Homework 2 : Traffic Light

1. Overview

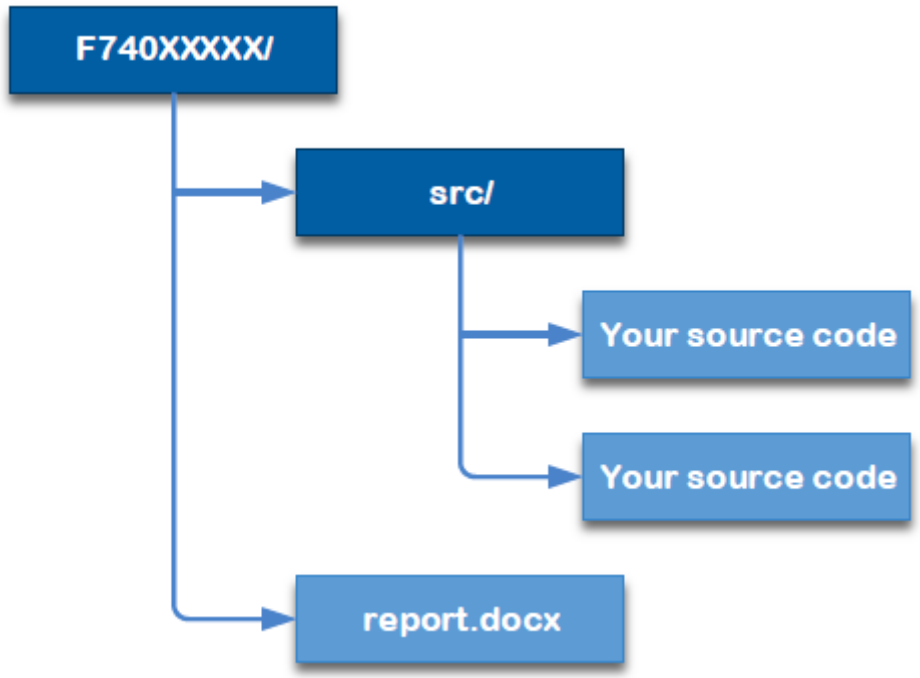
The purpose of this lab is to get you familiar with the Verilog development environment. Please implement the functionality of a traffic light using a finite state machine written in **Verilog**.

2. General rules for deliverables

You need to complete this homework **INDIVIDUALLY**. You can discuss the homework with other students, but you need to do the homework by yourself. You should **NOT copy** anything from someone else, and you should **NOT distribute** your homework to someone else. If you violate any of under these rules, you will get **NEGATIVE scores**, or even fail this course directly.

When submitting your homework, compress all files into a **single zip file**, and upload the compressed file to Moodle.

Please follow the file hierarchy shown in the following figure.



1. F740XXXXX (folder)　(學號，預設是 PXXXXXXX)
2. src (folder)
3. report.docx (Briefly explain you design)

3. Design specification

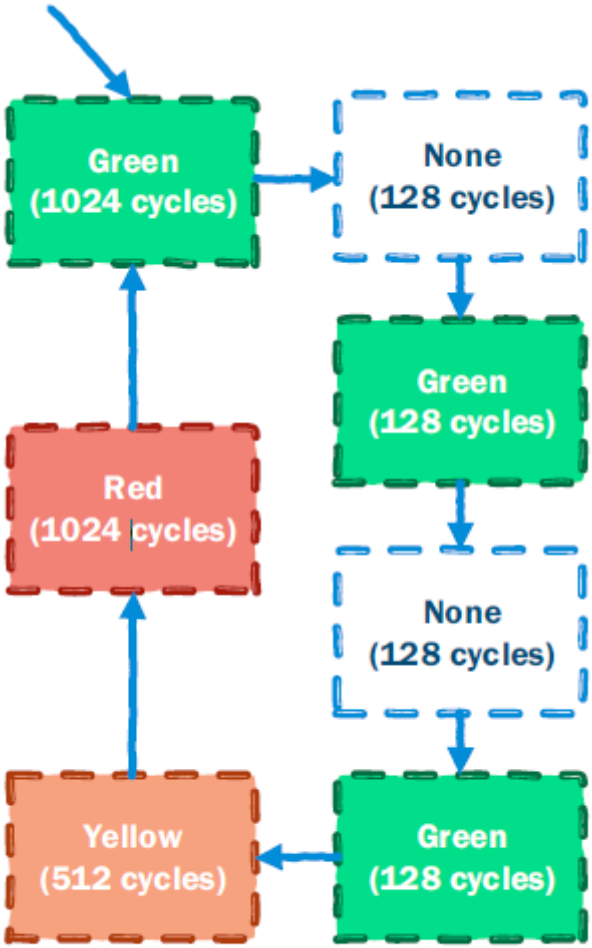


Figure 2. 紅綠燈狀態圖

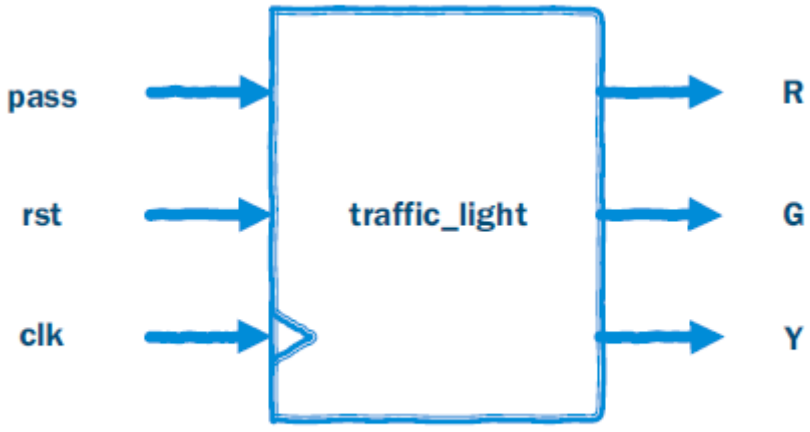


Figure 3. 紅綠燈控制模組的示意圖

作業規則如下：

1. 綠燈維持 1024 個 cycles。（起始狀態）
2. 沒有任何燈號維持 128 個 cycles。
3. 綠燈維持 128 個 cycles。
4. 沒有任何燈號維持 128 個 cycles。

5. 綠燈維持 128 個 cycles。
6. 切換成黃燈維持 512 個 cycles。
7. 再切換成紅燈維持 1024 個 cycles

輸入訊號：(電路為 clock 正緣觸發

1. pass：1bit 訊號，當 pass 為 1 時，若當前狀態非起始狀態之綠燈，強制切換成起始狀態之綠燈第 1 個 cycle，若原本為起始狀態之綠燈則不改變燈號和 cycle。當 pass 為 0 則沒有任何動作。
2. rst：1bit 訊號，非同步正緣時觸發，將燈號狀態設成綠燈第 1 個 cycle。
3. clk：1bit clock 訊號。

輸出訊號：

R：1bit 訊號，代表紅燈的輸出訊號。
G：1bit 訊號，代表綠燈的輸出訊號。
Y：1bit 訊號，代表黃燈的輸出訊號。

4. Project structure

In this course, our work falls in the category of **Cell-based IC design** which consists of the following steps:

1. Verilog coding
    1. Write your Verilog code to meet the design specification.
2. RTL simulation
    1. Verify you design by running **test bench**
    2. The simulation process is done purely with software. No hardware has been inferred yet.
3. Logic synthesis
    1. Synthesize your Verilog design using EDA tools that convert your Verilog code to **actual hardware.**
    2. EDA tools will synthesize your Verilog code into **gate-level netlist**, which is a file specifying the logic gates and the wire connection between them.
4. Pre-layout simulation
    1. In this step we will verify your design after logic synthesis. Make sure your design meets the specification.
    2. Take the **gate-level netlist**, and verify your design again using the Verilog **test bench.**
5. Automatic Placement & Routing (APR)
    1. APR stands for **Automatic Placement & Routing.** You don't have to do this part in this course.

In this lab, you have to verify your design till **pre-layout simulation in our computing environment.**
To help you design your traffic light module in Verilog, we prepared a Makefile-based project that take case of the following process for you:

1. RTL simulation => **make rtl**
2. Logic synthesis => **make syn**
3. Pre-layout simulation => **make pre**

First download the project for this lab from this repo. Unzip the file and go to the directory labeled "**hw1_traffic_light**". There are several folders in this project.

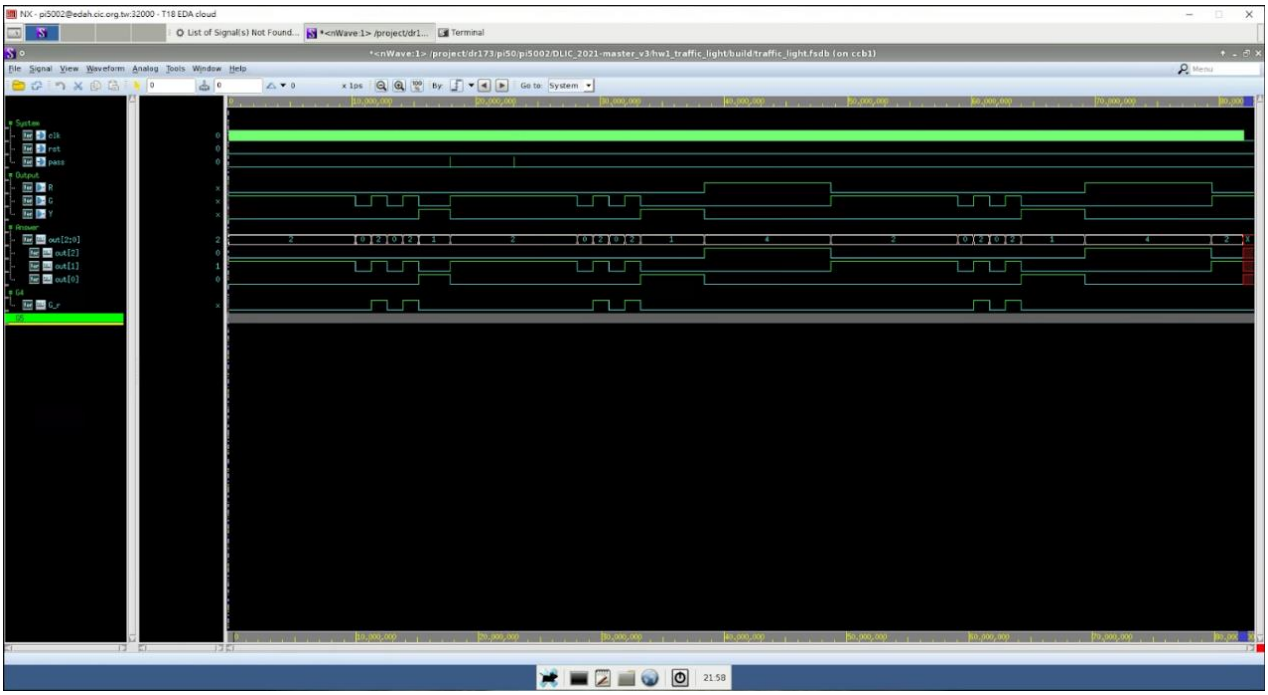
conf	This directory stores the configuration files for different waveform viewing software (i.e. simvision, nWave)
doc	Contains documentation for this project.
script	Contains **synopsys_dc.setup, synthesize.tcl, traffic_light.sdc.** 1. **synopsys_dc.setup** file specifies the cell library related information used in this project. We set the configuration to match the setup of the U18 process. 2. **synthesize.tcl** script tells the **Design Compiler** software how to synthesize your design into hardware. 3. **traffic_light.sdc** is the **design constraint file** specifying the performance requirements of your design. You are allowed to change the cycle time of your design by changing the line **"set cycle xx.x"** in this file.
sim	This directory contains the **Verilog test bench** and **related text files** used for testing your design. Also, there is a soft link **fsa0m_a_generic_core_21.lib.src** file that points to a file in the cell library. This file is necessary for pre-layout simulation please don't delete it.
src	This is where you write your **Verilog code.** There should be an empty Verilog file named "**traffic_light.v**" where you can design your module. You are allowed to add multiple Verilog files here if you know what you are doing. **Make sure you submit ALL Verilog files in this directory to Moodle so we can grade your work.**
Makefile	This Makefile defines multiple targets that help you test your design in a very efficient way. This Makefile contains the following targets: 1. **make rtl: Run RTL simulation** 2. **make nw: Run nWave program** 3. **make syn: Synthesize your design using Design Compiler** 4. **make pre: Run pre-layout simulation**

5. Start coding

The design flow consists of the following steps:

1. Write your Verilog code in **src/** directory using any text editor you like.

2. Compile and run RTL simulation by typing **“make rtl”** in the project directory. You will see the simulation process running and telling you which part of your design does not meet the specification. Type “**make nw**” to launch the **nWave** software that shows the waveform of your design. Then, observe the waveform and make modifications to your code.



3. Make sure you pass the RTL simulation before you proceed to the next step.

```
.............. Done
        Generating native compiled code:
                worklib.ctrl:v <0x39822e7a>
                        streams:   5, words:  3878
                worklib.dp:v <0x6a1d5793>
                        streams:  23, words:  9209
                worklib.traffic_light:v <0x63a94f6a>
                        streams:   1, words:   146
                worklib.traffic_light_tb:v <0x15c32d45>
                        streams:  13, words: 16837
        Building instance specific data structures.
        Loading native compiled code:     ..................... Done
        Design hierarchy summary:
                                Instances  Unique
                Modules:                4       4
                Registers:             18      18
                Scalar wires:          12       -
                Vectored wires:         5       -
                Always blocks:         17      17
                Initial blocks:         5       5
                Cont. assignments:      6      11
                Pseudo assignments:     1       1
                Simulation timescale:  10ps
        Writing initial simulation snapshot: worklib.traffic_light_tb:v
Loading snapshot worklib.traffic_light_tb:v ..................... Done
*Verdi* Loading libsscore_ius152.so
ncsim> source /usr/cad/cadence/INCISIV/cur/tools/inca/files/ncsimrc
ncsim> run
FSDB Dumper for IUS, Release Verdi_O-2018.09, Linux x86_64/64bit, 08/30/2018
(C) 1996 - 2018 by Synopsys, Inc.
*Verdi* FSDB WARNING: The FSDB file already exists. Overwriting the FSDB file may crash the programs that are using this file.
*Verdi* : Create FSDB file 'traffic_light.fsdb'
*Verdi* : Begin traversing the scopes, layer (0).
*Verdi* : End of traversing.
*Verdi* : Begin traversing the MDAs, layer (0).
*Verdi* : Enable +mda and +packedmda dumping.
*Verdi* : End of traversing the MDAs.


        ****************************
        **                        **       /|__/|
        **  Congratulations !!    **      / 0,0  |
        **                        **    /_______ |
        **  Simulation PASS!!     **   /^ ^ ^ ^ \ |
        **                        **  |^ ^ ^ ^ |w|
        ************** ************   \m___m__|_|


Simulation complete via $finish(1) at time 81921 NS + 0
../sim/traffic_light_tb.v:85     $finish;
ncsim> exit
tail: option used in invalid context -- 1
➜  hw1_traffic_light git:(master) make rtl
```

4. Synthesize your design by typing “**make syn**” in the project directory and the **Design Compiler** software will start running. At the end of the process, the **Design Compiler** software will stop and wait for your inputs.

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


  Optimization Complete
  ---------------------
1
#compile_ultra -no_autoungroup -no_boundary_optimization -retime -gate_clock
#compile_ultra -no_autoungroup -no_boundary_optimization -retime
#compile_ultra -incremental
current_design [get_designs ${top}]
Current design is 'traffic_light'.
{traffic_light}
remove_unconnected_ports -blast_buses [get_cells -hierarchical *]
Removing port 'fb_flags[4]' from design 'ctrl'
1
set bus_inference_style {%s[%d]}
%s[%d]
set bus_naming_style {%s[%d]}
%s[%d]
set hdlout_internal_busses true
true
change_names -hierarchy -rule verilog
1
define_name_rules name_rule -allowed {a-z A-Z 0-9 _}   -max_length 255 -type cell
1
define_name_rules name_rule -allowed {a-z A-Z 0-9 _[]} -max_length 255 -type net
1
define_name_rules name_rule -map {{"\\*cell\\*" "cell"}}
1
define_name_rules name_rule -case_insensitive
1
change_names -hierarchy -rules name_rule
1
# Write sdf
write -format ddc -hierarchy -output "${top}_syn.ddc"
Writing ddc file 'traffic_light_syn.ddc'.
1
write_file -format verilog -hierarchy -output ../syn/${top}_syn.v
Writing verilog file '/home/wei/git/DLIC_2021/hw1_traffic_light/syn/traffic_light_syn.v'.
1
write_sdf -version 2.0 -context verilog  ../syn/${top}_syn.sdf
Information: Annotated 'cell' delays are assumed to include load delay. (UID-282)
Information: Writing timing information to file '/home/wei/git/DLIC_2021/hw1_traffic_light/syn/traffic_light_syn.sdf'. (WT-3)
Information: Updating design information... (UID-85)
1
write_sdc -version 2.0 ${top}.sdc
1
report_area > area.log
report_timing > timing.log
report_qor > ${top}_syn.qor
# exit
Warning: Cannot use command line editor for terminal type 'xterm'.  (UI-74)
dcnxt_shell>
```

5. You can type **“report_timing”** to look at your timing information or type “**report_area”** to look at your area information. Or type “**exit**” to leave **Design Compiler.** Make sure the timing of your design is **MET** before you proceed to the next step.

```
Version: Q-2019.12
Date   : Tue Oct 12 11:43:24 2021
****************************************

Operating Conditions: WCCOM   Library: fsa0m_a_generic_core_ss1p62v125c
Wire Load Model Mode: enclosed

  Startpoint: pass (input port clocked by clk)
  Endpoint: ul_dp/cnt_reg_0_
            (rising edge-triggered flip-flop clocked by clk)
  Path Group: clk
  Path Type: max

  Des/Clust/Port     Wire Load Model       Library
  ------------------------------------------------
  traffic_light      G5K                   fsa0m_a_generic_core_ss1p62v125c
  ctrl               enG5K                 fsa0m_a_generic_core_ss1p62v125c
  dp                 enG5K                 fsa0m_a_generic_core_ss1p62v125c

  Point                                    Incr       Path
  -----------------------------------------------------------
  clock clk (rise edge)                    0.00       0.00
  clock network delay (ideal)              0.50       0.50
  input external delay                     5.00       5.50 f
  pass (in)                                0.01       5.51 f
  ul_ctrl/pass (ctrl)                      0.00       5.51 f
  ul_ctrl/U7/O (INV1S)                     0.21       5.72 r
  ul_ctrl/U9/O (OAI112HS)                  0.18       5.90 f
  ul_ctrl/U26/O (ND2)                      0.13       6.02 r
  ul_ctrl/U8/O (ND3)                       0.16       6.19 f
  ul_ctrl/dp_cnt_rst (ctrl)                0.00       6.19 f
  ul_dp/cnt_rst (dp)                       0.00       6.19 f
  ul_dp/U5/O (AOI13HS)                     1.10       7.29 r
  ul_dp/U6/O (OR2)                         0.71       8.00 r
  ul_dp/U14/O (MOAI1S)                     0.25       8.25 f
  ul_dp/cnt_reg_0_/D (QDFFRBN)             0.00       8.25 f
  data arrival time                                   8.25

  clock clk (rise edge)                   10.00      10.00
  clock network delay (ideal)              0.50      10.50
  clock uncertainty                       -0.10      10.40
  ul_dp/cnt_reg_0_/CK (QDFFRBN)            0.00      10.40 r
  library setup time                      -0.15      10.25
  data required time                                 10.25
  -----------------------------------------------------------
  data required time                                 10.25
  data arrival time                                  -8.25
  -----------------------------------------------------------
  slack (MET)                                         2.00


1
dcnxt_shell>
```

```
1
dcnxt_shell> report_area

****************************************
Report : area
Design : traffic_light
Version: Q-2019.12
Date   : Tue Oct 12 11:43:44 2021
****************************************

Library(s) Used:

    fsa0m_a_generic_core_ss1p62v125c (File: /usr/cad/CBDK/CBDK018_UMC_Faraday_v1.0/CIC/SynopsysDC/db/fsa0m_a_generic_core_ss1p62
v125c.db)

Number of ports:                           65
Number of nets:                           200
Number of cells:                          129
Number of combinational cells:            102
Number of sequential cells:                24
Number of macros/black boxes:               0
Number of buf/inv:                         29
Number of references:                       4

Combinational area:               1674.892780
Buf/Inv area:                      193.737598
Noncombinational area:            1428.033607
Macro/Black Box area:                0.000000
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:                  3102.926387
Total area:                 undefined
1
dcnxt_shell>
```

6. You can now find **two files: traffic_light_syn.sdf, traffic_light_syn.v** in the **syn/** directory. These two files will be used in pre-sim later.

7. Type “**make pre**” to run pre-sim. If you see “**Congratulations !! Simulation PASS !!**” then you have successfully completed this lab. If not, you should modify your design until everything is correct.

```
                worklib.traffic_light:v <0x02573b17>
                        streams:   0, words:     0
                worklib.traffic_light_tb:v <0x0112be21>
                        streams:  14, words: 17194
        Building instance specific data structures.
        Loading native compiled code:     ..................... Done
        Design hierarchy summary:
                                 Instances  Unique
                Modules:               131      32
                UDPs:                   26       2
                Primitives:            194       7
                Timing outputs:        138      18
                Registers:              35      16
                Scalar wires:          171       -
                Vectored wires:          1       -
                Always blocks:           8       8
                Initial blocks:          6       6
                Cont. assignments:       1       4
                Pseudo assignments:      1       1
                Timing checks:         209      32
                Interconnect:          301       -
                Delayed tcheck signals:   69      27
                Simulation timescale:    1ps
        Writing initial simulation snapshot: worklib.traffic_light_tb:v
Loading snapshot worklib.traffic_light_tb:v ..................... Done
*Verdi* Loading libsscore_ius152.so
ncsim> source /usr/cad/cadence/INCISIV/cur/tools/inca/files/ncsimrc
ncsim> run
FSDB Dumper for IUS, Release Verdi_0-2018.09, Linux x86_64/64bit, 08/30/2018
(C) 1996 - 2018 by Synopsys, Inc.
*Verdi* FSDB WARNING: The FSDB file already exists. Overwriting the FSDB file may crash the programs that are using this file.
*Verdi* : Create FSDB file 'traffic_light.fsdb'
*Verdi* : Begin traversing the scopes, layer (0).
*Verdi* : End of traversing.
*Verdi* : Begin traversing the MDAs, layer (0).
*Verdi* : Enable +mda and +packedmda dumping.
*Verdi* : End of traversing the MDAs.


        ****************************
        **                        **       /|__/|
        **  Congratulations !!    **      / 0,0  |
        **                        **     /_____   |
        **  Simulation PASS!!     **    /^ ^ ^ \  |
        **                        **   |^ ^ ^ ^ |w|
        ************** ************    \m___m__|_|


Simulation complete via $finish(1) at time 81921 NS + 0
../sim/traffic_light_tb.v:85     $finish;
ncsim> exit
tail: option used in invalid context -- 1
➜  hw1_traffic_light git:(master) make pre
```

6. Homework requirement and Grading

- **50%**
  - **Pass RTL simulation to get 50%**
  - **Pass Pre-layout simulation to get the remaining 50%**
- **50%**
  - **Report**