Project 1 (6%)

ECGR 2181 – Spring 2020

Due: Friday, March 20

For both parts of this project, choose 2 partners of your own choice **(teams of 3 only)**. You are expected to put forth an equal amount of effort to complete this assignment. You will get a chance to review yourself and your teammate using the form available on Canvas. Each team is expected to submit one PDF report for both parts of this project, one .zip archive of source code, and three (one per team member) self- and peer evaluations. Peer evaluations should be submitted individually as they are to remain anonymous and will affect your grade negatively if not submitted. For both parts of this project, you should include a write-up of your approach (the basic design process, anything you had troubles with, an explanation of anything that doesn't work, and any results).

Part 1:

Assignment Overview

In this assignment you will use Vivado 2019.2 or any previous version to write and simulate some simple VHDL files. There are two main tasks in this assignment. The first being a simple 1-bit adder and the second being a slightly more complicated circuit. Using the techniques learned in class and from the tutorial, complete both tasks. You'll also need to turn in a single PDF to Canvas (instructions at the end of the file).

Getting started

1. Create a directory to store all of the assignment's files.
2. Open Vivado 2019.2 and create a new project called **computer_assignment_1**.
3. It will be an RTL project
4. No sources to input. Although if the target and simulation languages aren't set to VHDL, change them to VHDL.
5. No existing IP.
6. No constraints file.
7. The Basys3 uses an **XC7A35T-CPG236C** FPGA
   - Family: Artix-7
   - Package: CPG236
   - Speed: -1
   - Choose the part with 41600 FlipFlops

Task 1

This task is to implement the functions:

Inputs: A, B, Cin

Outputs: Sum, Cout

$$\boldsymbol{Sum = A \oplus B \oplus Cin}$$

$$\boldsymbol{Cout = AB + ACin + BCin}$$

in task1.vhd. It may seem trivial but it’s a good starting point. Make sure to read the instructions before starting.

1. Create the task1.vhd file:
    a. Under “Flow Navigator” click “Add sources.”
    b. Select “Create or add design sources.”
    c. Click “Create File”, enter the file name “task1”, hit “OK”, and then click “Finish.”
2. A window will pop-up to define the port connections. All inputs and outputs above will be 1 bit in length (**`std_logic`**).
3. Write the code above in the architecture portion of your entity.
4. Make sure it’s syntax error free.
5. Create task1_tb.vhd file:
    a. Under “Flow Navigator” click “Add sources.”
    b. Select “Create or add simulation sources.”
    c. Create task1_tb.vhd. Leave the entity empty, i.e. no inputs or outputs. Click “OK” if asked whether to use the given values (empty entity).
    d. Make sure simulation set is “sim_1.”
    e. Click “Finish”.
6. Select the right simulation set:
    a. Under “Project Manager” click “Settings.”
    b. Under “Simulation” tab, under “Simulation top module name”, select “task1_tb.”
    c. Click “Ok.”
7. Run the simulation and check your results. The expected output is on the last page of Part 1. **Make sure that your testbench is automated, it cannot be manually tested.**
8. Vivado doesn’t include a way to print test bench results so you will have to take a screenshot and attach it to your report.

Task 2

This task is to implement the function

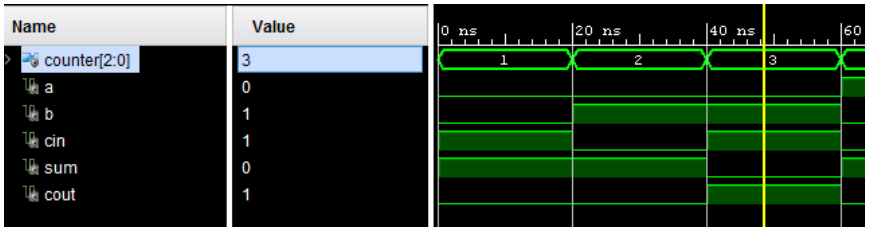
$F(A,B,C,D) = \overline{\overline{A\overline{C}\overline{D} \oplus \left(\overline{A}B + \overline{BC}\right)} + \overline{A}CD(B\overline{C} + A\overline{B}C\overline{D} + \overline{A}C\overline{D})}$ in task2.vhd.

Inputs: A, B, C, D

Outputs: F, G, H

1. Create the task2.vhd file:
    a. Under “Flow Navigator” click “Add sources.”
    b. Select “Create or add design sources.”
    c. Click “Create File”, enter the file name “task2”, hit “OK”, and then click “Finish.”
2. A window will pop-up to define the port connections. All inputs and outputs above will be 1 bit in length.
3. Simplify the equation above (call this output G) using Boolean algebra theorems and axioms and obtain the canonical SOP equation (call this output H). Write the code for all 3 equations (F as the original equation above, G as the simplified equation (only use the theorems and axioms to simplify it, show all work), and the canonical SOP equation in H) in the architecture portion of your entity. Make sure you have all 3 equations as you will compare these later in simulation.

4. Make sure it's syntax error free and can be simulated.
5. Create the task2_tb.vhd file:
    a. Under "Flow Navigator" click "Add sources."
    b. Select "Create or add simulation sources."
    c. Create task2_tb.vhd. Leave the entity empty, i.e. no inputs or outputs. Click "OK" if asked whether to use the given values (empty entity).
    d. Make sure simulation set is "sim_1", click "Finish."
6. Select the right simulation set:
    a. Under "Project Manager" click "Settings."
    b. Under "Simulation" tab, under "Simulation top module name", select "task2_tb.",
    c. Click "Ok."
7. Run the simulation and check your results. All 3 equations should produce the same output. If they do not, then your canonical form is not correct or you have made a mistake somewhere else in your simplification. No example outputs are given for this part. **<u>Make sure your testbench is automated, no manual testing is allowed.</u>**
8. Vivado doesn't include a way to print test bench results so you will have to take a screenshot and attach it to your report. Include your equation simplifications in your report as well.



Part of waveform of Task1

Grading

Code of task1.vhd	/3 Points
Code of task2.vhd	/5 Points
Screenshots of task1 simulation results	/4 Points
Screenshots of task2 simulation results	/4 Points
Simplification (G) and SOP expansion (H) of original function	/6 Points
Demo of task1 and task2	/16 Points
	/Total of 38 Points

<u>Part 2:</u>

Assignment Overview

In this assignment you will use Vivado 2019.2 Webpack to write and simulate a hex to seven segment display converter. You'll also need to turn in a single PDF to Canvas (instructions below).

Background

A seven segment display is a way to display a decimal or hex digit. There are various ways to label the segments of the display or encoding the digit, but below is what we're using for this computer assignment.

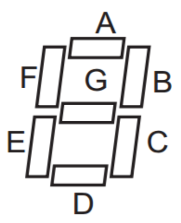


Figure 1: Seven Segment Display Layout

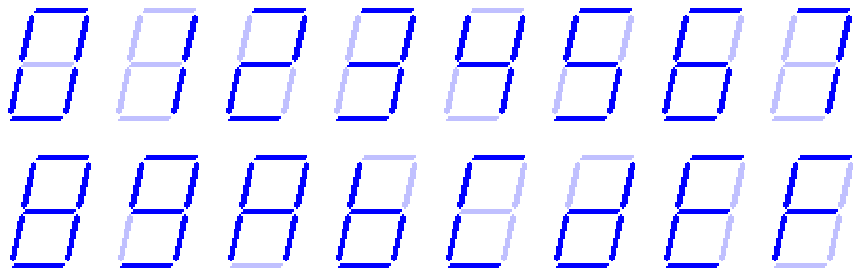


Figure 2: Hex Encoding

in(3)	in(2)	in(1)	in(0)	A	B	C	D	E	F	G
0	0	0	0							
0	0	0	1							
0	0	1	0							
0	0	1	1							
0	1	0	0							
0	1	0	1							
0	1	1	0							
0	1	1	1							
1	0	0	0							
1	0	0	1							
1	0	1	0							
1	0	1	1							
1	1	0	0							
1	1	0	1							
1	1	1	0							
1	1	1	1							

Table 1: Truth Table of the Encoder

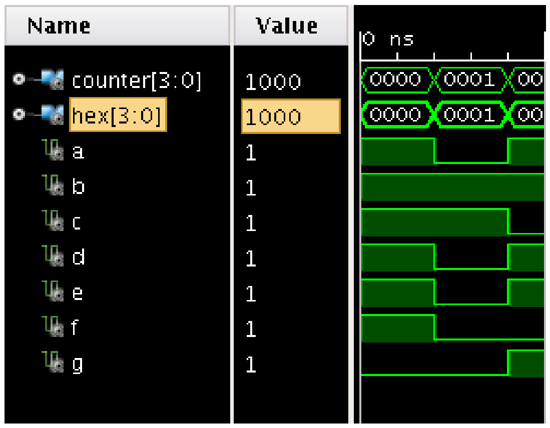
Getting started

1. Create a directory to store all of the assignment's files.
2. Open Vivado 2019.2 and create a new project called **computer_assignment_2**.
3. It will be an RTL project
4. No sources to input. Although if the target and simulation languages aren't set to VHDL, change them to VHDL.
5. No existing IP.
6. No constraints file.
7. The Basis3 uses an **XC7A35T-CPG236C** FPGA
    - Family: Artix-7
    - Package: CPG236
    - Speed: -1
    - Choose the part with 41600 FlipFlops

Task Details

This task is to implement an encoder whose input is a one 4-bit hex signal and the outputs are seven 1-bit signals for each display segment (A-G).

1. Create the encoder.vhd file:
    a. Under "Flow Navigator" click "Add sources."
    b. Select "Create or add design sources."
    c. Create task3.vhd and then click "Finish."
    d. Create an input named "hex_in" which is 4 bits wide (use a std_logic_vector for this).
    e. Create 7 outputs named "A" through "G" which are 1 bit wide each.
2. Once you have your VHDL source file, edit it to implement the encoder above. Make sure to fill in the table above and submit it with your report.
    - A when/else or if/else statement might be helpful
    - A internal signal of the outputs grouped together as a bus might help clean up the code
3. Make sure it's syntax error free.
4. Create the encoder_tb.vhd file:
    a. Under "Flow Navigator" click "Add sources."
    b. Select "Create or add simulation sources."
    c. Create encoder_tb.vhd.
    d. Make sure simulation set is "sim_1."
    e. Click "Finish".
5. Select the right simulation set:
    a. Under "Project Manager" click "Settings."
    b. Under "Simulation" tab, under "Simulation top module name", select "encoder_tb."
    c. Click "Ok."
6. Run the simulation and check your results. The expected output is on the last page of Part 1. **Make sure that your testbench is automated, it cannot be manually tested.**
7. Vivado doesn't include a way to print test bench results so you will have to take a clear screenshot of at least one iteration of the circuit.



Sample of the Expected Waveform (your view may be different)

Grading

Write up (both parts)	/20 Points
Filled out encoder table	/6 Points
Code of encoder.vhd	/6 Points
Screenshot of simulation results/waveform	/8 Points
Demo	/12 Points
	/Total of 52 Points

Self and Peer Evaluations (this will be scaled up significantly if an unequal amount of work has been shown between team members) /10 pts

Turn in

Make sure everything is turned in as a single PDF (both parts of this project) on Canvas. You can join PDFs with Adobe Acrobat if you have multiple PDFs. Name the assignment P1-xxxxxx-yyyyyy-zzzzzz, where xxxxxx is the last name of one lab partner, yyyyyy is the last name of the 2nd lab partner, and zzzzzz is the last name of 3rd lab partner.

Turn in only one PDF under one of the lab partner's Canvas account. Each team member should also submit a self- and peer-evaluation individually in Canvas. Make sure you evaluate both yourself and your teammate (don't forget to add all of your full names in your evaluation for full points).

Turn in one .zip archive of the project folder with all 3 task folders under this archive. You can use any archiving application to get a .zip as long as it includes all files, not just the project file.