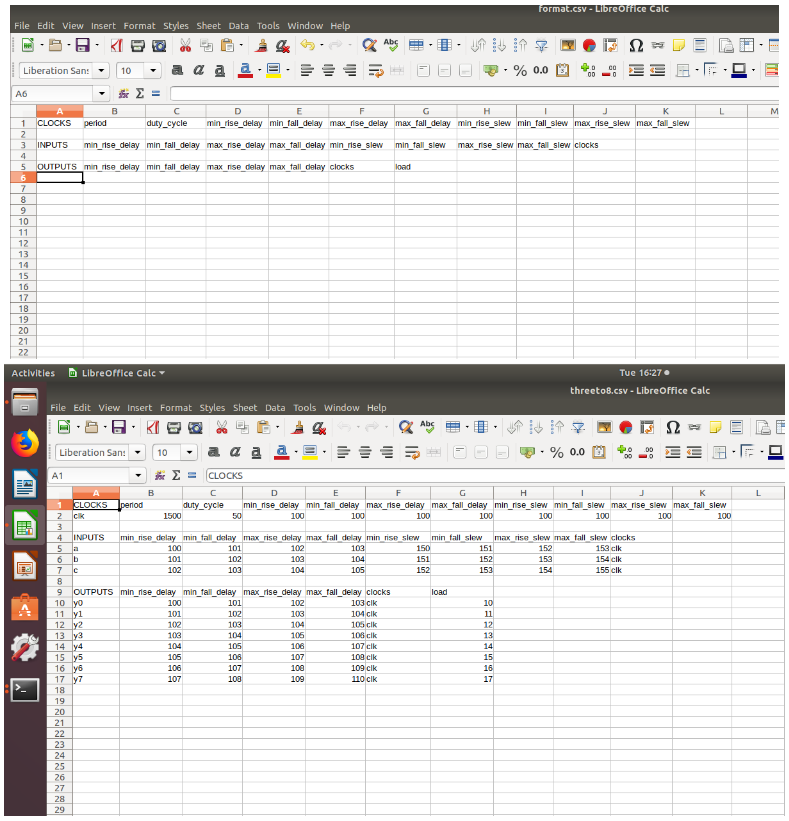
The Tool usage has been tested for the design of a three to eight decoder and the commands given in this section is for the same design. To run the tool for any other design, the verilog and constraints files need to be changed and every other dependency remains the same. To Invoke the GUI, the command is **user@user: python Flow.py**. The image for the same is as shown below in figure 5.



```
amey@amey-VirtualBox:~/Automation_1$ python Flow.py
```

Figure 5: Command to invoke GUI

After invoking the GUI, all the files mentioned in the GUI need to be specified. The **design name** is the name of the **toplevel module**, the output directory is any empty directory where output needs to be saved. The **Library path** is the directory where all the **liberty files(.lib)** are saved. The **Netlist path** is the directory where the **Verilog files(.v)** are saved. The Early Library path and Late Library Path fields are to be filled by the liberty files that are to be used for setup and hold analysis corners. The **Constraints file** is a **'.csv' file** which contains details about the clocks to the design, inputs and outputs of the design as required. This file is then converted into a **'.sdc' file** which is then used for synthesis. The format of the file is a shown below in figure 6.
The Captable filed should contain a file of extension **'.capTbl'**, the QRC_Tech field should contain a file of extension **'.tch'** and the Technology and Macro LEF fields should contain files of extension **'.lef'**. After all the fields are filled, by clicking on the OK button, the tools get launched and the corresponding messages are as shown on the terminal. The GUI with all fields filled should appear as shown in fgure 7.
The tool can also be launched from command line by invoking the command
user@user: tclsh synthesis_pnr.tcl designname_details.csv where the designname_details.csv file should have the format as shown in figure 8.
When the tool is being invoked, the following files are required to be in the same directory:


format.csv - LibreOffice Calc
File Edit View Insert Format Styles Sheet Data Tools Window Help
A6
threeto8.csv - LibreOffice Calc
Activities
LibreOffice Calc
Tue 16:27
A1
CLOCKS

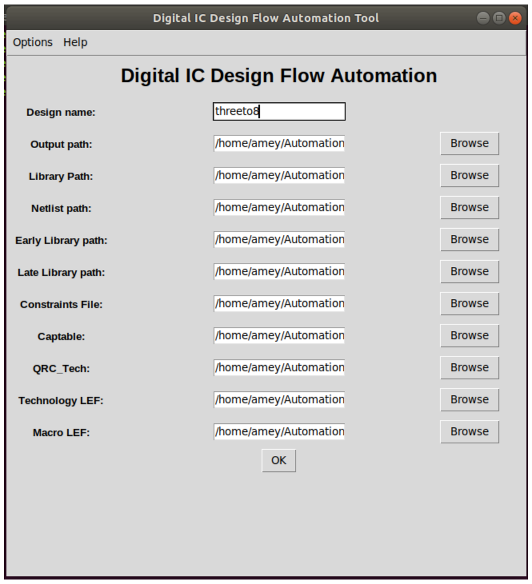
	A	B	C	D	E	F	G	H	I	J	K
1	CLOCKS	period	duty_cycle	min_rise_delay	min_fall_delay	max_rise_delay	max_fall_delay	min_rise_slew	min_fall_slew	max_rise_slew	max_fall_slew
2											
3	INPUTS	min_rise_delay	min_fall_delay	max_rise_delay	max_fall_delay	min_rise_slew	min_fall_slew	max_rise_slew	max_fall_slew	clocks	
4											
5	OUTPUTS	min_rise_delay	min_fall_delay	max_rise_delay	max_fall_delay	clocks	load				

	A	B	C	D	E	F	G	H	I	J	K
1	CLOCKS	period	duty_cycle	min_rise_delay	min_fall_delay	max_rise_delay	max_fall_delay	min_rise_slew	min_fall_slew	max_rise_slew	max_fall_slew
2	clk	1500	50	100	100	100	100	100	100	100	100
3											
4	INPUTS	min_rise_delay	min_fall_delay	max_rise_delay	max_fall_delay	min_rise_slew	min_fall_slew	max_rise_slew	max_fall_slew	clocks	
5	a	100	101	102	103	150	151	152	153	clk	
6	b	101	102	103	104	151	152	153	154	clk	
7	c	102	103	104	105	152	153	154	155	clk	
8											
9	OUTPUTS	min_rise_delay	min_fall_delay	max_rise_delay	max_fall_delay	clocks	load				
10	y0	100	101	102	103	clk	10				
11	y1	101	102	103	104	clk	11				
12	y2	102	103	104	105	clk	12				
13	y3	103	104	105	106	clk	13				
14	y4	104	105	106	107	clk	14				
15	y5	105	106	107	108	clk	15				
16	y6	106	107	108	109	clk	16				
17	y7	107	108	109	110	clk	17				

Figure 6: Constraints File

1. **Flow.py**
2. **cad.csh**
3. **synthesis_pnr.tcl**
4. Cadence environment variable configuration file**(cshrc_hub)**

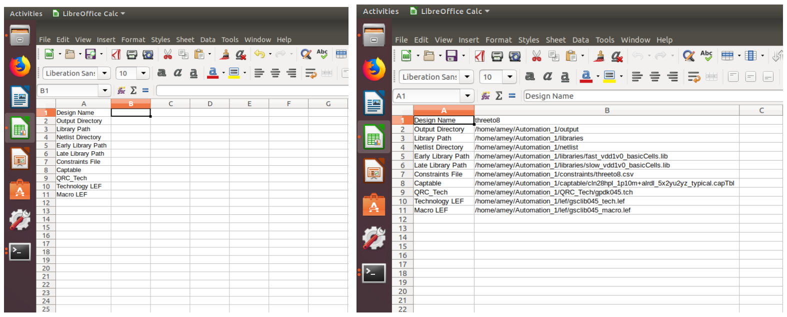
For running the code using command line, the Flow.py file need not be in the same directory, but the designname_details.csv file needs to be manually typed in as shown in figure 8.



Digital IC Design Flow Automation

Design name:	threeto8	
Output path:	/home/amey/Automation	Browse
Library Path:	/home/amey/Automation	Browse
Netlist path:	/home/amey/Automation	Browse
Early Library path:	/home/amey/Automation	Browse
Late Library path:	/home/amey/Automation	Browse
Constraints File:	/home/amey/Automation	Browse
Captable:	/home/amey/Automation	Browse
QRC_Tech:	/home/amey/Automation	Browse
Technology LEF:	/home/amey/Automation	Browse
Macro LEF:	/home/amey/Automation	Browse

Figure 7: GUI with all fields filled

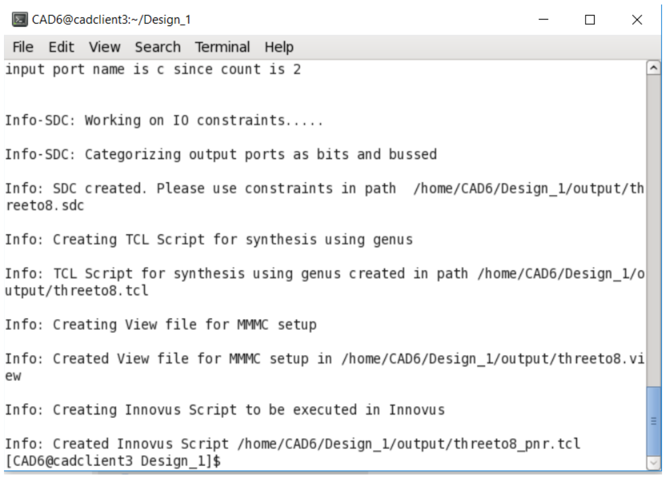

Activities
LibreOffice Calc
File Edit View Insert Format Styles Sheet Data Tools Window Help

	A	B
1	Design Name	threeto8
2	Output Directory	/home/amey/Automation_1/output
3	Library Path	/home/amey/Automation_1/libraries
4	Netlist Directory	/home/amey/Automation_1/netlist
5	Early Library Path	/home/amey/Automation_1/libraries/fast_vdd1v0_basicCells.lib
6	Late Library Path	/home/amey/Automation_1/libraries/slow_vdd1v0_basicCells.lib
7	Constraints File	/home/amey/Automation_1/constraints/threeto8.csv
8	Captable	/home/amey/Automation_1/captable/cln28hpl_1p10m+alrdl_5x2yu2yz_typical.capTbl
9	QRC_Tech	/home/amey/Automation_1/QRC_Tech/gpdk045.tch
10	Technology LEF	/home/amey/Automation_1/lef/gsclib045_tech.lef
11	Macro LEF	/home/amey/Automation_1/lef/gsclib045_macro.lef

Figure 8: Command Line Argument File Format

4 Results

For testing the proper working of the tool, a design for a three to eight decoder with enable signal given to clock has been used. The Constraints file is written in **'.csv'** format as shown in figure 6. Then the GUI is launched and all the fields are filled as shown in figure 7. If the usage is through command line, then an additional **'.csv'** file has to be written as shown in figure 8. When the tool is launched, the messages are displayed on the terminal as each stae is completed. After the tool is launched, the messages that are displayed as shown in figure 9.


CAD6@cadclient3:~/Design_1
File Edit View Search Terminal Help

```
input port name is c since count is 2


Info-SDC: Working on IO constraints.....

Info-SDC: Categorizing output ports as bits and bussed

Info: SDC created. Please use constraints in path  /home/CAD6/Design_1/output/th
reeto8.sdc

Info: Creating TCL Script for synthesis using genus

Info: TCL Script for synthesis using genus created in path /home/CAD6/Design_1/o
utput/threeto8.tcl

Info: Creating View file for MMMC setup

Info: Created View file for MMMC setup in /home/CAD6/Design_1/output/threeto8.vi
ew

Info: Creating Innovus Script to be executed in Innovus

Info: Created Innovus Script /home/CAD6/Design_1/output/threeto8_pnr.tcl
[CAD6@cadclient3 Design_1]$
```

Figure 9: Terminal Messages after tool launch

The tool creates two **'.tcl'** script files to be executed in Genus and Innovus respectively and a **'.view'** file and **'.sdc'** file that are required by the tools. The files are generated by the script and stored in the output directory. In addition to the files mentioned, the output directory also has two more directories to store the output from the synthesis and placement & routing stages. The contents of the output directory are as shown in figure 10.

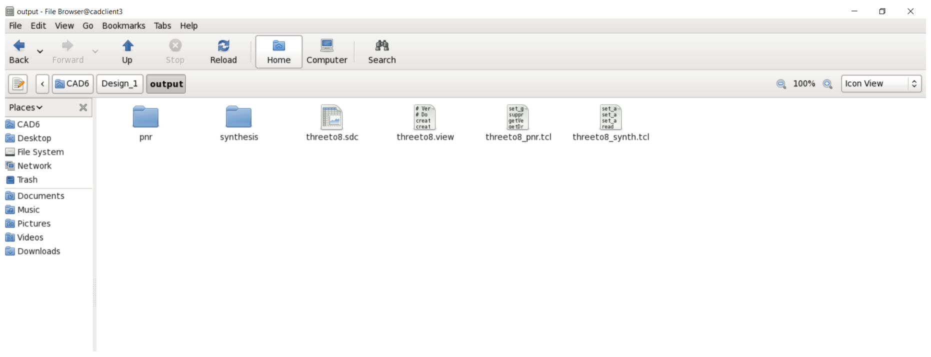
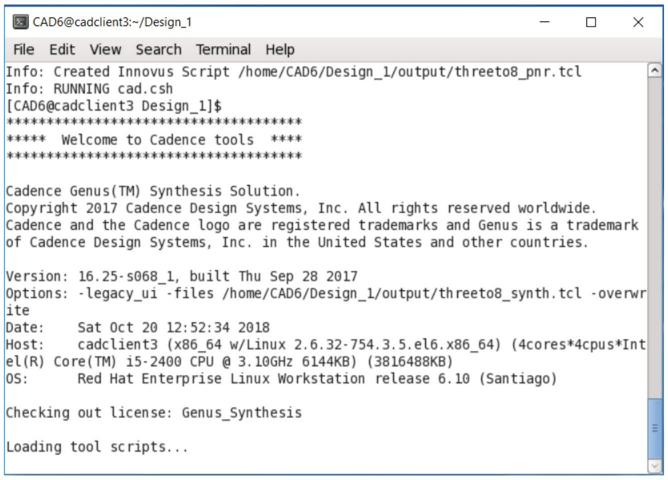


Figure 10: Contents of Output Directory

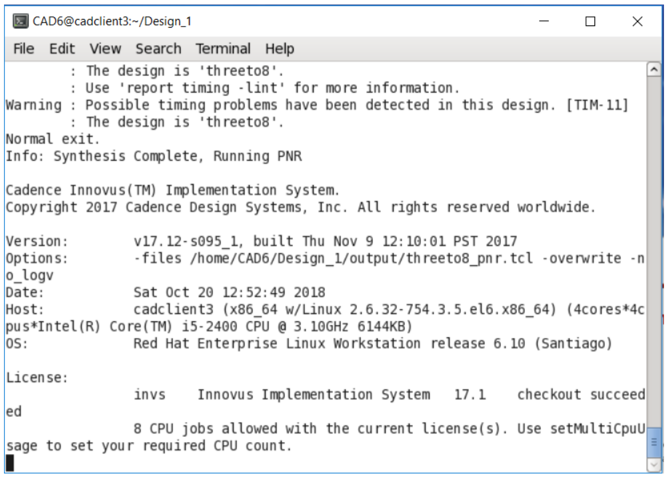
After the completion of execution of the main **'synthesis_pnr.tcl'** script, the terminal then executes the **'cad.csh'** script in the background. This is because the cadence configuration environment is in C Shell. This **'.csh'** script first sets the environment variables for cadence tools by running another C Shell Script namely **'cshrc_hub'**. This script must be present in the same directory as the **'synthesis_pnr.tcl'** script and **'cad.csh'** script to work correctly. During execution of the **'cad.csh'** script, two cadence tools namely Genus and Innovus get launched. These tools are launched by passing the generated scripts in the output directory as shown in figure 10. First the Genus tool is launched and when launched, the messages displayed on the terminal are as shown in figure 11. After the completion of synthesis, the placement and routing stage is started by launching the tool innovus as shown is figure 12. The placement and routing stage is launched only when the synthesis stage is successful. If there are any errors during synthesis, the tool is stopped and the entire process must be done again by rectifying the error shown in the terminal or in the generated log file.


CAD6@cadclient3:~/Design_1
File Edit View Search Terminal Help

```
Info: Created Innovus Script /home/CAD6/Design_1/output/threeto8_pnr.tcl
Info: RUNNING cad.csh
[CAD6@cadclient3 Design_1]$
*************************************
*****  Welcome to Cadence tools  ****
*************************************

Cadence Genus(TM) Synthesis Solution.
Copyright 2017 Cadence Design Systems, Inc. All rights reserved worldwide.
Cadence and the Cadence logo are registered trademarks and Genus is a trademark
of Cadence Design Systems, Inc. in the United States and other countries.

Version: 16.25-s068_1, built Thu Sep 28 2017
Options: -legacy_ui -files /home/CAD6/Design_1/output/threeto8_synth.tcl -overwr
ite
Date:    Sat Oct 20 12:52:34 2018
Host:    cadclient3 (x86_64 w/Linux 2.6.32-754.3.5.el6.x86_64) (4cores*4cpus*Int
el(R) Core(TM) i5-2400 CPU @ 3.10GHz 6144KB) (3816488KB)
OS:      Red Hat Enterprise Linux Workstation release 6.10 (Santiago)

Checking out license: Genus_Synthesis

Loading tool scripts...
```

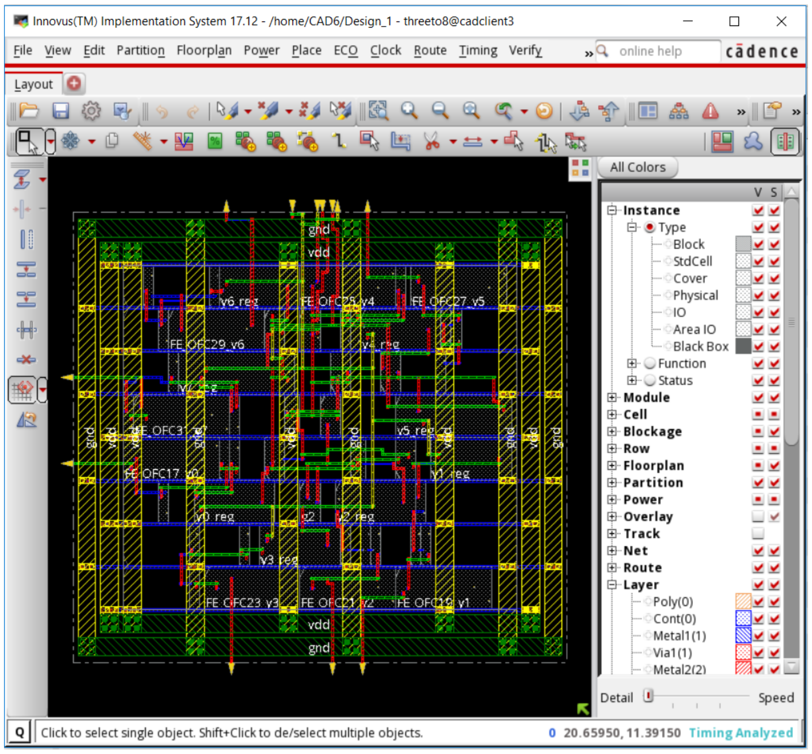
Figure 11: Terminal Messages when Genus is launched

After successful execution of the tool, there will be a layout of the design that will be apperaring as shown in figure 13 and the terminal will display a message as shown in figure 14. All output files will be stored in the output directories under the synthesis and pnr folders. All reports generated during intermediate stages will be stored in the reports folders under the synthesis and pnr folders. The output files and reports generated after synthesis are as shown in figure 15 and output files and reports generated after placement & routing are as shown in figure 16. To verify the correctness of the design generated, an output Gate Level Netlist is generated in the pnr directory. This netlist can be verified using a testbench to check if it matches the input verilog code functionality. Also a Netlist file is generated in the pnr directory which can be run in Spectre and can be used to verify the functional correctness of the generated layout.



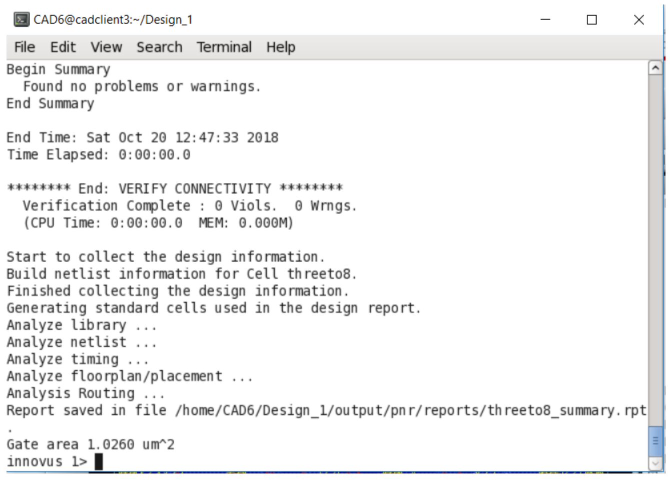
```
CAD6@cadclient3:~/Design_1
File  Edit  View  Search  Terminal  Help
        : The design is 'threeto8'.
        : Use 'report timing -lint' for more information.
Warning : Possible timing problems have been detected in this design. [TIM-11]
        : The design is 'threeto8'.
Normal exit.
Info: Synthesis Complete, Running PNR

Cadence Innovus(TM) Implementation System.
Copyright 2017 Cadence Design Systems, Inc. All rights reserved worldwide.

Version:        v17.12-s095_1, built Thu Nov 9 12:10:01 PST 2017
Options:        -files /home/CAD6/Design_1/output/threeto8_pnr.tcl -overwrite -n
o_logv
Date:           Sat Oct 20 12:52:49 2018
Host:           cadclient3 (x86_64 w/Linux 2.6.32-754.3.5.el6.x86_64) (4cores*4c
pus*Intel(R) Core(TM) i5-2400 CPU @ 3.10GHz 6144KB)
OS:             Red Hat Enterprise Linux Workstation release 6.10 (Santiago)

License:
                invs    Innovus Implementation System   17.1    checkout succeed
ed
                8 CPU jobs allowed with the current license(s). Use setMultiCpuU
sage to set your required CPU count.
```

Figure 12: Terminal Messages when Innovus is launched



```
Innovus(TM) Implementation System 17.12 - /home/CAD6/Design_1 - threeto8@cadclient3
Click to select single object. Shift+Click to de/select multiple objects.   0  20.65950, 11.39150  Timing Analyzed
```

Figure 13: Generated Layout


CAD6@cadclient3:~/Design_1
File Edit View Search Terminal Help

```
Begin Summary
  Found no problems or warnings.
End Summary

End Time: Sat Oct 20 12:47:33 2018
Time Elapsed: 0:00:00.0

******** End: VERIFY CONNECTIVITY ********
  Verification Complete : 0 Viols.  0 Wrngs.
  (CPU Time: 0:00:00.0  MEM: 0.000M)

Start to collect the design information.
Build netlist information for Cell threeto8.
Finished collecting the design information.
Generating standard cells used in the design report.
Analyze library ...
Analyze netlist ...
Analyze timing ...
Analyze floorplan/placement ...
Analysis Routing ...
Report saved in file /home/CAD6/Design_1/output/pnr/reports/threeto8_summary.rpt
.
Gate area 1.0260 um^2
innovus 1>
```

Figure 14: Terminal Messages after Placement and Routing Stage

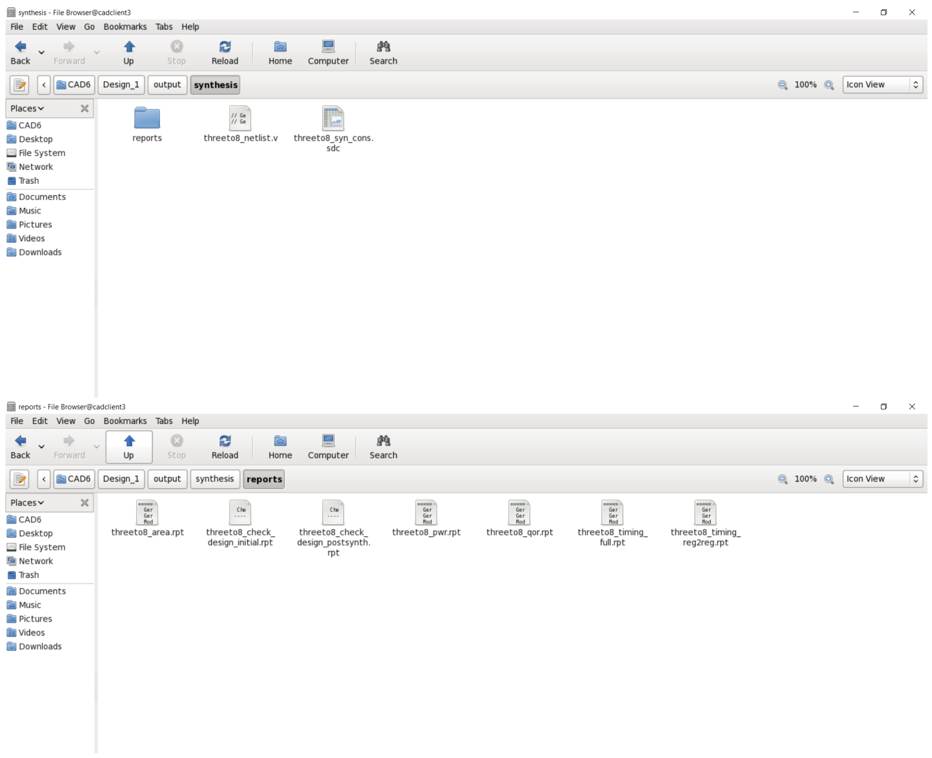

synthesis - File Browser@cadclient3
File Edit View Go Bookmarks Tabs Help
Back Forward Up Stop Reload Home Computer Search
CAD6 Design_1 output synthesis
100% Icon View
Places
CAD6
Desktop
File System
Network
Trash
Documents
Music
Pictures
Videos
Downloads
reports
threeto8_netlist.v
threeto8_syn_cons.sdc
reports - File Browser@cadclient3
File Edit View Go Bookmarks Tabs Help
Back Forward Up Stop Reload Home Computer Search
CAD6 Design_1 output synthesis reports
100% Icon View
Places
CAD6
Desktop
File System
Network
Trash
Documents
Music
Pictures
Videos
Downloads
threeto8_area.rpt
threeto8_check_design_initial.rpt
threeto8_check_design_postsynth.rpt
threeto8_pwr.rpt
threeto8_qor.rpt
threeto8_timing_full.rpt
threeto8_timing_reg2reg.rpt

Figure 15: Synthesis Output files and reports

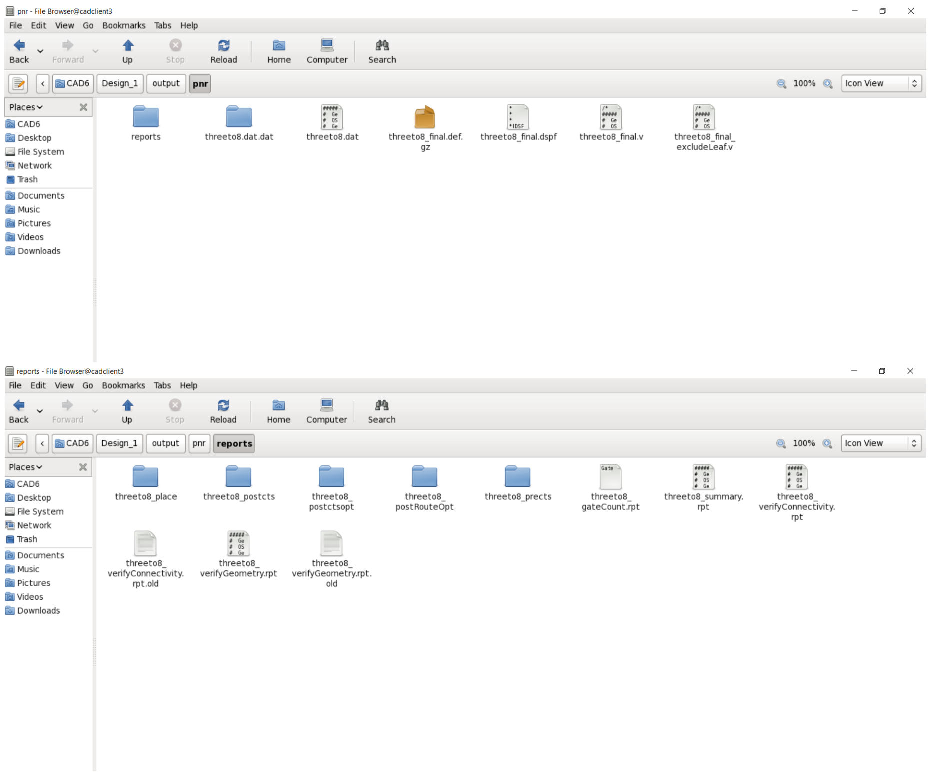


Figure 16: Placement & Routing Output files and reports

5 Conclusions

In this report, a description on the design and usage of a tool that can generate a layout from a verilog code is given. However, when giving the input, a few modifications have to be made to the verilog code. When an input or output of the verilog code is declared both as input and reg the declaration must be as below:

$$input\ a,\ b;\ -> \ Incorrect\ Declaration \tag{1}$$
$$input\ a;\ -> \ Correct\ Declaration \tag{2}$$
$$input\ b;\ -> \ Correct\ Declaration \tag{3}$$
$$output\ reg\ y;\ -> \ Incorrect\ Declaration \tag{4}$$
$$output\ y;\ -> \ Correct\ Declaration \tag{5}$$
$$reg\ y;\ -> \ Correct\ Declaration \tag{6}$$
$$output\ reg\ [1:0]\ y;\ -> \ Correct\ Declaration \tag{7}$$

This is because the synthesis_pnr.tcl script generates a **'.sdc'** file using string matching which matched the output pattern as given in the constraints file to each and every line in the verilog code(s). If the Declaration is as in 4 or 1, the code will consider the output bit as an output bus and hence the declaration should be as 2 & 3 for 1 and 5 & 6 for 4. If the declaration is for a bus as 7 then there would be no issue in generation of **'.sdc'** file. Also the ports must not be declared as input or output in the declaration of the module. If they are declared input or output inside the module definition, then the proper **'.sdc'** file will not be generated.
Also this tool has been written specifically for designs that contain a single clock. If the design is combinational or having more than one clock, then this script will raise an error and not run to completion.
This tool has been tested on only a few designs only. It cannot be run on larger designs as the placement & routing stages have some values which are fixed because it would be diffcult to estimate these values when only a behavioral verilog code is specified as an input.
Note: This tool has been designed for the design dependencies for obtaining the layout in a 45nm process node. If the design is required to give a layout in any other technology node and the dependency files remain the same, then the section of the script that generates the Innovus Script in the code **synthesis_pnr.tcl** particularly the floorplanning, ring and stripe dimensions would need to be changed as required.

End of report