COSE221: Computer Architecture
Design Lab #3

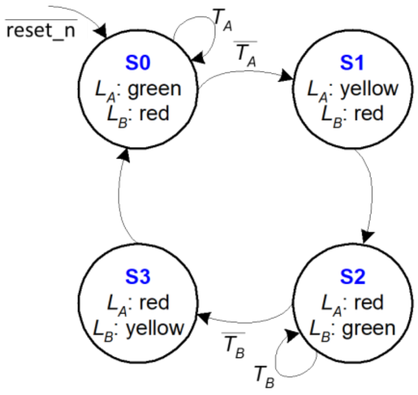
Due: May 22, 2023 (Monday) 11:59pm on Blackboard

Total score: 30 pts

In this lab, you will practice implementing *finite state machine (FSM)* blocks using SystemVerilog. For this lab assignment, we will use the FSM models we covered during the lectures.

1. Simple FSM

Implement an FSM module for the following FSM state transition diagram we covered during the lecture. Unlike the original FSM, the FSM module receives *asynchronous negative* reset. Namely, the FSM state goes to the initial state (i.e. S0) if `reset_n` is 0.

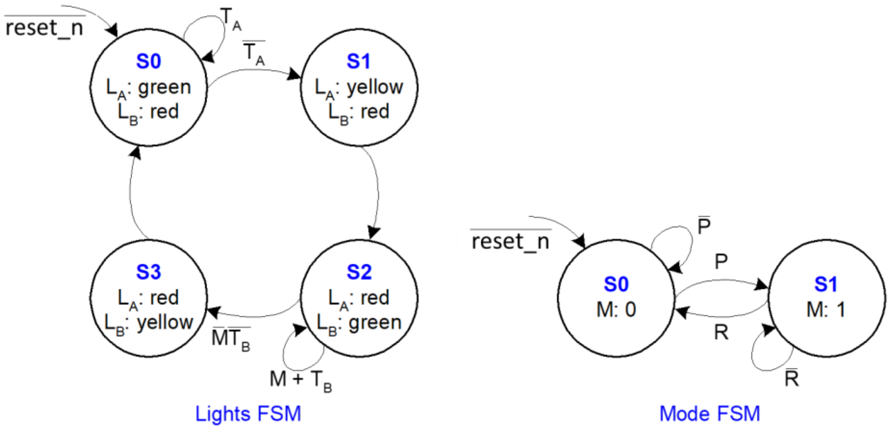


You can compile and execute testbench code as follows.

```
$ iverilog -g2005-sv -o fsm1 tb_fsm1.sv fsm1.sv
$ vvp -v fsm1
```

2. Factored FSM

Design an FSM module that implements the following *factored FSM* we covered during the lecture. Unlike the original design, this FSM module receives *asynchronous negative* reset.



For this FSM design, you are requested to design the module hierarchically. Namely, you need to design the mode FSM module (i.e. `fsm_mode`) and the light FSM module (i.e. `fsm_light`). Then these modules are instantiated in the top module (i.e. `fsm2`). In the top module, you need to define and connect internal nodes between the two FSM modules.

You can compile and execute testbench code as follows.

```
$ iverilog -g2005-sv -o fsm2 tb_fsm2.sv fsm2.sv
$ vvp -v fsm2
```

3. What to do

(a) Complete the provided modules (i.e. `fsm1.sv` and `fsm2.sv`). Simulate each module using the provided testbench. You don't need to modify the testbench modules. Capture the waveform of each design. You need to make the waveforms displayed until the end of simulation. Your waveforms should include all signals in the DUT (design under test) modules. For the factored FSM design, your waveform includes the all internal signals in `fsm_mode` and `fsm_light`. Embed the captured images in the report document.

(b) Compress your PDF file (report), source codes (`fsm1.sv` and `fsm2.sv`), the generated VCD files (`*.vcd`), and output files (`*.out`) in **one zip file**. You must name your zip file as "`FirstName_LastName.zip`". (e.g. `Gildong_Hong.zip` for Gildong Hong) If your submission file does not meet this rule, we will reduce 1 point from your score. 🙁

<NOTE>

SystemVerilog syntax `always_comb`, `always_ff`, and `always_latch` is supported by Icarus Verilog v11.0. Unfortunately, the older versions of Icarus Verilog cannot understand `always_comb`, `always_ff`, and `always_latch`. You can install Icarus Verilog v11.0 from Ubuntu's default repository if you are using Ubuntu newer than version 22.04. If not, you can try one of the following methods.

Method 1 (recommended):

Change always statements in the source codes as follows.

```
      always_comb → always @ (*)
      always_ff → always
```

Method 2:

You can compile the newer version of Icarus Verilog from the source code. Please follow the instructions in https://iverilog.fandom.com/wiki/Installation_Guide#Compiling_on_Linux/Unix. Before installing the newer version of Icarus Verilog, you need to remove the installed Icarus Verilog.

```
$ sudo apt purge iverilog
$ sudo apt update
$ sudo apt install bison, flex, autoconf, gperf
$ mkdir -p workspace
$ cd workspace
$ git clone https://github.com/steveicarus/iverilog.git
$ cd iverilog
$ git checkout --track -b v11-branch origin/v11-branch
$ git pull
$ ./configure
$ make
$ sudo make install
```