RTR532 2018
Problem set #03

Total value - 100 points

Submission deadline – 2018-03-21 23:59

Aim of this problem set – to practice:

1. Design entry and finite state machine
2. Design verification

Variant table

Student	Task 1 PARAMETER
Gotlaufs Roberts	3
Ivanovs Ruslans	5
Kuzminovs Glebs	2
Palamarcuks Dmitrijs	6
Smirnovs Glebs	7
Smeiksts Linards	4
Zuters Karlis	5
"I was not present at first lecture"	6

Starting notes

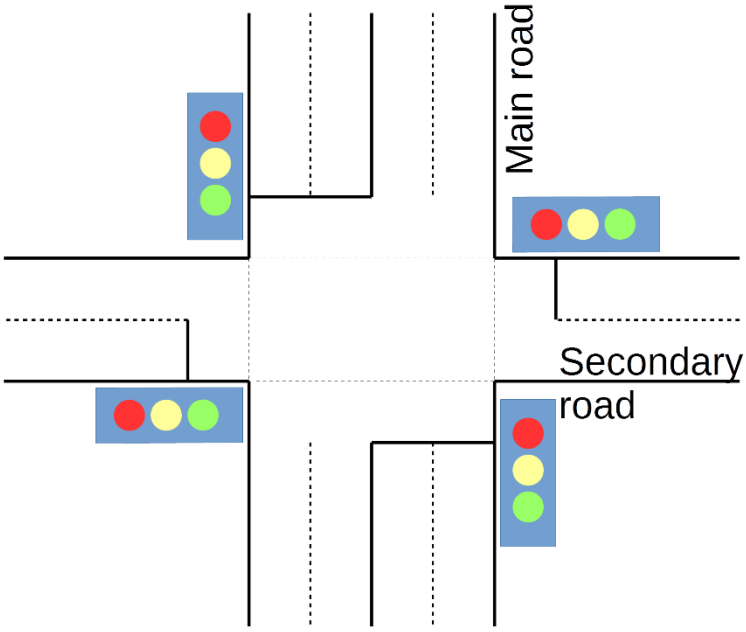
Copy the repo \rtr532-2018\problem_sets\ps03\ contents to your repository (\repo-student\problem_sets\ps03\). Remember to:

- add the new files (.qpf, .qsf and .vhd files) to the repository
- commit early and commit often the changes of your .vhd source to your local repository.
- When finished, don't forget to push your repo to the server.
- No other files, except .qpf, .qsf, .vhd should be added to the repo (as others are not "source files").
- Problem sets will be graded from your repositories, as before.

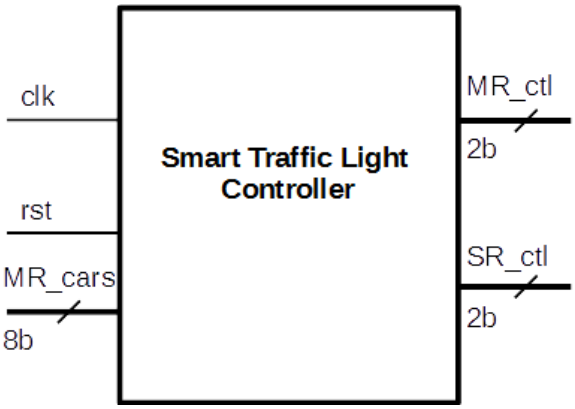
Your task will be to prepare a simple verified FPGA project from scratch. No project template is given in this problem set. You must create a new project, create a new VHDL file, write the state machine logic, then write a testbench and run a functional simulation. Analysis and Synthesis should complete with no errors (otherwise you will not be able to simulate it). Warnings are OK. Although the problem set is divided in two tasks, I suggest you do both tasks simultaneously.

Task 1 – Designing a state machine (70 / 100 points)

> *"I was driving down the Main road in the middle of the night and suddenly – an intersection with red light! I was standing there for 30 seconds, not a single soul was visible anywhere near the intersection… I really hate when I have to stop at red light when there is no traffic on the other road, just because somebody was lazy enough not to install a smart traffic light system. Even blinking yellows were better… God, I hate this…"*
>
> - Quote of a Latvian driver on the 6th July night, 2016.



Consider the intersection drawn above. Your task is to develop a **Smart Traffic Light Controller** VHDL module using finite state machine principle (FSM). Start the task by creating a new project in the **\repo-student\problem_sets\ps03** folder. You can choose your own name for the project and the entity. **Port names must be named exactly as in the figure below and table Port Description**.



Port description

Port name	Direction	Description
Clk	In	1 GHz clock
Rst	In	Active high reset
MR_cars	In	An 8-bit input signal indicating the number of waiting cars at the secondary road traffic light. Interpret as unsigned.
MR_ctl	Out	Main road traffic light controlling signal
SR_ctl	Out	Secondary road traffic light controlling signal

Output light mapping for MR_ctl and SR_ctl signals to be used

Value	Light output
00	No light – traffic light is dark
01	Red light
10	Yellow light
11	Green light

Required functionality

When rst is high, outputs should be dark, no light visible

When rst is low then:

- Minimum main road green period length should be at least 30 nanoseconds. When green period ends:
    - If there are zero cars waiting on the secondary road, then main road starts the green state again
    - If there are less than PARAMETER cars waiting on the secondary road, green period length is extended by 30 nanoseconds and then switches to red (and secondary to green)
    - If there are equals/more than PARAMETER cars waiting on the secondary road, main road switches to red (and secondary to green)
- Secondary road green period length is 10 nanoseconds, after that, main road switches to green (and secondary to red)
- Each transition (red to green and green to red) should go through 3 nanoseconds yellow light. Yellow lights can happen at the same time in both traffic directions.

MR_ctl and SR_ctl both must not be green at the same time to avoid traffic accidents.

PARAMETER – values defined in the variant table. The value should be passed to the state machine module through a Generic port.

Draw a state transition diagram with state names, output statuses in it. You can use any drawing/sketching software (ms paint, libreoffice draw, ms word, online tools, paper and pencil and photo/scan, or other ways to produce the drawing). Add the drawing file to the repository.

Task 2 – Verifying the state machine – (30 / 100 points)

Second task is to verify that your state machine works. Result should be a Quartus project that I can open, click Tools – Run Simulation Tool – RTL Simulation and it launches for a time length where operation of the design can be observed (around ~100-300 nanoseconds)

Testbench should pass the generic PARAMETER to the smart traffic light controller module.

Generating the stimuli

Clk - Generate the 1GHz clock input

Rst – generate both high and low reset to ensure correct start up, per **Required functionality**

MR_cars – generate multiple values to simulate real world traffic situation (e.g. at night some cars arrive once per 60nanoseconds, in day – more cars) and verify that smart traffic light controller is working correctly.