Faculty of Engineering and Green Technology

Simulation and Debugging on Traffic Light Controller

Students' Name & ID No.	:	1. Anand Low Hong Ren 1801371 2. Jovin Ooi Choo Chuan 2002791 3. Ong Ting Yi 2002710
Unit Code	:	UGEB2916
Unit Tile	:	Capstone Project 2
Course	:	BTech (Honours) in Electronic Systems
Trimester (Month & Year)	:	June 2022
Academic Year	:	2022/2023
Lecturer	:	Ir. Dr. Loh Siu Hong

Table of Contents

Introduction

Humans nowadays are stepping into a world of higher technologies. It is a huge technological improvement compared to the old generation. The improvement of technology brings convenience to humans, but it also benefits in providing safety purposes to everyone. The traffic light is an example of technology that allows the driver to follow the rules by indicating the red, yellow, and green lights. Hence it helps to prevent accidents happening to drivers who follow the rules.

Nevertheless, have we ever wondered how an engineer constructs traffic light? It comprises simple components like a timer, counter, power supply, and intelligent controller. However, designing a logic system to allow more than one traffic light to work together without interruption is the most difficult part.

We enhance the previous TLC system with “smart” features during this report. Also, this project aims to illustrate the IC design pipeline. The front-end and back-end elements of the IC design cycle may be separated into two groups. The front-end design flow identifies the issue and describes a solution’s behavioural description. The RTL description is then created using the behavioural description. The RTL description is then put through functional testing and verification to ensure it functions as intended by the design specification. The actual execution of the RTL description is a part of the back-end design flow. The RTL description is converted into a gate-level netlist by logic synthesis. A physical layout is subsequently created from the gate-level netlist using floor planning and the place and route method. Verifications are carried out at each level to ensure the functionality complies with the requirements stated in the design specification. A functioning IC is created once every step has been completed successfully. The whole design path for an IC is depicted in the following figure:

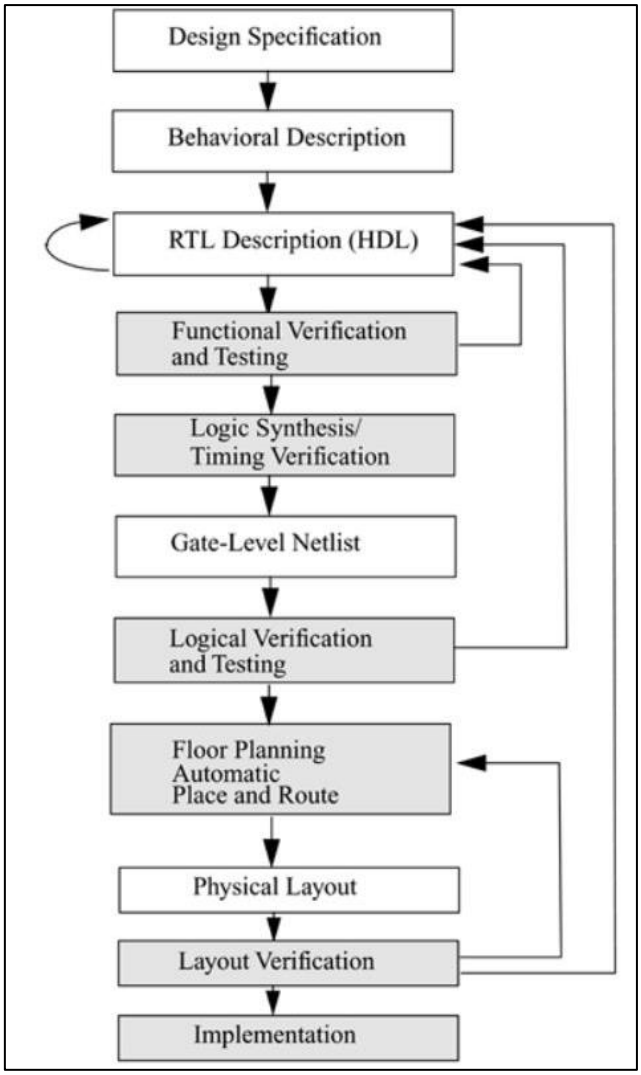


Figure 1: Design Flow of an IC

In more specific terms, the behavioural description serves as the basis for the RTL description of the design. As the tool for simulation for RTL description matured, **Synopsys VCS**, an industrial standard technique for simulation and compilation of IC designs, is frequently used. To make the process of functional verification easier, a testbench should be created in addition to the RTL description of the design. A testbench simulates the unit being tested, supplies it with a variety of inputs, and evaluates the results of the test that was run.

The RTL design is transformed into generic gates using Synopsys Design Compiler (DC) once the RTL description has undergone successful testing and functional verification. A robust gate-level synthesised netlist may be created using **Synopsys Design Compiler (DC)**, a logic synthesis tool. Additionally, technology mapping will be done on the RTL design using

linked target, link, and symbol libraries. Target library is a technology library were, during technology mapping, DC maps to the design. The definition of the cells used in the mapping design may be found in the link library, also known as the technology library. The definitions of the visual symbols used to represent the library cells are included in the symbol library.

In the event that a design constraint is sourced, DC will optimise the design to guarantee that the restrictions specified in the design constraint script file are satisfied. In order to decrease the area of the circuit and lessen the likelihood of timing violations caused by duplicate nets in the circuit, DC often performs numerous repetitions of the area recovery phase. Following compilation, DC begins by converting high-level descriptions into generic logic, followed by Boolean logic optimization. The design circuit is then mapped to include any standard cells or logic gates that are accessible. By producing time, area, and power reports at this stage, the gate-level netlist has been successfully synthesised, and the outcomes can be checked. The optimised gate-level netlist will be stored in Verilog format at the conclusion of RTL synthesis for physical design in the **Synopsys IC compiler (ICC).**

Gate-level netlist may be implemented into geometrical database standards for information interchange (GDSII) format files using the place and route system ICC. The tools required to complete the back-end design of the extremely deep submicron designs are provided by ICC. ICC will receive several inputs, including the gate-level netlist created from DC or other tools, the detailed floorplan created from prior Design Planning through ICC or other tools, the design constraints, physical and timing libraries provided by the manufacturers, as well as data for the foundry-process, in order to produce the GDSII format file that is ready for tape out of the chip. Placement of logical cells, clock tree synthesis, and routing are the three fundamental components of the place and route method, also known as physical design flow, used by the ICC.

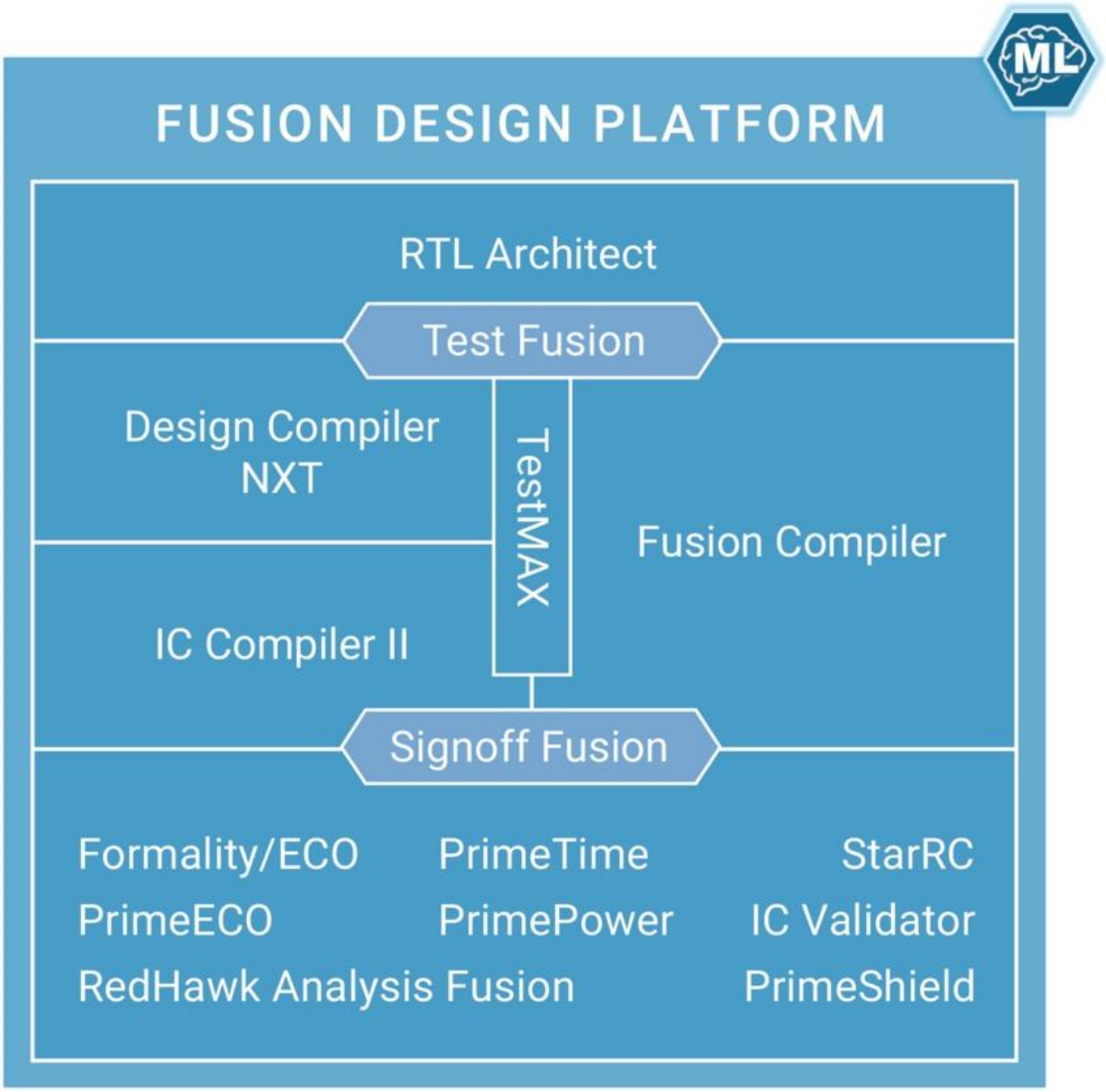


Figure 2: Physical Design Flow (Synopsys, 2022)

One way to conceptualise the IC design process is as a set of hierarchical breakdown processes. With the intention of creating a circuit on a silicon wafer that authentically fulfils the intended function, high-level requirements are broken down into more specific needs. The main steps that comprise an IC design flow are as follows:

- **Architecture Design**. Here, the necessary IC functionality is described. The capabilities of the particular IC under consideration will be considered in the context of the system being constructed. What tasks must the IC perform? What is the necessary speed and power usage? What is the device's target price? The responses to these inquiries will guide ensuing decisions regarding the precise technology that will be applied to the device's

implementation. The key word at this point is "what" is needed. It is still unclear "how" it will be put into action.

- **Logic/Circuit Design.** Here, macro-level building pieces are put together and linked to carry out the necessary IC functions. Usually, pre-existing components like memory, processors, and sensors are employed as building blocks. Circuit elements with high-level functional descriptions are broken down into the necessary low-level circuit parts. Logic synthesis software is used to automate this procedure. To confirm the design's functionality, the group of devices is simulated. Depending on the amount of modelling detail required, either a digital logic simulator or an analogue circuit simulator will be employed. Custom circuit design strategies are employed if the macro-level building blocks need to be changed in order to meet the needs of the IC. This stage sees the beginning of defining "how" the chip will be implemented.
- **Physical Design.** The layout of the linked forms that actually implement all the necessary circuit parts on the silicon wafer is produced in this step. The first step in the procedure is to create a "floor plan" for the chip, which specifies the locations of the chip's main operations as well as its input and output ports. In order to prepare for manufacture, the last circuit components are then put into position and routed. Custom layout procedures using an IC layout editing tool are employed if the macro-level building blocks need to be changed to meet the needs of the IC. The chips "how" have now been thoroughly established.
- **Physical Verification.** Now it is possible to simulate every physical consequence that the manufacturing process contributes to the design. There are various factors that need to be considered in this case, including additional resistance from the wiring, signal crosstalk, and unpredictability in the manufacturing process itself. Will the circuit continue to function properly under these strains? In order for the circuit to be manufactured, there are several design requirements for how it must be physically arranged on the silicon wafer. This stage also involves checking these design principles.
- **Signoff.** Before the design is delivered to manufacturing, this is the last phase. Here, all of the crucial factors that will affect the chip's functionality or ability to be manufactured are checked against the outcomes of "golden signoff" quality tools. During this stage, design rules and design for manufacturability rules are both completely verified. This process also involves verifying and "closing" the design's timing, power use, and signal integrity. To

guarantee that the physical impacts of the procedure are thoroughly understood, precise parasite extraction must be carried out during signoff.

Objectives

In this assignment, there are several objectives to achieve. First, we must create a source file and testbench for the enhanced Traffic Light Controller (TLC) system. We should have sensors in all directions: West, South, East, and North. Then will prioritise the direction with the most vehicles recognised by the TLC system. Next, we compile, simulate and view the testbench signals in DVE, together with the explanation of the waveforms by referring to the testbench and design codes. After that, we invoke Design Vision from the appropriate directory by screenshot after the read and link steps for the record purpose. Lastly, an examination of the timing and area information of the synthesised schematic has to be done once the RTL code for the traffic light controller has been synthesised into a gate-level netlist. The gate-level netlist created by design compilation is then used for placement, clock tree synthesis, and routing to finish the place and route process.

Methodology

Figure 3: System design flow

Figure 3 shows the design flow for the system. The first task of the assignment is to create a source file and testbench for the enhanced traffic light controller. The general specification of the system is provided. On top of having all the features of a basic traffic light controller designed beforehand, the new enhanced traffic light controller should have sensors in all directions. Besides the system should be able to recognize which direction has the greatest number of vehicles and prioritize the respective direction.

A 3-bit output register is used for a better viewing and understanding during coding and debugging.

Five cases are designed for the traffic light controller to switch around, first case will trigger and operates when the vehicle count is equal for the four lanes or all of them are empty, the traffic flows for this case is goes from West to South, then to East and lastly North.

Second case happens when the West direction has the most car, then the system will prioritize the lanes in West direction. Third case happens when the South direction has the most car, priority will be given to the lanes in South direction. Followed by the East direction as the fourth case and North direction as the fifth case.

The Verilog source file and testbench are first done in EDA Playground as we are more familiar with that website, it will be easier for us to design and debug the system there.

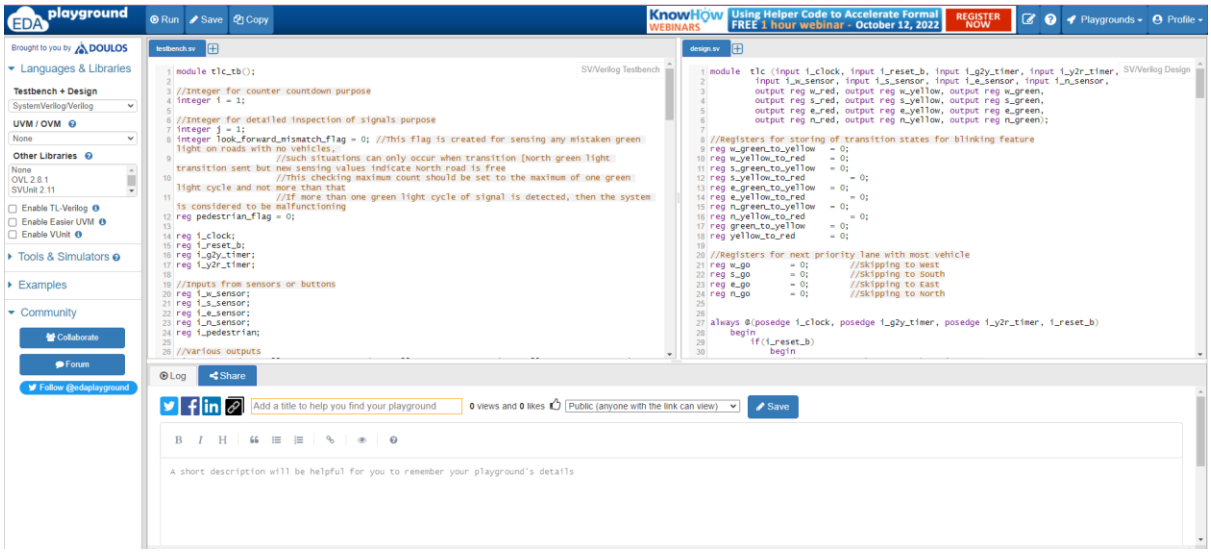


Figure 4: EDA Playground

After the system operates successfully, we moved to the second task which is to compile the Verilog files and generate the simulation binary executable (simv). The testbench and traffic light controller Verilog source file are compiled and simulated using Synopsys VCS by using the "vcs Verilog File>" command. Followed by simulation using "./simv" command. The messages generated after compilation and simulation are recorded. By adding "-gui" command after "./simv " command and adding "-debug all" command to the compilation command line will launch the DVE graphical user interface which used to visualize and analyse the output results. The analysed results are discussed and recorded. The RTL coding is then purposefully made to contain errors to keep track of the testbench. The error message generated are recorded and discussed.

The design compiler was launched using the command line. The Verilog code must comply strict RTL coding rules or the design compiler will have error reading the design. The Verilog code has to be modified until it complies the RTL coding rules, only then will the design compiler will read the design.

To create a gate-level netlist, the design compiler performs logic synthesis for the RTL coding. All relevant libraries, including the target library, link library, and symbol library, as well as the design, are imported into the design compiler during the Data Setup stage. The compilation of the gate-level netlist is done during the Technology Mapping stage using the "-compile" command. After the compilation is done, all the messages generated are recorded. Based on

the libraries linked, the design compiler automatically conducts the technology mapping. It is noticed and noted how the schematic views change before and after compilation. The "-report timing" and "-report area" command options, respectively, are used to generate the timing report and the area report during the Design Checking step. The optimised gate-level netlist, also known as the gate-level netlist, is finally saved into Verilog file format during the Post-Synthesis Output Data stage. The symbol views and schematic views generated are recorded and discussed.

Similar to the design compiler, the IC Compiler was launched using the command line. There are five phases for IC Compiler to carry out in order to produce a proper physical layout of RTL design.

In the first phase, data setup phase, the gate-level netlist that was generated in the previous task was imported to the IC Compiler. The compiled RTL was saved in ".v" format. Then a Milkyway design library was created where we specify the technology file, attach reference libraries and load TLU+ models to achieve an accurate parasitic modelling. Then the logical connections between the power/grounds and nets are defined.

Aspect ratio and core use are determined during the floor planning phase. While the utilisation specifies the space filled by the standard cells, macros, and additional cells, the aspect ratio dictates the size and shape of the chip. Additionally, the power straps and rectangle rings are made.

The next step is core placement and optimization to make sure the pads have sufficient ground and power connections. To reduce clock skew and insertion delay, Clock Tree Synthesis (CTS) is used. Routing and optimization of the core are the last steps.

Results

Task 1: Create a source file and testbench for the traffic light controller

The Verilog source code for the traffic light controller and testbench has been successfully coded and are shown in the appendix section as Listing 1 and Listing 2 respectively. The following shows the flowchart of the traffic light controller:

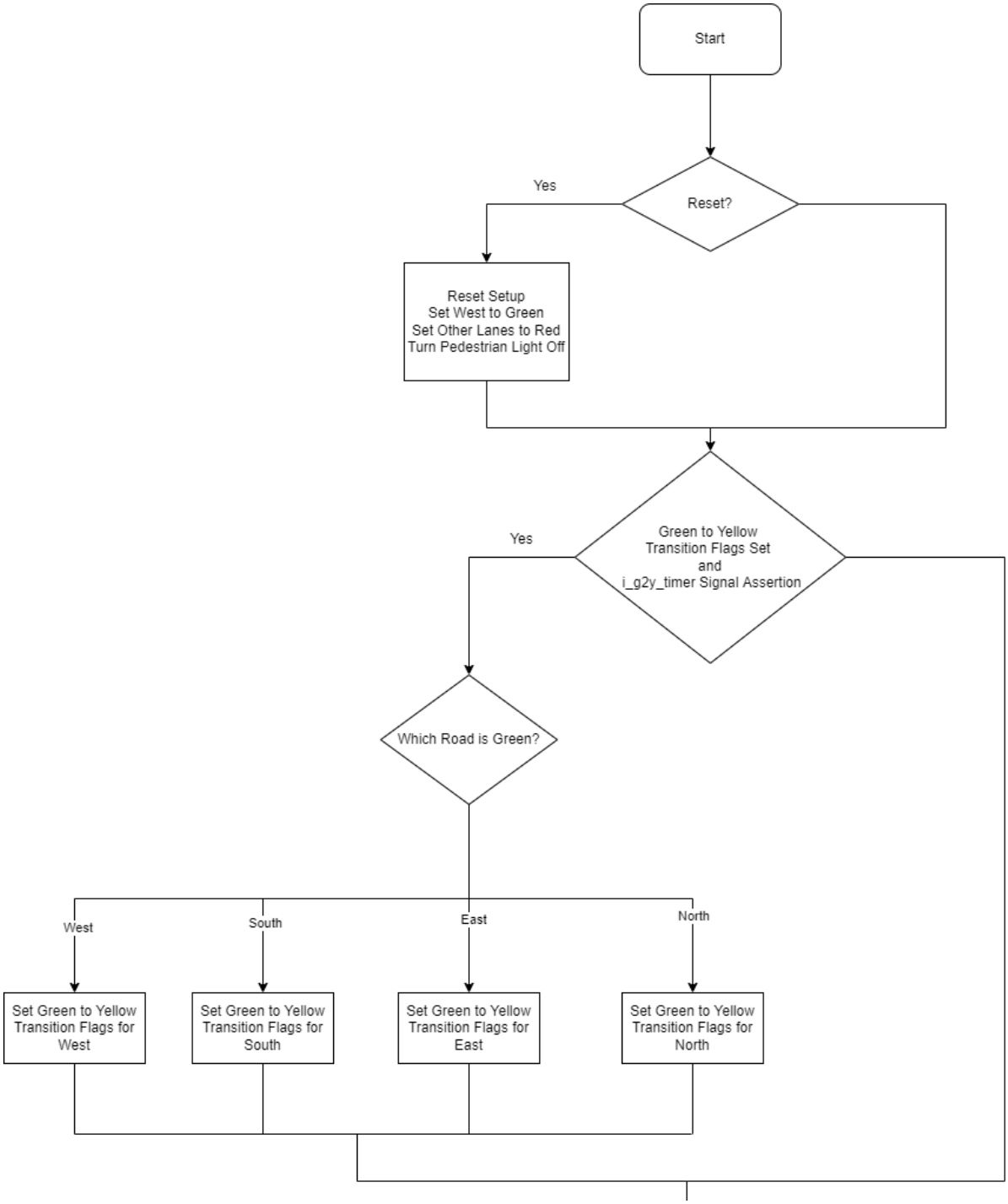


Figure 5: Traffic Light Controller Flowchart Part 1.

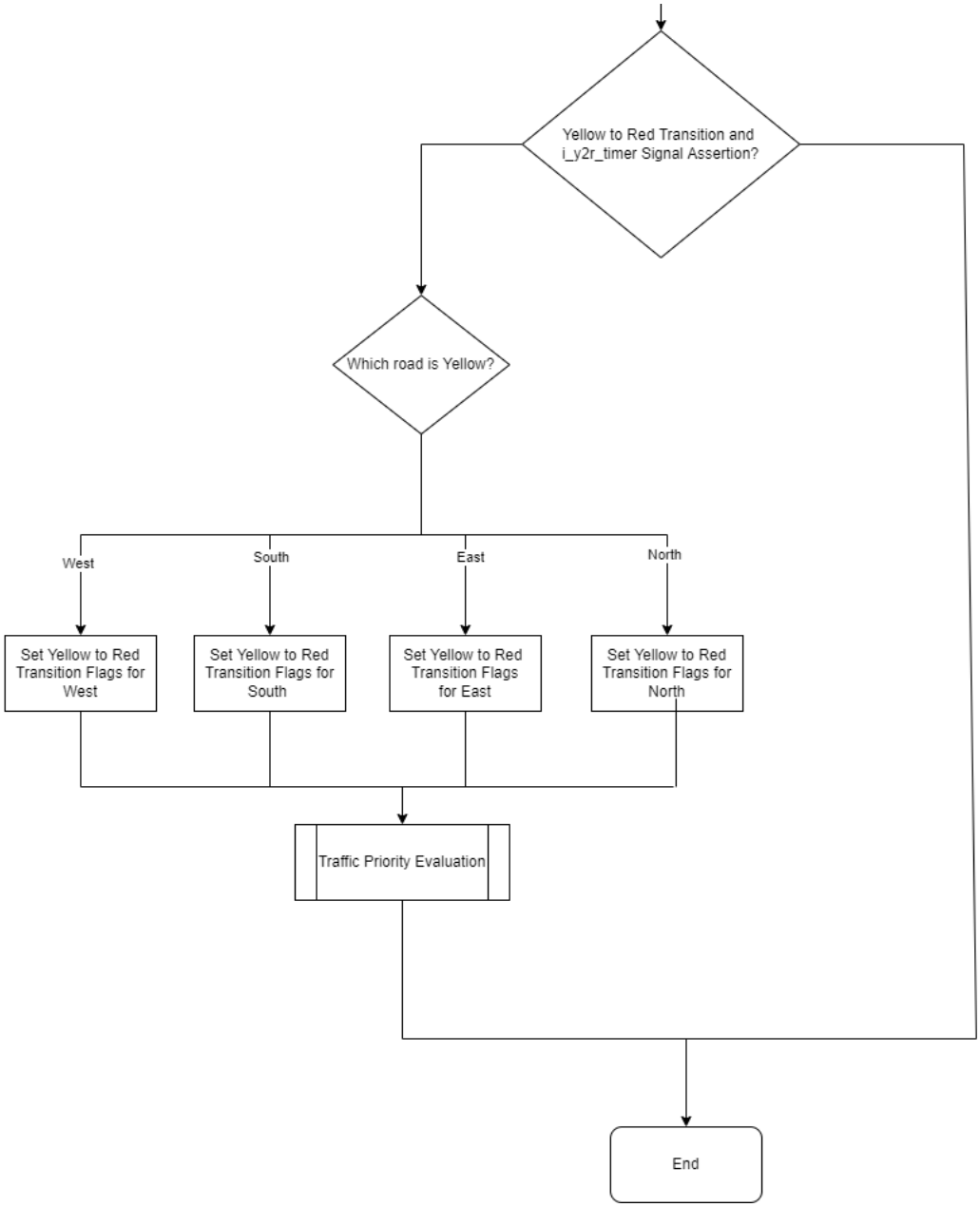


Figure 6: Traffic Light Controller Flowchart Part 2.

Task 2: Compilation and simulation

When the traffic light controller Verilog coding and the testbench Verilog coding are compiled and simulated, the following messages are generated:

```
[user@vm1 assignment]$ vcs tlc_tb.v tlc.v -o tlctest
                        Chronologic VCS (TM)
      Version P-2019.06-SP2-1_Full64 -- Sun Sep 11 15:31:28 2022
               Copyright (c) 1991-2019 by Synopsys Inc.
                        ALL RIGHTS RESERVED

This program is proprietary and confidential information of Synopsys Inc.
and may be used and disclosed only as authorized in a license agreement
controlling such use and disclosure.

Parsing design file 'tlc_tb.v'
Parsing design file 'tlc.v'
Top Level Modules:
       tlc_tb
No TimeScale specified
Starting vcs inline pass...
1 module and 0 UDP read.
recompiling module tlc_tb
rm -f _csrc*.so pre_vcsobj_*.so share_vcsobj_*.so
if [ -x ../tlctest ]; then chmod a-x ../tlctest; fi
g++  -o ../tlctest    -rdynamic  -Wl,-rpath='$ORIGIN'/tlctest.daidir -Wl,-rpath=./tlct
est.daidir -Wl,-rpath=/synopsys/vcs/P-2019.06-SP2-1/linux64/lib -L/synopsys/vcs/P-2019
.06-SP2-1/linux64/lib  -Wl,-rpath-link=./   objs/amcQw_d.o   _3825_archive_1.so  SIM_l
.o       rmapats_mop.o rmapats.o rmar.o rmar_nd.o  rmar_llvm_0_1.o rmar_llvm_0_0.o
      -lnuma -lvirsim -lerrorinf -lsnpsmalloc -lvfs    -lvcsnew -lsimprofile -luclinat
ive /synopsys/vcs/P-2019.06-SP2-1/linux64/lib/vcs_tls.o   -Wl,-whole-archive -lvcsucli
 -Wl,-no-whole-archive          /synopsys/vcs/P-2019.06-SP2-1/linux64/lib/vcs_save_res
tore_new.o -ldl  -lc -lm -lpthread -ldl
../tlctest up to date
CPU time: .649 seconds to compile + .893 seconds to elab + .634 seconds to link
```

Figure 7: Messages Generated after VCS Verilog Code Compilation.

```
[user@vm1 assignment]$ ./tlctest
Chronologic VCS simulator copyright 1991-2019
Contains Synopsys proprietary information.
Compiler version P-2019.06-SP2-1_Full64; Runtime version P-2019.06-SP2-1_Full64;  Sep
11 15:34 2022
***Setup Completion - Begin Testing Traffic Light Controller***
Sensor States: west: 00 south: 00 east: 00 north: 00
Sensor States: west: 11 south: 10 east: 01 north: 00
Sensor States: west: 00 south: 11 east: 10 north: 01
Sensor States: west: 01 south: 00 east: 11 north: 10
Sensor States: west: 10 south: 01 east: 00 north: 11
*** Testbench Successfully Passed! ***
$finish called from file "tlc_tb.v", line 201.
$finish at simulation time                25300
           V C S   S i m u l a t i o n   R e p o r t
Time: 25300
CPU Time:      0.980 seconds;       Data structure size:   0.0Mb
Sun Sep 11 15:34:37 2022
```

Figure 8: Messages Generated after VCS Verilog Code Simulation.

Task 3: Explanation using DVE

For easier viewing, the traffic light outputs have been concatenated into a 3-bit output with the following parameters:

Table 1: Concatenated Traffic Light Output Indication.

Traffic Light Hexadecimal Output	Traffic Light Bit Output	Traffic Light Display Output
0	000	No Light (Blinking)
1	001	Green
2	010	Yellow
4	100	Red

Traffic Light Output [2:0] = {Red, Yellow, Green}

When vehicle count is equal for the four lanes or all of them are empty, the traffic flows from West → South → East → North.

Case 1: Normal Operation when all lanes have the same number of vehicles

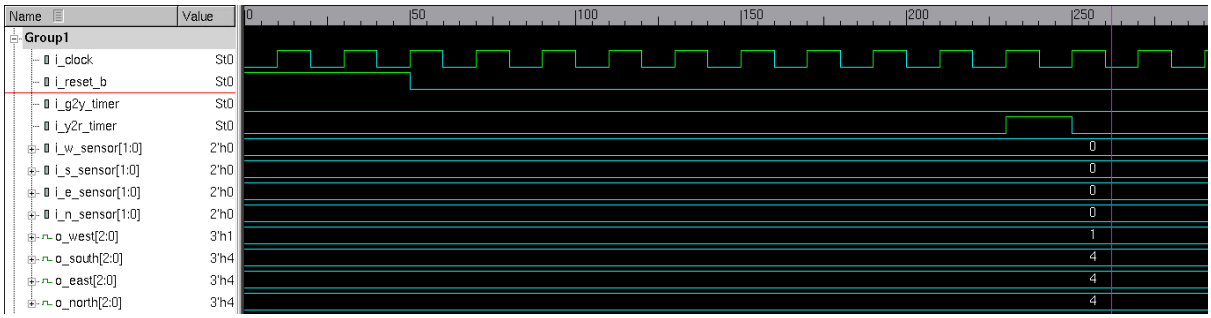


Figure 9: West light turns green while other lights turn red.

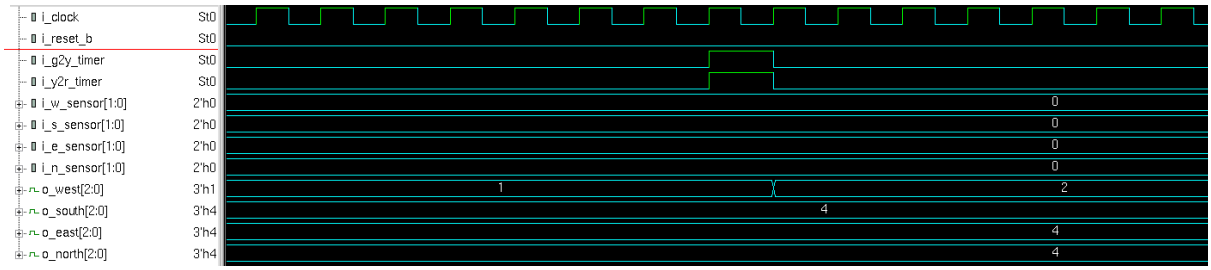


Figure 10: West light transitions to yellow light after 15 seconds.

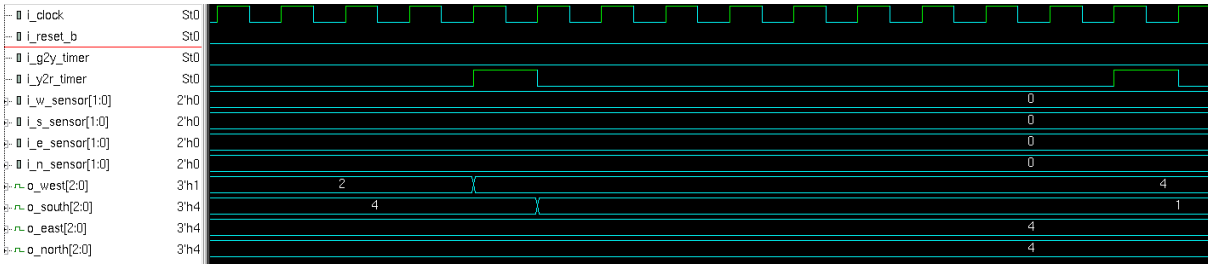


Figure 11: West light transitions from yellow to red light, then South light transitions from Red to Green.

Case 2: West most car

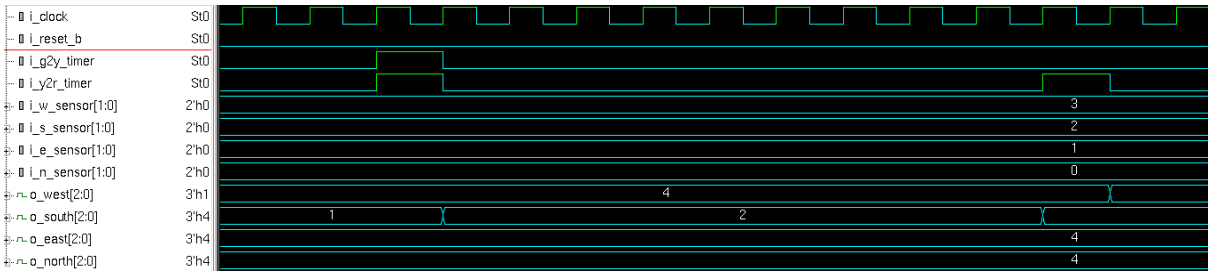


Figure 12: Currently, the South is Green, and the traffic light controller detects that the West Lane has the most vehicles. Thus, priority is given to West Lane. West Lane will turn green next.

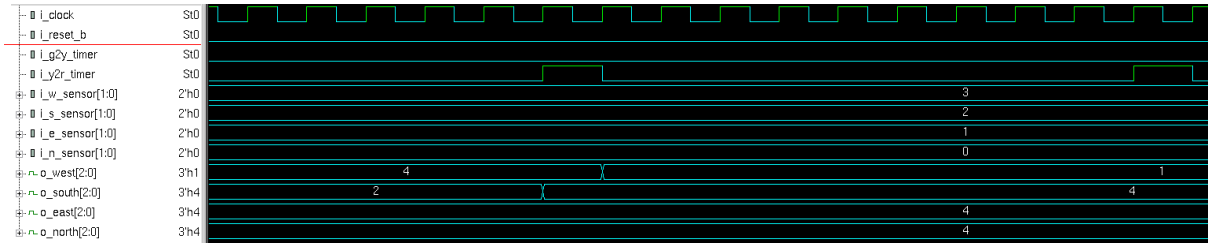


Figure 13: The priority system works since West Lane is given priority instead of East, following the normal operating sequence.

Case 3: South most car

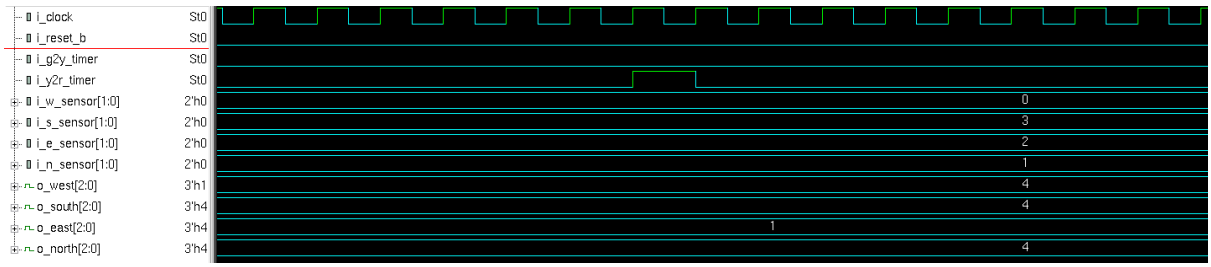


Figure 14: Currently, East Lane is green. South Lane has the most vehicles; thus, it is given priority to turn green next.

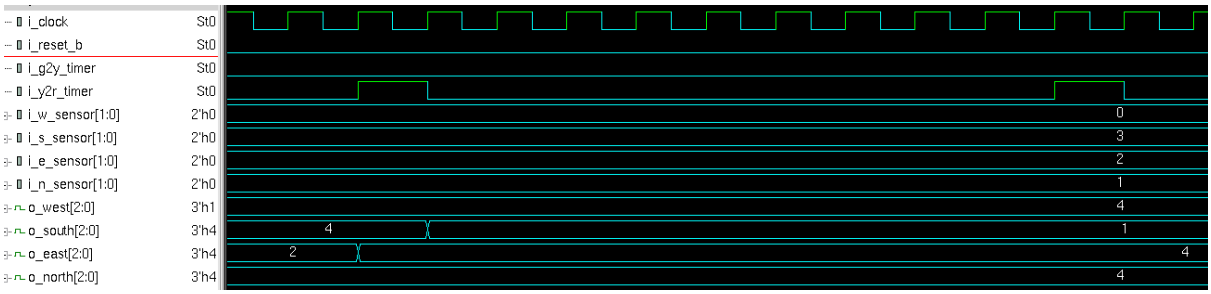


Figure 15: South Lane turns green after.

Case 4: In East Lane, most car

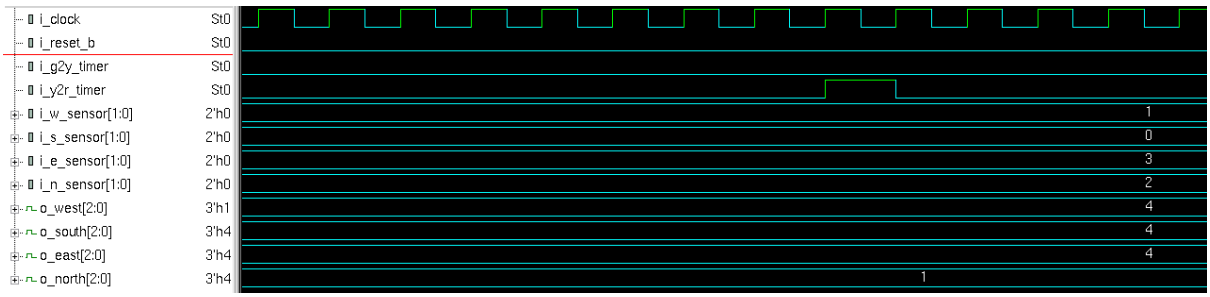


Figure 16: Currently, North is green. East has the most vehicles; thus, it is given priority to turn green next.

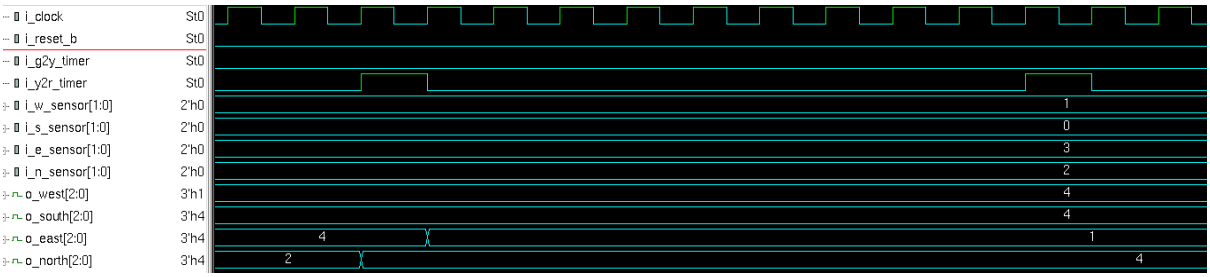


Figure 17: North turns red while East turns green.

Case 5: North most car

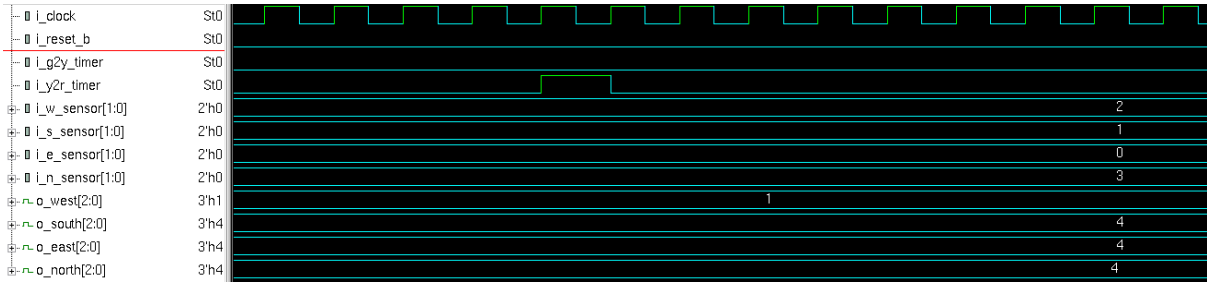


Figure 18: Currently, West is green. North has the most vehicles. Thus, it is given priority to turn green next.

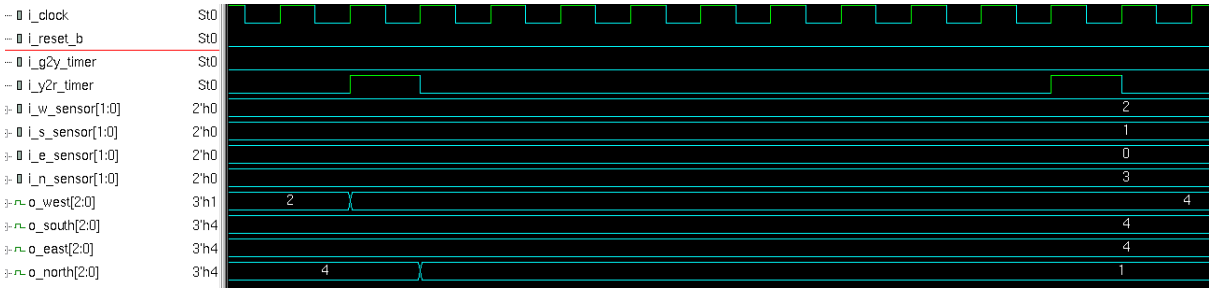


Figure 19: West turns red, North turns green.

Task 4: Design compiler synthesis steps

```
Warning:  /home/user/Documents/UGEB2916/anand/dc/lab2/rtl/Anand.v:10: The construct 'declaration initial assignment' is not supported in synthesis; it is
Warning:  /home/user/Documents/UGEB2916/anand/dc/lab2/rtl/Anand.v:11: The construct 'declaration initial assignment' is not supported in synthesis; it is
Warning:  /home/user/Documents/UGEB2916/anand/dc/lab2/rtl/Anand.v:12: The construct 'declaration initial assignment' is not supported in synthesis; it is
Warning:  /home/user/Documents/UGEB2916/anand/dc/lab2/rtl/Anand.v:13: The construct 'declaration initial assignment' is not supported in synthesis; it is
Warning:  /home/user/Documents/UGEB2916/anand/dc/lab2/rtl/Anand.v:14: The construct 'declaration initial assignment' is not supported in synthesis; it is
Warning:  /home/user/Documents/UGEB2916/anand/dc/lab2/rtl/Anand.v:15: The construct 'declaration initial assignment' is not supported in synthesis; it is
Warning:  /home/user/Documents/UGEB2916/anand/dc/lab2/rtl/Anand.v:16: The construct 'declaration initial assignment' is not supported in synthesis; it is
Warning:  /home/user/Documents/UGEB2916/anand/dc/lab2/rtl/Anand.v:17: The construct 'declaration initial assignment' is not supported in synthesis; it is
Warning:  /home/user/Documents/UGEB2916/anand/dc/lab2/rtl/Anand.v:18: The construct 'declaration initial assignment' is not supported in synthesis; it is
Warning:  /home/user/Documents/UGEB2916/anand/dc/lab2/rtl/Anand.v:19: The construct 'declaration initial assignment' is not supported in synthesis; it is
Warning:  /home/user/Documents/UGEB2916/anand/dc/lab2/rtl/Anand.v:22: The construct 'declaration initial assignment' is not supported in synthesis; it is
Warning:  /home/user/Documents/UGEB2916/anand/dc/lab2/rtl/Anand.v:23: The construct 'declaration initial assignment' is not supported in synthesis; it is
Warning:  /home/user/Documents/UGEB2916/anand/dc/lab2/rtl/Anand.v:24: The construct 'declaration initial assignment' is not supported in synthesis; it is
Warning:  /home/user/Documents/UGEB2916/anand/dc/lab2/rtl/Anand.v:25: The construct 'declaration initial assignment' is not supported in synthesis; it is
Error:  /home/user/Documents/UGEB2916/anand/dc/lab2/rtl/Anand.v:28: The event depends on both edge and nonedge expressions, which synthesis does not supp
*** Presto compilation terminated with 1 errors. ***
Error: Can't read 'verilog' file '/home/user/Documents/UGEB2916/anand/dc/lab2/rtl/Anand.v'. (UID-59)
No designs were read
design_vision>
```

Figure 20: Designer compiler have error reading design

As seen in Figure 18, the design compiler has error reading the design due to the Verilog Code not complying strict RTL coding rules. The Verilog code has to be modified until it complies the RTL coding rules, only then will the design compiler will read the design.

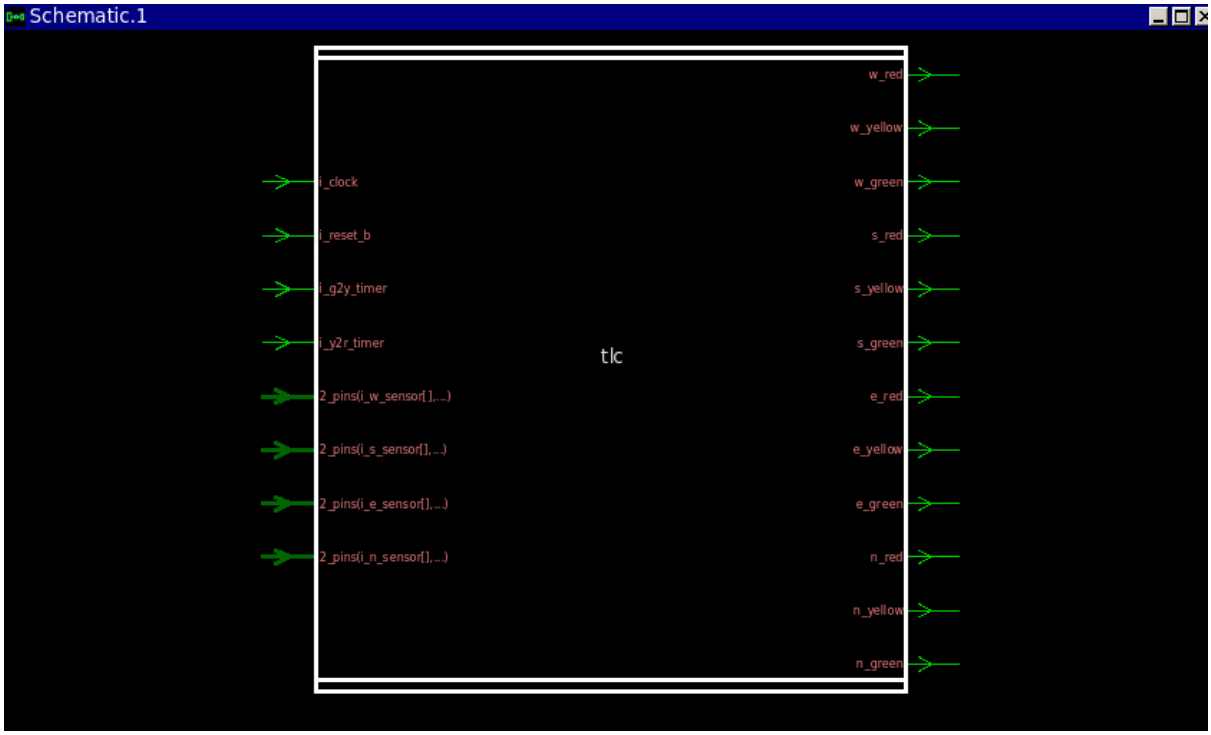


Figure 21: Symbol View of Traffic Light Controller

When the RTL description is successfully imported into the Design Compiler, the symbol view of the RTL description is as shown in Figure 19.

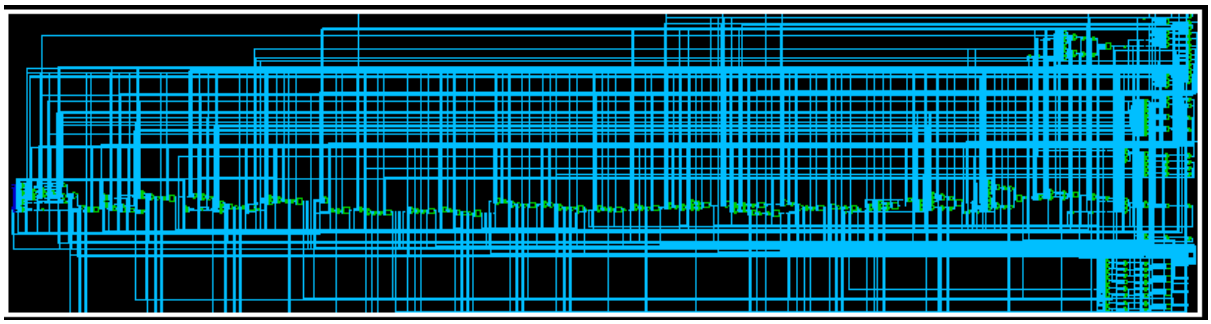


Figure 22: Schematic View of Traffic Light Controller before technology mapping

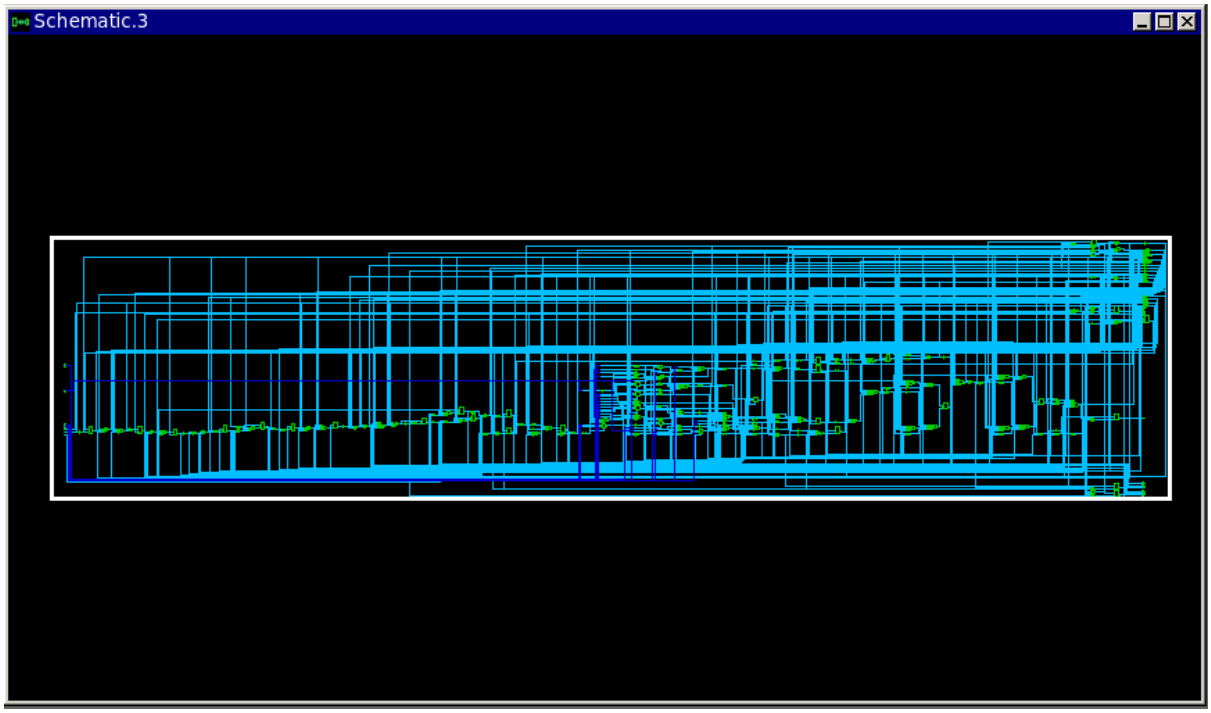


Figure 23: Schematic View of Traffic Light Controller after Technology Mapping

The following figures show the zoomed-in schematic view of the cells utilized in the design before and after technology mapping respectively:

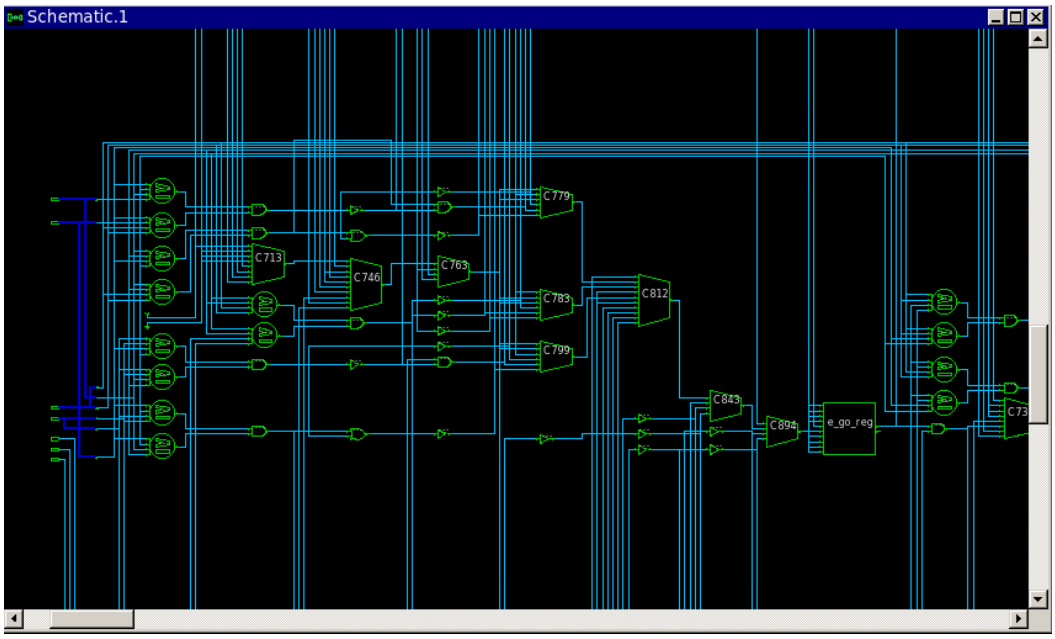


Figure 24: Zoomed-in Schematic View before technology mapping

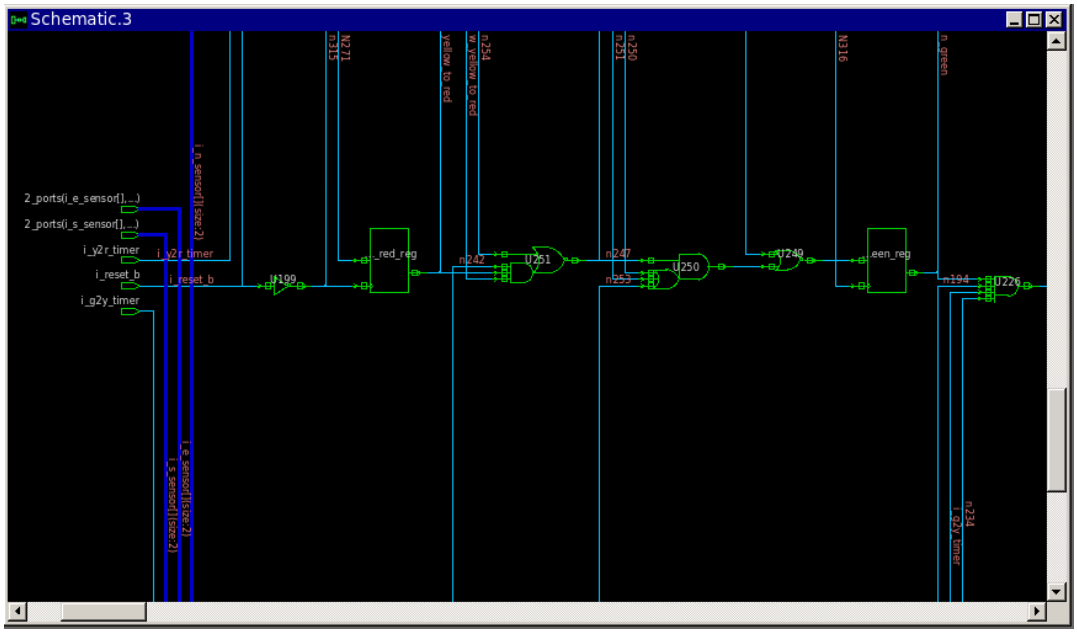


Figure 25: Zoomed-in Schematic View after technology mapping

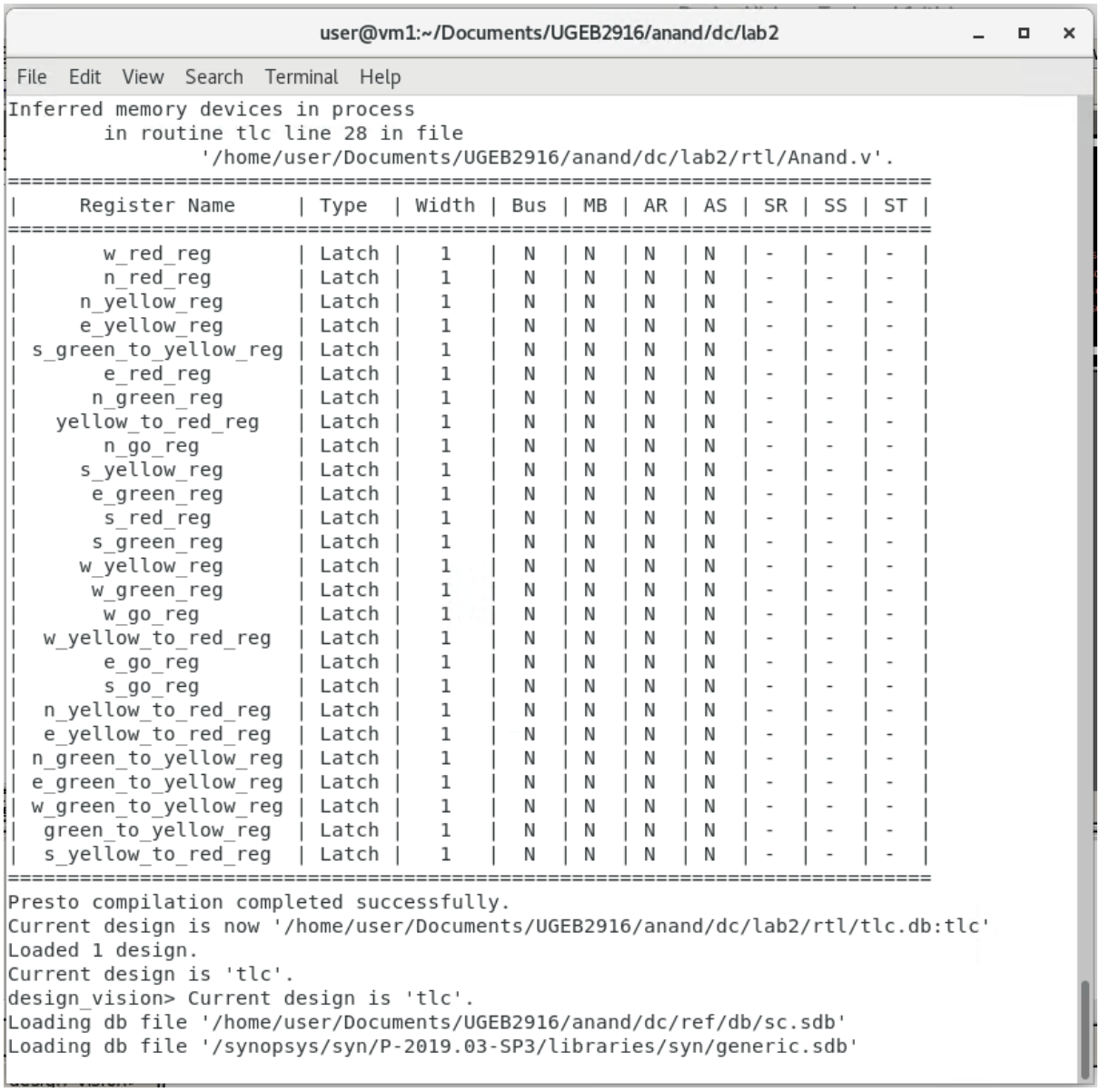


Figure 26: Messages generated after Design Compiler Read and Link

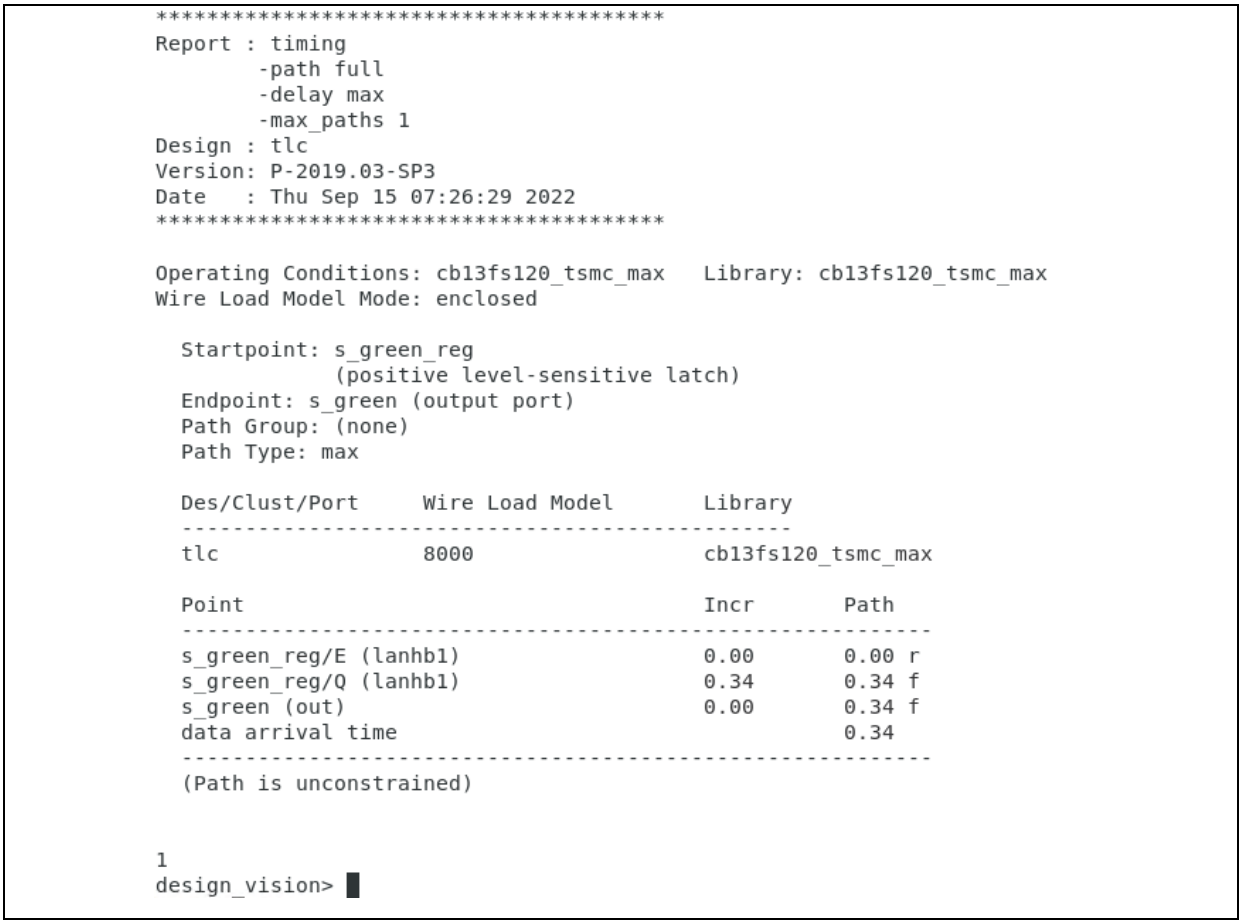


Figure 27: Report Timing of the Compiled gate-Level Netlist

```
****************************************
Report : area
Design : tlc
Version: P-2019.03-SP3
Date   : Thu Sep 15 07:29:35 2022
****************************************

Library(s) Used:

    cb13fs120_tsmc_max (File: /home/user/Documents/UGEB2916/anand/dc/ref/db/sc_max.db)

Number of ports:                          24
Number of nets:                          215
Number of cells:                         198
Number of combinational cells:           172
Number of sequential cells:               26
Number of macros/black boxes:              0
Number of buf/inv:                        44
Number of references:                     35

Combinational area:               244.500000
Buf/Inv area:                      33.000000
Noncombinational area:             92.500000
Macro/Black Box area:               0.000000
Net Interconnect area:             59.163449

Total cell area:                  337.000000
Total area:                       396.163449
1
design_vision>
```

Figure 28: Report Area of the Compiled Gate-Level Netlist

```
  Beginning Implementation Selection
  ----------------------------------

  Beginning Mapping Optimizations  (Medium effort)
  -------------------------------
  Structuring 'tlc'
  Mapping 'tlc'

                                TOTAL
   ELAPSED            WORST NEG   SETUP    DESIGN
    TIME      AREA      SLACK     COST    RULE COST         ENDPOINT
  --------- --------- --------- --------- --------- -------------------------
    0:00:54     337.8      0.00       0.0       1.0
    0:00:54     337.8      0.00       0.0       1.0
    0:00:54     337.8      0.00       0.0       1.0
    0:00:55     337.8      0.00       0.0       1.0
    0:00:55     337.8      0.00       0.0       1.0
    0:00:55     337.8      0.00       0.0       1.0
    0:00:55     337.8      0.00       0.0       1.0
    0:00:55     337.8      0.00       0.0       1.0
    0:00:55     337.8      0.00       0.0       1.0
    0:00:55     337.8      0.00       0.0       0.0
    0:00:55     337.8      0.00       0.0       0.0
    0:00:55     337.8      0.00       0.0       0.0
    0:00:55     337.8      0.00       0.0       0.0


  Beginning Delay Optimization Phase
  ----------------------------------

                                TOTAL
   ELAPSED            WORST NEG   SETUP    DESIGN
    TIME      AREA      SLACK     COST    RULE COST         ENDPOINT
  --------- --------- --------- --------- --------- -------------------------
    0:00:55     337.8      0.00       0.0       0.0
    0:00:55     337.8      0.00       0.0       0.0
    0:00:55     337.8      0.00       0.0       0.0


  Beginning Area-Recovery Phase  (cleanup)
  ------------------------------

                                TOTAL
   ELAPSED            WORST NEG   SETUP    DESIGN
    TIME      AREA      SLACK     COST    RULE COST         ENDPOINT
  --------- --------- --------- --------- --------- -------------------------
    0:00:55     337.8      0.00       0.0       0.0
    0:00:55     337.8      0.00       0.0       0.0
    0:00:55     337.0      0.00       0.0       0.0
    0:00:55     337.0      0.00       0.0       0.0
    0:00:55     337.0      0.00       0.0       0.0
    0:00:55     337.0      0.00       0.0       0.0
    0:00:55     337.0      0.00       0.0       0.0
    0:00:55     337.0      0.00       0.0       0.0
    0:00:55     337.0      0.00       0.0       0.0
    0:00:55     337.0      0.00       0.0       0.0
    0:00:55     337.0      0.00       0.0       0.0
    0:00:55     337.0      0.00       0.0       0.0
Loading db file '/home/user/Documents/UGEB2916/anand/dc/ref/db/sc_max.db'


Note: Symbol # after min delay cost means estimated hold TNS across all active scenarios


  Optimization Complete
  ---------------------
1
```

```
****************************************
  Report : Chip Summary
  Design : tlc
  Version: R-2020.09-SP5
  Date   : Thu Sep 29 06:21:32 2022
****************************************
Std cell utilization: 73.50%  (1348/(1834-0))
(Non-fixed + Fixed)
Std cell utilization: 73.50%  (1348/(1834-0))
(Non-fixed only)
Chip area:            1834     sites, bbox (1.00 1.00 54.71 52.66) um
Std cell area:        1348     sites, (non-fixed:1348   fixed:0)
                      198      cells, (non-fixed:198    fixed:0)
Macro cell area:      0        sites
                      0        cells
Placement blockages:  0        sites, (excluding fixed std cells)
                      0        sites, (include fixed std cells & chimney area)
                      0        sites, (complete p/g net blockages)
Routing blockages:    0        sites, (partial p/g net blockages)
                      0        sites, (routing blockages and signal pre-route)
Lib cell count:       35
Avg. std cell width:  3.36 um
Site array:           unit     (width: 0.41 um, height: 3.69 um, rows: 14)
Physical DB scale:    1000 db_unit = 1 um
```

Figure 29: Messages Generated after Design Compiler Technology Mapping

Task 5: IC Compiler data setup and basic flow

The following figures verify the data setup for the parasitic information as well as the linked library respectively:

```
Loaded 1 design.
Current design is 'tlc'.
Current design is 'tlc'.

  Linking design 'tlc'
  Using the following designs and libraries:
  --------------------------------------------------------------------------
  tlc                         /home/user/Documents/UGEB2916/anand/dc/lab2/mapped
/tlcnetlist.ddc
  cb13fs120_tsmc_max (library) /home/user/Documents/UGEB2916/anand/dc/lab2/mappe
d/icc/ref/db/sc_max.db
  cb13io320_tsmc_max (library) /home/user/Documents/UGEB2916/anand/dc/lab2/mappe
d/icc/ref/db/io_max.db
  ram16x128_max (library)     /home/user/Documents/UGEB2916/anand/dc/lab2/mapped
/icc/ref/db/ram16x128_max.db

Info: Creating auto CEL.
Preparing data for query...................
Information: Performing CEL netlist consistency check. (MWDC-118)
Information: CEL consistency check PASSED. (MWDC-119)
Information: Saved design named tlc. (UIG-5)
Preparing data for query...................
1
icc_shell>
```

Figure 30: TLU+ Files Setup

```
icc_shell> list_libs
Logical Libraries:
--------------------------------------------------------------------------
Library         File                    Path
-------         ----                    ----
M cb13fs120_tsmc_max sc_max.db          /home/user/Documents/UGEB2916/anand/dc/lab2/mapped/icc/ref/db
m cb13fs120_tsmc_min sc_min.db          /home/user/Documents/UGEB2916/anand/dc/lab2/mapped/icc/ref/db
M cb13io320_tsmc_max io_max.db          /home/user/Documents/UGEB2916/anand/dc/lab2/mapped/icc/ref/db
m cb13io320_tsmc_min io_min.db          /home/user/Documents/UGEB2916/anand/dc/lab2/mapped/icc/ref/db
M ram16x128_max ram16x128_max.db        /home/user/Documents/UGEB2916/anand/dc/lab2/mapped/icc/ref/db
m ram16x128_min ram16x128_min.db        /home/user/Documents/UGEB2916/anand/dc/lab2/mapped/icc/ref/db
  gtech         gtech.db                /synopsys/icc/R-2020.09-SP5/libraries/syn
  standard.sldb standard.sldb           /synopsys/icc/R-2020.09-SP5/libraries/syn
1
icc_shell>
```

Figure 31: Linked Library Files

```
icc_shell> derive_pg_connection -power_net VDD -power_pin VDD -ground_net VSS -ground_pin VSS
Information: connected 198 power ports and 198 ground ports
1
icc_shell> derive_pg_connection -power_net VDD -ground_net VSS -tie
reconnected total 0 tie highs and 0 tie lows
1
```

Figure 32: Power/Ground Pins and Nets Logical Connections Setup

Upon import of the gate-level netlist, the following messages are generated:

```
Reading ddc file '/home/user/Documents/UGEB2916/anand/dc/lab2/mapped/tlcnetlist.ddc'.
Loaded 1 design.
Current design is 'tlc'.
Current design is 'tlc'.

  Linking design 'tlc'
  Using the following designs and libraries:
  --------------------------------------------------------------------------
  tlc                         /home/user/Documents/UGEB2916/anand/dc/lab2/mapped/tlcnetlist.ddc
  cb13fs120_tsmc_max (library) /home/user/Documents/UGEB2916/anand/dc/lab2/mapped/icc/ref/db/sc_max.db
  cb13io320_tsmc_max (library) /home/user/Documents/UGEB2916/anand/dc/lab2/mapped/icc/ref/db/io_max.db
  ram16x128_max (library)     /home/user/Documents/UGEB2916/anand/dc/lab2/mapped/icc/ref/db/ram16x128_max.d
b

Info: Creating auto CEL.
Preparing data for query...................
Information: Performing CEL netlist consistency check. (MWDC-118)
Information: CEL consistency check PASSED. (MWDC-119)
Information: Saved design named tlc. (UIG-5)
Preparing data for query...................
1
```

Figure 33: Gate-level Netlist Import

In the layout view, the cells of the gate-level netlist imported can be observed as shown in the following figure:

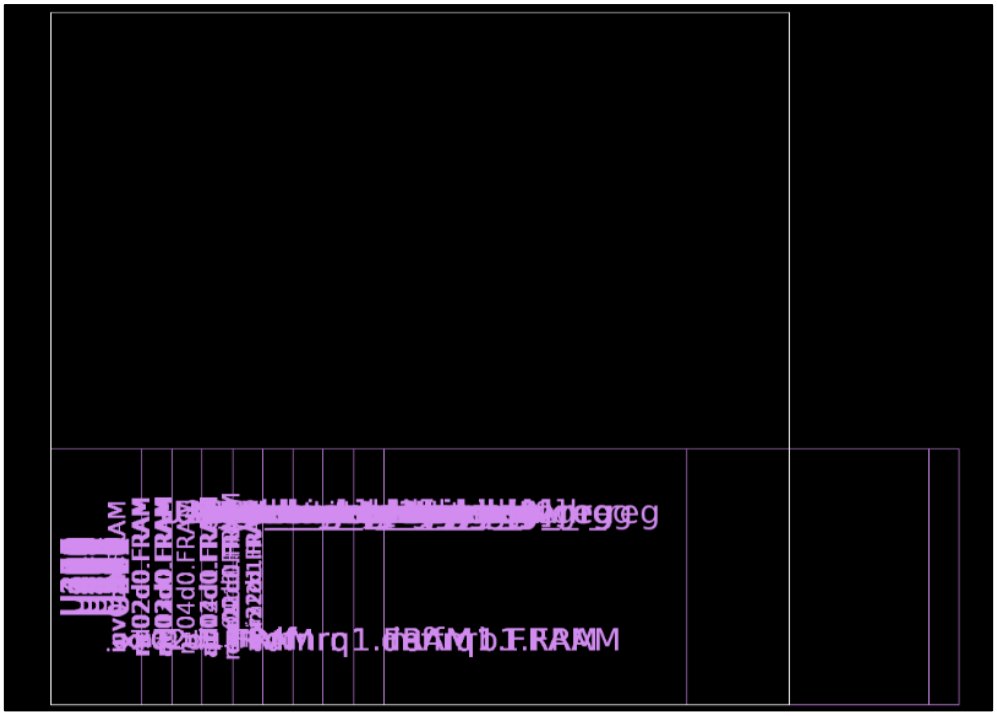


Figure 34: Layout View of Gate-Level Netlist Cells

Figure 35: Floorplan Initialisation

For the floorplan initialization, the core size is set to 1 square unit with a core utilization of 35%. After the floorplan initialization is completed, the layout in Figure 33 is shown.

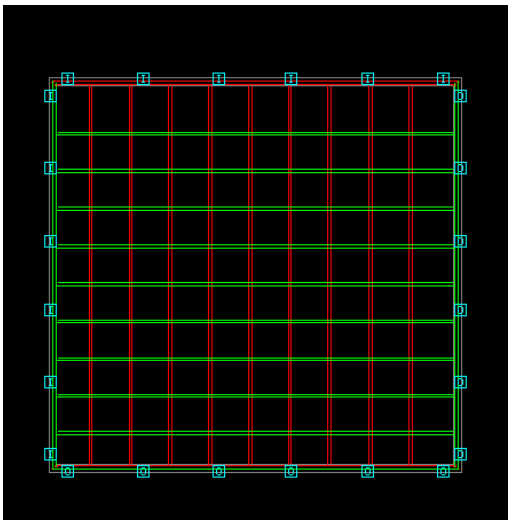


Figure 36: Core with Power Rings and Power Straps

Power rings is then created with metal layer 3 for the top and bottom rings and metal layer 4 for the left and right rings. Horizontal and vertical power straps are created using metal layer 4 and 3 respectively, both with groups of 10 and steps of 5.4. The structure in Figure 34 is shown.

Upon performing cell placements, the cells are placed within the structure created and the following layout is obtained:

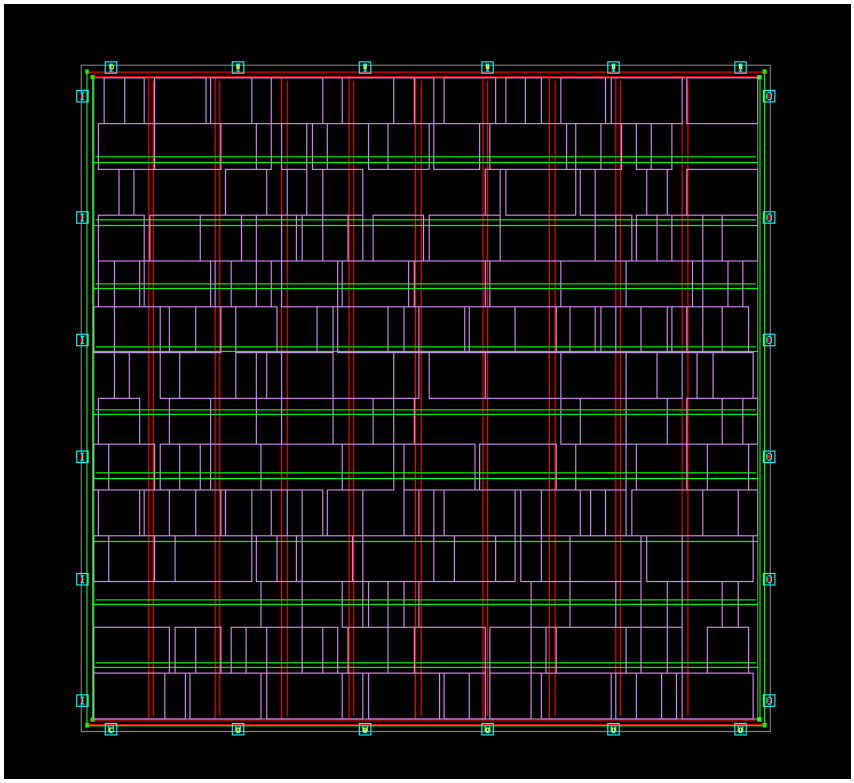


Figure 37: Layout View of the Design after Cell Placement

```
  Placement Optimization (Stage 1)
  --------------------------------

[begin initializing data for legality checker]

Initializing Data Structure ...
INFO: legalizer_via_spacing_check_mode 0
  Reading technology information ...
    Technology table contains 6 routable metal layers
    This is considered as a 6-metal-layer design
    Reading library information from DB ...
  Reading misc information ...
    array <unit> has 0 vertical and 14 horizontal rows
    GRC ref loc X corrected
    GRC ref loc Y corrected
    45 pre-routes for placement blockage/checking
    45 pre-routes for map congestion calculation
  Checking information read in ...
    design style = Horizontal masters, Horizontal rows

  Preprocessing design ...
    splitting rows by natural obstacles ...
... design style 0
... number of base array 1 0
INFO:... use original rows...
[end initializing data for legality checker]

**************************************
  Report : Chip Summary
  Design : tlc
  Version: R-2020.09-SP5
  Date   : Thu Sep 29 06:21:32 2022
**************************************
Std cell utilization: 73.50%  (1348/(1834-0))
(Non-fixed + Fixed)
Std cell utilization: 73.50%  (1348/(1834-0))
(Non-fixed only)
Chip area:            1834     sites, bbox (1.00 1.00 54.71 52.66) um
Std cell area:        1348     sites, (non-fixed:1348   fixed:0)
                      198      cells, (non-fixed:198    fixed:0)
Macro cell area:      0        sites
                      0        cells
Placement blockages:  0        sites, (excluding fixed std cells)
                      0        sites, (include fixed std cells & chimney area)
                      0        sites, (complete p/g net blockages)
Routing blockages:    0        sites, (partial p/g net blockages)
                      0        sites, (routing blockages and signal pre-route)
Lib cell count:       35


Avg. std cell width:  3.36 um
Site array:           unit     (width: 0.41 um, height: 3.69 um, rows: 14)
Physical DB scale:    1000 db_unit = 1 um


**************************************
  Report : pnet options
  Design : tlc
  Version: R-2020.09-SP5
  Date   : Thu Sep 29 06:21:32 2022
**************************************


------------------------------------------------------------------------
Layer      Blockage   Min_width   Min_height   Via_additive     Density
------------------------------------------------------------------------
METAL      none          ---         ---       via additive      ---
METAL2     none          ---         ---       via additive      ---
METAL3     none          ---         ---       via additive      ---
METAL4     none          ---         ---       via additive      ---
METAL5     none          ---         ---       via additive      ---
METAL6     none          ---         ---       via additive      ---

**************************************
  Report : Legalize Displacement
  Design : tlc
  Version: R-2020.09-SP5
  Date   : Thu Sep 29 06:21:32 2022
**************************************

No cell displacement.


  Placement Optimization Complete
  -------------------------------
```

Figure 38: Report Generated during Cell Placement

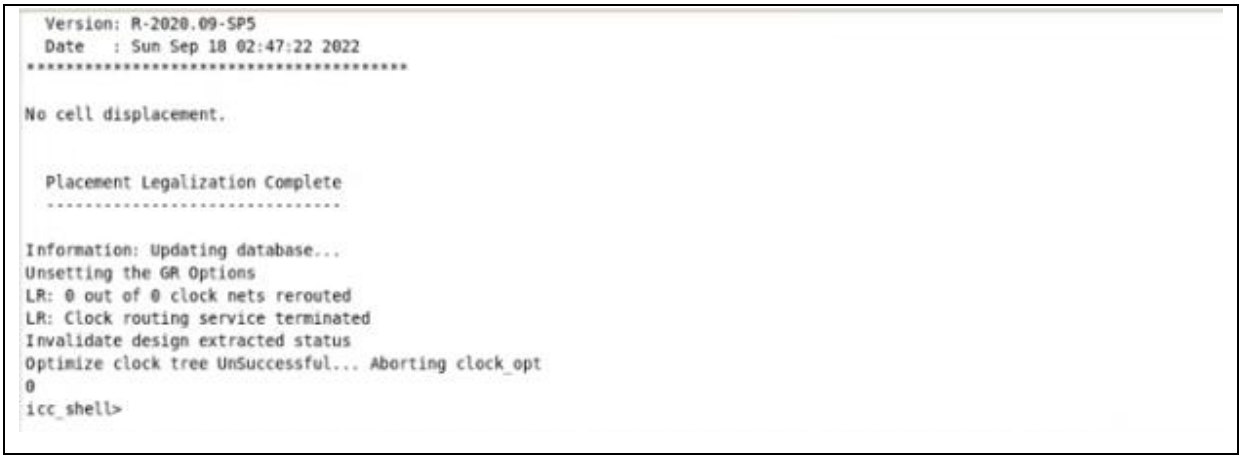


Figure 39: Error Message Generated for Clock Tree Synthesis

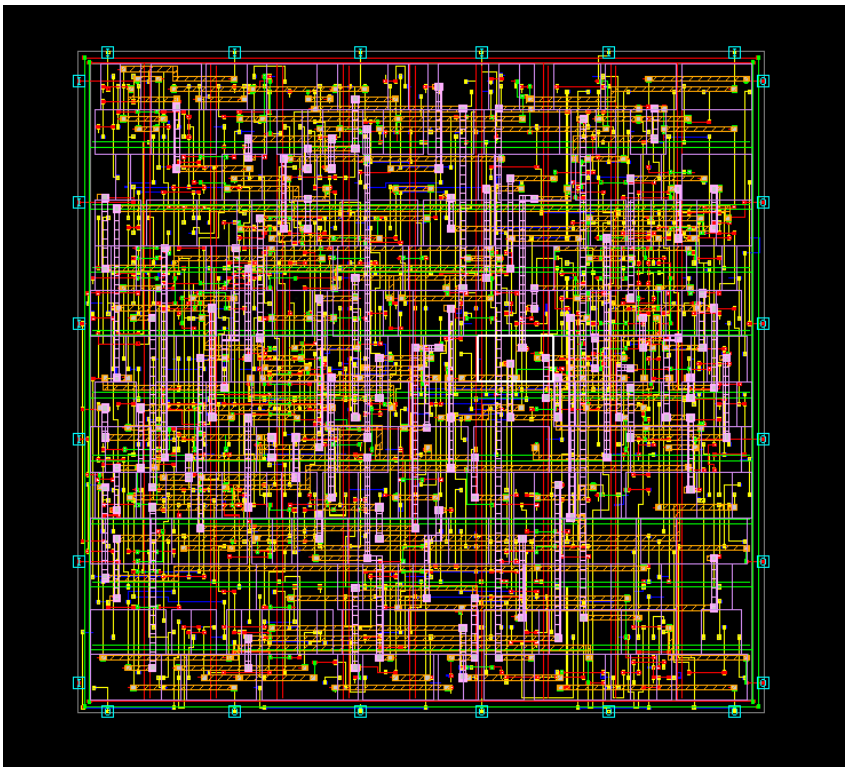
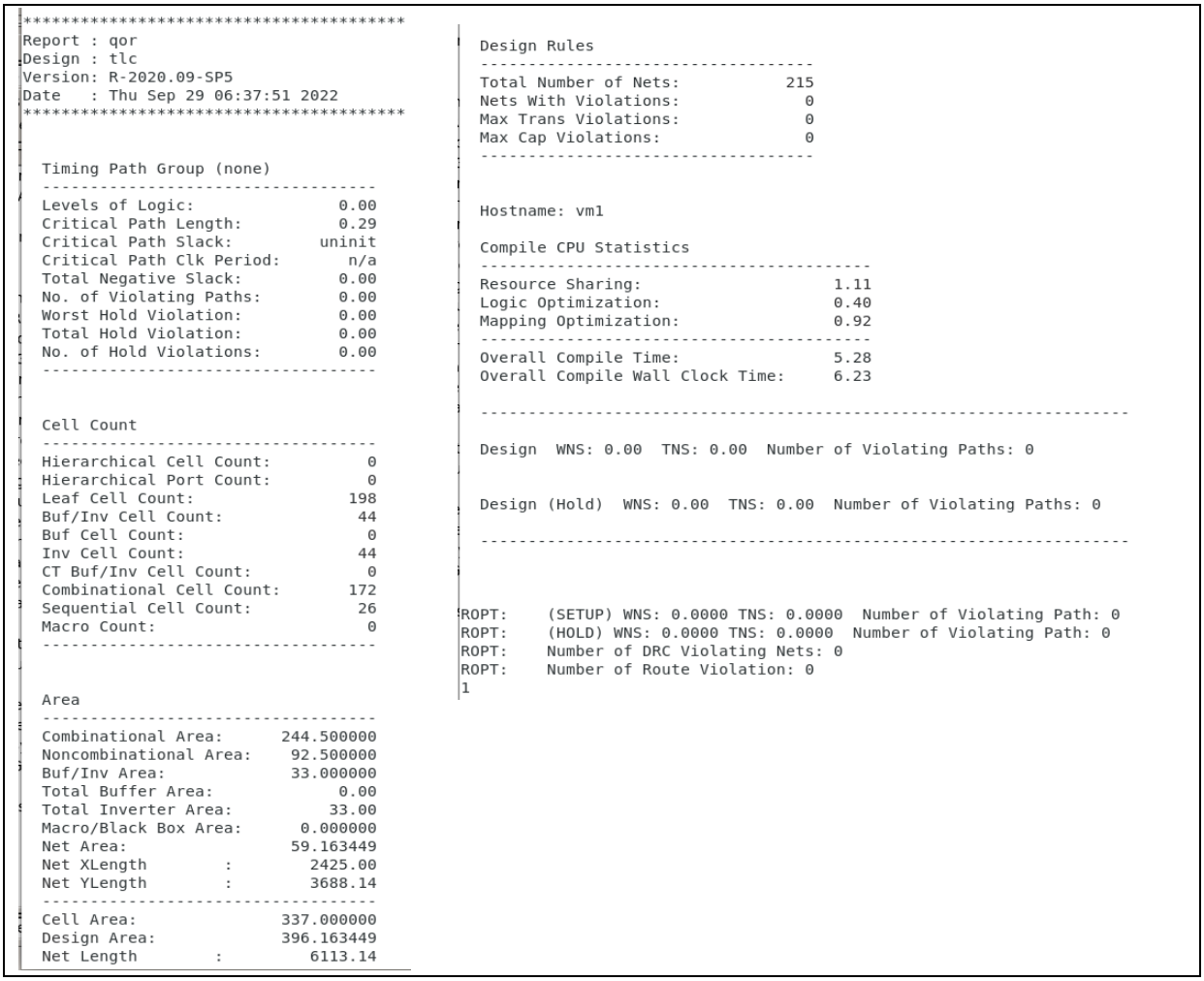


Figure 40: Layout View of the Design after Routing



```
****************************************
Report : qor
Design : tlc
Version: R-2020.09-SP5
Date   : Thu Sep 29 06:37:51 2022
****************************************


  Timing Path Group (none)
  -----------------------------------
  Levels of Logic:               0.00
  Critical Path Length:          0.29
  Critical Path Slack:         uninit
  Critical Path Clk Period:       n/a
  Total Negative Slack:          0.00
  No. of Violating Paths:        0.00
  Worst Hold Violation:          0.00
  Total Hold Violation:          0.00
  No. of Hold Violations:        0.00
  -----------------------------------


  Cell Count
  -----------------------------------
  Hierarchical Cell Count:           0
  Hierarchical Port Count:           0
  Leaf Cell Count:                 198
  Buf/Inv Cell Count:               44
  Buf Cell Count:                    0
  Inv Cell Count:                   44
  CT Buf/Inv Cell Count:             0
  Combinational Cell Count:        172
  Sequential Cell Count:            26
  Macro Count:                       0
  -----------------------------------


  Area
  -----------------------------------
  Combinational Area:      244.500000
  Noncombinational Area:    92.500000
  Buf/Inv Area:             33.000000
  Total Buffer Area:             0.00
  Total Inverter Area:          33.00
  Macro/Black Box Area:      0.000000
  Net Area:                 59.163449
  Net XLength        :        2425.00
  Net YLength        :        3688.14
  -----------------------------------
  Cell Area:               337.000000
  Design Area:             396.163449
  Net Length        :         6113.14


  Design Rules
  -----------------------------------
  Total Number of Nets:           215
  Nets With Violations:             0
  Max Trans Violations:             0
  Max Cap Violations:               0
  -----------------------------------


  Hostname: vm1

  Compile CPU Statistics
  ----------------------------------------
  Resource Sharing:                   1.11
  Logic Optimization:                 0.40
  Mapping Optimization:               0.92
  ----------------------------------------
  Overall Compile Time:               5.28
  Overall Compile Wall Clock Time:    6.23

  --------------------------------------------------------------------

  Design  WNS: 0.00  TNS: 0.00  Number of Violating Paths: 0


  Design (Hold)  WNS: 0.00  TNS: 0.00  Number of Violating Paths: 0

  --------------------------------------------------------------------



ROPT:    (SETUP) WNS: 0.0000 TNS: 0.0000  Number of Violating Path: 0
ROPT:    (HOLD) WNS: 0.0000 TNS: 0.0000  Number of Violating Path: 0
ROPT:    Number of DRC Violating Nets: 0
ROPT:    Number of Route Violation: 0
1
```

Figure 41: Report generated after Routing

```
****************************************
Report : timing
         -path full
         -delay max
         -max_paths 1
Design : tlc
Version: R-2020.09-SP5
Date   : Thu Sep 29 06:49:04 2022
****************************************

 * Some/all delay information is back-annotated.

Operating Conditions: cb13fs120_tsmc_max   Library: cb13fs120_tsmc_max
         Parasitic source    : LPE
         Parasitic mode      : RealRC
         Extraction mode     : MIN_MAX
         Extraction derating : 125/125/125

Information: Percent of Arnoldi-based delays =  8.94%

  Startpoint: n_green_reg
              (positive level-sensitive latch)
  Endpoint: n_green (output port)
  Path Group: (none)
  Path Type: max

  Point                                    Incr       Path
  -----------------------------------------------------------
  n_green_reg/E (lanhq1)                   0.00       0.00 r
  n_green_reg/Q (lanhq1)                   0.29       0.29 f
  n_green (out)                            0.00 &     0.29 f
  data arrival time                                   0.29
  -----------------------------------------------------------
  (Path is unconstrained)


1
```

Figure 42: Timing Report of the Final Layout

```
****************************** P&R Summary *********************************
Date : Fri Sep 30 08:09:10 2022
Machine Host Name: vm1
Working Directory: /home/user/Documents/UGEB2916/anand/icc/lab1_data_setup
Library Name:      tlc
Cell Name:         tlc.CEL;1
Design Statistics:
    Number of Module Cells:        198
    Number of Pins:                1077
    Number of IO Pins:             24
    Number of Nets:                218
    Average Pins Per Net (Signal): 3.26389

Chip Utilization:
    Total Std Cell Area:           2039.39
    Core Size:     width 53.71, height 51.66; area 2774.66
    Chip Size:     width 55.71, height 53.66; area 2989.40
    Std cells utilization:         73.50%
    Cell/Core Ratio:               73.50%
    Cell/Chip Ratio:               68.22%
    Number of Cell Rows:            14

Master Instantiation:
```

Figure 43: Physical Design Report of the Final Layout

```
Library(s) Used:

    cb13fs120_tsmc_max (File: /home/user/Documents/UGEB2916/anand/icc/ref/db/sc_max.db)


Operating Conditions: cb13fs120_tsmc_max   Library: cb13fs120_tsmc_max
Wire Load Model Mode: enclosed

Design        Wire Load Model            Library
------------------------------------------------
tlc                   8000              cb13fs120_tsmc_max


Global Operating Voltage = 1.08
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1pW


  Cell Internal Power  =  17.8307 uW   (72%)
  Net Switching Power  =   7.0299 uW   (28%)
                         ---------
Total Dynamic Power    =  24.8606 uW  (100%)

Cell Leakage Power     =   1.6427 uW

Information: report_power power group summary does not include estimated clock tree power. (PWR-789)

                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )  Attrs
--------------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000            0.0000             0.0000  (   0.00%)
memory             0.0000           0.0000            0.0000             0.0000  (   0.00%)
black_box          0.0000           0.0000            0.0000             0.0000  (   0.00%)
clock_network      0.0000           0.0000            0.0000             0.0000  (   0.00%)
register           0.0000           0.0000            0.0000             0.0000  (   0.00%)
sequential     2.7816e-03        9.4572e-04        5.9261e+05         4.3199e-03  (  16.30%)
combinational  1.5049e-02        6.0842e-03        1.0501e+06         2.2183e-02  (  83.70%)
--------------------------------------------------------------------------------------------------
Total          1.7831e-02 mW     7.0299e-03 mW     1.6427e+06 pW      2.6503e-02 mW
1
```

Figure 44: Power Report of the Final Layout

Discussion

Verilog Coding and VCS Simulation

The information provided for the traffic light controller operation shown below has been used as a basis for the traffic light controller designed:

Table 2: State Transition Table for Cross-junction Traffic Light Controller.

Present State	Next State	Condition
W_Green	W_Green	Reset
W_Green	W_Yellow	After i_g2y_timer
W_Yellow	S_Green	After i_y2r_timer
S_Green	S_Yellow	After i_g2y_timer
S_Yellow	E_Green	After i_y2r_timer
E_Green	E_Yellow	After i_g2y_timer
E_Yellow	N_Green	After i_y2r_timer
N_Green	N_Yellow	After i_g2y_timer
N_Yellow	W_Green	After i_y2r_timer

With the state transition as the basis for the traffic light controller, vehicle priority is added as a feature to give priority to the lane with the most vehicles. This way, traffic is at its most efficient.

When vehicle count for all lanes is equal or zero, the traffic follows the default operation with the flow of **West → South → East → North** which can be seen from **Figure 7** to **Figure 9**.

Based on the bits on **Figure 10**, West Lane has the most car among the four lanes. The sequence **West → South → East → North** is not followed. Instead, West is given priority to turn green next. The traffic light system will go from South Green to West Green by skipping the East Lane. This feature will provide more efficiency towards traffic control.

With all the functionalities verified, the RTL design is then sent for design compilation to be converted into a gate-level netlist.

Logic Synthesis with Design Compiler

Upon importing the RTL design onto the graphical user interface of design vision, the symbol view of the traffic light controller is shown in **Figure 19** and the schematic view is shown in **Figure 20**. When the schematic of the design is zoomed in, the generic gates shown in **Figure 22** are represented by GTECH format. Upon technology mapping, Design Compiler **maps the linked library technologies** onto the design and performs **slight optimization**. Even without design constraint script loaded, a **slight area optimization** can be observed in **Figure 27** whereby the area of the design is slightly reduced from 337.8-unit area to 337.0-unit area. This optimization is a result from **removing unconnected, floating and redundant nets in the schematic, leading to a reduced net interconnect area and a reduction of the total chip area.** With the removal of redundant nets, the **structure of the schematic is then changed** as observed in **Figure 21**.

The **technology mapping implemented the generic gates** shown in **Figure 22** represented in GTECH format with the **existing technologies in the library** as shown by the names of logic gates shown in **Figure 23**. GTECH circuit is the direct product of the implemented RTL design in which it is a technology independent Boolean gate. The arithmetic and relational operators in the RTL design are recognized and translated into generic gates with no logic in it. On compilation, the circuit is optimized based on the design constraint and mapped onto the linked technology library. The mapping process maps cells from the technology independent netlist (GTECH) to cells in the library specified by the technology library, resulting in the change in names of the logic gates from GTECH format to specific format.

As the traffic light controller design only has a main module with no sub-modules implemented, no re-partitioning can be performed to further improve the timing and area of the design. Upon completion of logic synthesis and verification, the gate-level netlist is then sent to IC compiler for physical layout process of the gate-level netlist.

Physical Layout with IC Compiler

In the **floorplanning** process, the core utilization, which is the percentage of core area used for standard cell placement has been set to 73%. The tool then determines the best core utilization whilst maintaining a close approximation to the initial value set. As such, a floorplan with an actual core utilization of 73.50% is used and shown in **Figure 33**. Following the creation of the floorplan, **power rings and power straps have been successfully created** as shown in **Figure 34** before the **cell placement** algorithm is executed to place the standard cells utilized for the design onto the floorplan as shown in **Figure 35**. As no design constraint has been included for the design compilation as well as IC compilation, when **clock tree synthesis** is executed, an **error message is generated** as shown in **Figure 37**, indicating the process is unsuccessful. With no timing constraint and clock information, there is no need for clock tree synthesis. Lastly, **routing of the nets** is performed and the final layout is then produced as shown in **Figure 38**. As no design constraint has been included in the design flow, the path is unconstrained and there will not be any timing violations as shown in **Figure 40**. In the physical design statistic, the final chip size of 2989.40-unit area is reported which is comparatively larger than the design area of 396.16-unit area. The larger chip size is suspected to be due to the high number of I/O ports, resulting in the requirement of more space for I/O power pads. The **chip size** can be **further decreased by concentrating the outputs**. Besides, it is also noted a total standard cell area of 2039.39-unit area have been utilized. The increase from 396.16-unit area to 2989.40-unit area is due to the core utilization of 73.50% is based on the final core size of 2989.40-unit area.

Leakage power is defined as the power consumed by the reverse biased diodes, sub-threshold currents, direct tunnelling, as well as gate oxide tunnelling in a CMOS transistor. These leakages can be reduced through techniques such as multi-threshold (MVT) CMOS design, transistor sizing, power gating, clock gating, or dynamic voltage and frequency scaling (DVFS) (Pedram, n.d.). Since the leakage power of the traffic light controller design is relatively small, which is **1.6427 uW**, it would only have a **minimal impact on the chip's power consumption and its performance**.

Conclusion

In a conclusion, the objectives of the assignment have been achieved. The Verilog coding of both the traffic light controller and its testbench have been successfully coded, compiled and simulated using Synopsys VCS. Functionality verification has been performed using DVE to observe and analyze the testbench signals. Besides, logic synthesis has also been successfully performed on the RTL design, producing a technology-mapped gate-level netlist. Analysis of the gate-level netlist timing and area report has also been performed. The gate-level netlist has an area of 396.16-unit area and does not violate any timing constraint as no design constraint has been sourced for the design. Physical layout has also been successfully carried out. The Verilog file format of the gate-level netlist has been successfully implemented into a physical layout through floorplanning, cell placement and routing processes. The final layout generated has no timing violations and has a chip size of 2989.40 unit-area. As the traffic light controller has been successfully designed from the behavioural description provided into a RTL coding which is then converted into a gate-level netlist and lastly into a physical layout, the assignment is said to have been completed.

Through this assignment, knowledge on IC design flow has been reinforced with more hands-on experience with industrial standard tools albeit the difference in complexity of the design and procedures. IC design is extensive and requires expertise in multiple areas as well as great teamwork among team members and great cooperation and communication between different teams working on different section of the design flow.

References

Synopsys, 2022. *What is IC Design?.* [Online] Available at: https://www.synopsys.com/glossary/what-is-ic-design.html [Accessed 15 August 2022].

Wikimedia Commons, 2021. *File:PhysicalDesign.png.* [Online] Available at: https://commons.wikimedia.org/wiki/File:PhysicalDesign.png [Accessed 30 August 2022].