RTR532 2018
Problem set #1

Total value - 100 points

Submission deadline – 2018-02-21 23:59

Aim of this problem set – getting started to rtr532

1. set up and verify that Git is working for you
2. Improve understanding of the look-up table element
3. Practice writing simple VHDL code

Starting notes

1. Copy the repo \rtr532-2018\problem_sets\ps01\ contents to your repository (\rtr532-2018-surname\problem_sets\ps01\)
2. Add the copied contents to (\rtr532-2018-surname\problem_sets\ps01\ by git command git add * , or by right clicking in the repository and selecting TortoiseGit -> Add -> select files
3. Remember to commit changes (of .vhd files) to your local repository, before pushing it to the master remote repository.
4. **Problem sets will be graded from your repositories.**

Submitting the problem set for evaluation

a) Make sure the required files of Tasks 1-2 are added to the repository.
https://tortoisegit.org/docs/tortoisegit/tgit-dug-add.html

b) Commit the changes to the repository (right click in repository folder, Git Commit -> master), include a short description. Check "Set Author Date, Set Author" checkboxes, then click Commit.
https://tortoisegit.org/docs/tortoisegit/tgit-dug-commit.html

c) Click Push after successful commit to push the changes to the remote Gitlab server (click ok on the window that appears). https://tortoisegit.org/docs/tortoisegit/tgit-dug-push.html

d) To make sure your result has been pushed to Gitlab server, log in to https://epkgit.rtu.lv/ open the repository and check last for the last commit date and contents.

Task 1 – The look-up table (40 / 100 points)

A LUT, which stands for Look-up Table, in general terms is basically a table that determines what the output is for any given input(s). In the context of combinational logic, it is the truth table. This truth table effectively defines how your combinatorial logic behaves.

- *Anon, internet*

Draw a 3-input look-up table Y = f(A,B,C) schematic. The LUT schematic must consist only of:

1. A number of single-bit memory elements; and
2. A number of 2:1 MUX.

Fill in the memory slots according to the variant and truth table.

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 a schematic symbol of look-up table).

Example with Y = f(A,B) - http://beta.ivc.no/blog/wp-content/uploads/2011/03/fpga-lut.png (example uses x1, x2 as inputs, rather than A, B and f1 as output, rather than Y)

Table 1 - truth table for LUT's

A	B	C	Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8
0	0	0	0	1	1	0	0	1	0	1
0	0	1	0	1	0	1	1	1	1	0
0	1	0	1	0	1	0	0	1	0	0
0	1	1	1	0	0	1	0	0	0	0
1	0	0	0	1	1	0	1	0	1	1
1	0	1	1	0	0	1	1	0	1	0
1	1	0	0	1	1	0	0	1	1	0
1	1	1	1	0	0	1	1	1	1	1

Table 2 - Variant table

Student	**Truth table option**
Gotlaufs Roberts	Y1
Ivanovs Ruslans	Y2
Kuzminovs Glebs	Y3
Palamarcuks Dmitrijs	Y4
Smirnovs Glebs	Y5
Smeiksts Linards	Y6
Zuters Karlis	Y7
“I was not present at first lecture”	Y8

Place the produced drawing/image file in your repository \rtr532-2018-surname\problem_sets\ps01 and add it to your repository (“git add *” or by right clicking on the file, select Tortoise Git -> Add -> Ok.

Task 2 – Your first VHDL code (60 / 100 points)

Modify the ps01q.vhd to implement the required functionality (see below). You can do the modifications using Quartus II (open project ps01q with Quartus), or by means of any other text editor software, such as Notepad++.

Required functionality:

Entity definition:

Entity name - ps01q

Three entity inputs (A, B, C) of type std_logic

Two entity outputs (Y1, Y2) of type std_logic

Architecture (before "begin" keyword)

One inner-module signal [signal] of type std_logic (choose your own name for it)

Architecture (after "begin" keyword)

Implement three logic gates and connect then as defined below:

Gate one

- inputs - A, B
- output - [signal]

Gate two

- input - C, [signal]
- output - Y2

Gate three

- input - A, B
- output - Y1

Table 3 - Task 2 variant table (gates)

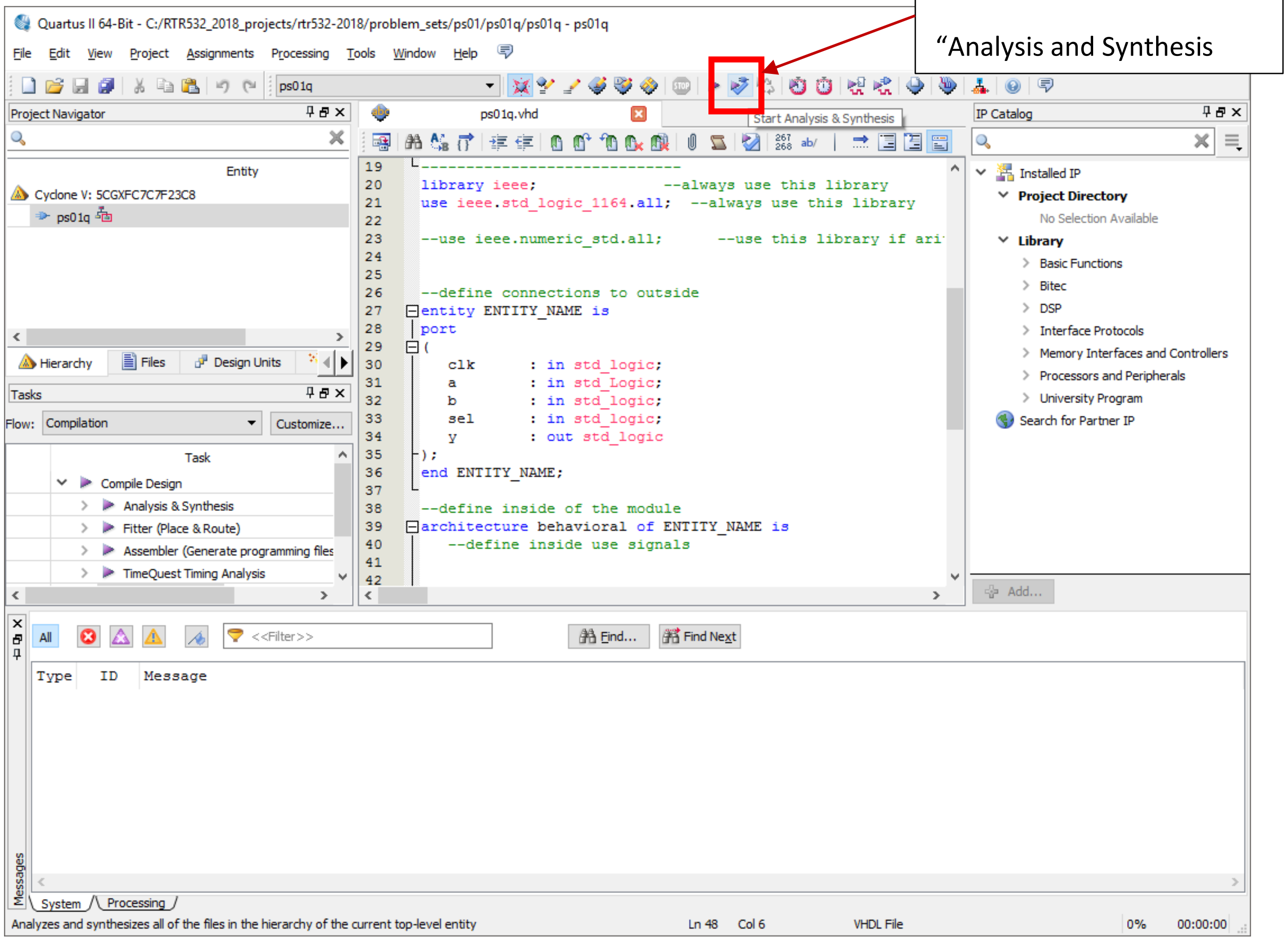
Student	Gate one	Gate two	Gate three
Gotlaufs Roberts	or	xor	Nor
Ivanovs Ruslans	and	nand	Xnor
Kuzminovs Glebs	and	or	Nand
Palamarcuks Dmitrijs	nor	and	or
Smirnovs Glebs	and	or	Nor
Smeiksts Linards	xor	nor	Nand
Zuters Karlis	and	Or	and
"I was not present at first lecture"	xor	xnor	not

Task 2 step by step

1. (optional) Draw schematic of logic gates, input output names, inner signal names so you are 100% sure what you are writing in the code.
2. Fill in the header in .vhd file
3. Fill in the entity definition part in .vhd file
4. Define required inner-module signals in .vhd file
5. Write behavioral code in .vhd file
6. Save the file, Click analysis and Synthesis (screenshot below), correct errors
7. Commit changes to repository (cmd: git commit -m “commit message”)
8. Repeat 1-7 till you are satisfied with the result.
9. Push the branch to central repository (epkgit.rtu.lv) so Kriss can grade the result. (cmd: git origin push head)
10. Enjoy nice grades when Kriss evaluates the result.

Click to run

“Analysis and Synthesis


Quartus II 64-Bit - C:/RTR532_2018_projects/rtr532-2018/problem_sets/ps01/ps01q/ps01q - ps01q
File Edit View Project Assignments Processing Tools Window Help
ps01q
Start Analysis & Synthesis
Project Navigator
Entity
Cyclone V: 5CGXFC7C7F23C8
ps01q
Hierarchy Files Design Units
Tasks
Flow: Compilation
Customize...
Task
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate programming files
TimeQuest Timing Analysis
ps01q.vhd
IP Catalog
Installed IP
Project Directory
No Selection Available
Library
Basic Functions
Bitec
DSP
Interface Protocols
Memory Interfaces and Controllers
Processors and Peripherals
University Program
Search for Partner IP
Add...
All
<<Filter>>
Find...
Find Next
Type ID Message
Messages
System
Processing
Analyzes and synthesizes all of the files in the hierarchy of the current top-level entity
Ln 48 Col 6 VHDL File 0% 00:00:00

```
-----------------------------
library ieee;              --always use this library
use ieee.std_logic_1164.all;  --always use this library

--use ieee.numeric_std.all;      --use this library if ari


--define connections to outside
entity ENTITY_NAME is
port
(
    clk      : in std_logic;
    a        : in std_logic;
    b        : in std_logic;
    sel      : in std_logic;
    y        : out std_logic
);
end ENTITY_NAME;

--define inside of the module
architecture behavioral of ENTITY_NAME is
    --define inside use signals
```

Grading for task 2.

- 10 points - for filled informative header part of the vhdl file

- 10 points - for entity definition according to required functionality.

- 10 points - for architecture inner-module signal definition

- 10 points - for architecture inner-module behavioral code

- 20 points - Analysis and Synthesis process completes successfully, without errors.