Assignment 06

Assigned: 03/19/15 Due: 04/12/15

For submissions, use the assignment link on Blackboard

- You must submit your assignment as a pdf.
  - Use the format: filename <fsuid>.hw6.<ext>
  - For source code submitted, use the format: filename.<fsuid>.hw6.vhd
- Late work will not be accepted

The purpose of this assignment is to give you experience applying the algorithms developed in class to perform binary multiplication and division with signed and unsigned numbers. This assignment will also provide experience writing simple VHDL.

Suggested Problems From the Textbook (Ungraded)

None

Problems From the Textbook (Graded)

None

Additional Problems (Graded)

1. Show how to perform the operation (23 / 9) using the optimized division algorithm. Assume the operands are 6b instead of 32b (be sure that you adjust all of the registers appropriately). You must show each step, from initialization to the final product. ***(8)***

2. Point your browser to http://www.edaplayground.com/. You will need a Google or Facebook account to use this tool – you can quickly create a fake account if you do not already have one.

   At the landing, you should see something similar to Figure 1. The page is divided into three sections: Options, Testbench, and Source.

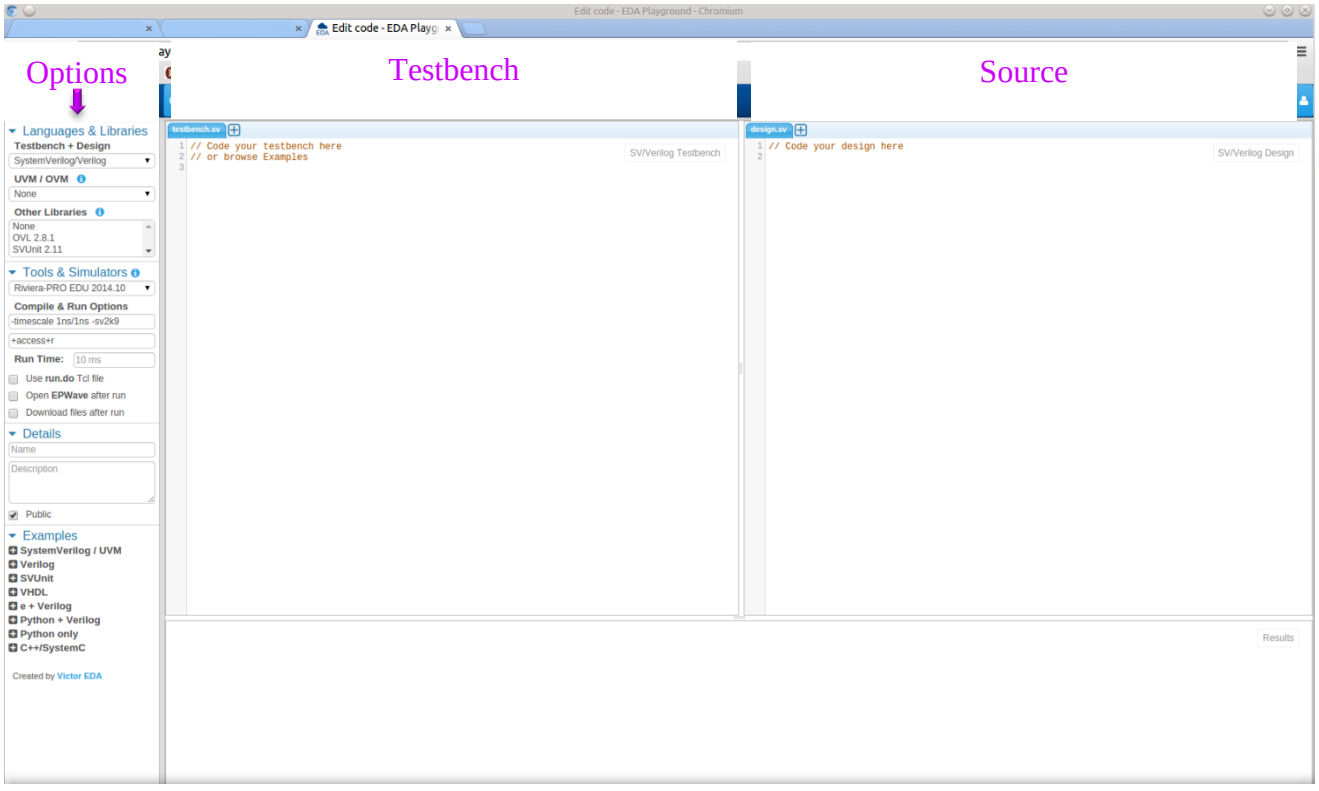


Figure 1 – EDA Playground Landing Page

Before we do anything, we need to change our environment to match our language. First, go to "Testbench + Design" and change the value in the drop-down menu from "SystemVerilog/Verilog" to "VHDL."

Now, expand the "Examples" dropdown menu and the VHDL tree. Select "OR Gate" from the menu. Afterward, you should have VHDL in both the testbench and source windows. Answer the questions below based on what you see in these windows.

a. What is the entity name in the design file? ***(1)***
b. What is the name of the architecture in the design file? ***(1)***
c. What is the name of the entity in the testbench file? ***(1)***
d. What is the name of the architecture in the testbench file? ***(1)***

3. Open https://en.wikipedia.org/wiki/IEEE_1164 in your browser.
   1. What does an 'X' represent when using the std_logic type? ***(1)***
   2. What does a 'Z' represent? ***(1)***

4. In the testbench window, find the line:

   b_in <= 'X'

   Change it to:

   b_in <= '0'

   Check the "Open EPWave after run" box on the left then click the "Run" button at the top of the page. This will take you to a functional simulation of the circuit. When you look at the diagram, you can see that *a*, *b*, and *q* follow *a_in*, *b_in*, and *q_out*. If your signals are missing, click the "Get Signals" box and select the *testbench* scope. Then select, "Append All" and close the dialog.
   a. Based on the VHDL in the source file, is *q_out* what you would expect it to be? Paste a screenshot of your timing diagram below (note: do not paste the entire page, crop it so that only the relevant information is shown). ***(4)***

5. Make a copy of the workspace by clicking "Copy" in the top menu. Now, remove the process from the architecture in the design file. Like this:

```
architecture rtl of or_gate is
begin
   q <= a OR b;
end rtl;
```

   a. Run the simulation again. Is the timing diagram the same or different? ***(1)***
   b. Based on the differences between sequential and concurrent operations in VHDL, is this what you would expect the behavior to be? ***(1)***

6. Now, you need to modify the source file (you can do this in the window or replace it by uploading an external copy). Add two outputs to the circuit, *x* and *y*. Add the appropriate code in the architecture to implement the functions:

   $x(a, b) = a \oplus b$
   $y(a, b) = a \cdot b$

   Replace the current testbench with *assignment.06.gate.test.vhd*.
   a. Look at the code beneath the PROCESS statement. What is this going to do? ***(1)***

7. Change the Run Time to 200 ns. Run your simulation and look at the waves.
   a. Are the results what you would expect? Paste a copy of your timing diagram below. ***(4)***

b. Upload a copy of your solution to Blackboard, rename it to problem01.<fsuid>.hw6.vhd and paste a copy of the source code here. Be sure that your solution has a header at the top that contains, at minimum, your name and your Blackboard user name. ***(4)***

```
Example Header
--------------------------------------------------------------------------
-- @file       design.ch03.vhd
-- @author     Clay Hughes
-- @date       01/11/11
-- @brief      Implements the circuit for Lab 1
--
-- @section DESCRIPTION
-- Not always necessary but a longer description can
-- be added here.
--------------------------------------------------------------------------
```

8. Create a new project (point your browser back to the homepage). Select VHDL as the language. Paste *assignment.06.alu.vhd* into the source code window and *assignment.06.alu.test.vhd* into the testbench window. You'll notice that you have been given a basic template that contains an entity and architecture declaration.

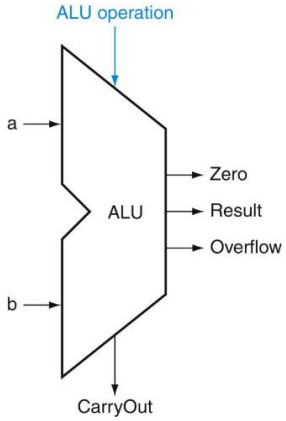
   If you look at the signals in the PORT declaration, you'll see that they are all of type std_ulogic_vector. Open https://www.cs.sfu.ca/~ggbaker/reference/std_logic/1164/std_logic_vector.html and look at the examples.
   a. What is a std_logic_vector? ***(1)***
   b. Give an example on how to declare a standard logic vector that can hold 32 elements. ***(1)***
   c. Show how you would assign element 11 of that vector to a signal of type std_logic, *a*. ***(1)***

   Below, in the architecture, you can see that there are additional signals defined, but they are of type unsigned. The unsigned type is derived from the std_logic type. In fact, when you look at the definition for unsigned, you can immediately see that it's just an array of type std_logic!

   ```
   type UNSIGNED is array (NATURAL range <>) of STD_LOGIC;
   ```

   So, why do we need this new type if it is exactly the same as std_logic_vector? Because it isn't! VHDL is a strongly typed language and sometimes we need to be able to perform arithmetic! Open the Wikipedia page for IEEE's numeric std package (https://en.wikipedia.org/wiki/Numeric_std). You'll find the list of operators here (for more details, you can go to https://standards.ieee.org/downloads/1076/1076.2-1996/numeric_std.vhdl). The only operators defined for the std_logic type are the logical operators – no arithmetic! You can get the details of the 1164 package here https://standards.ieee.org/downloads/1076/1076.2-1996/std_logic_1164-body.vhdl.

9. There are two pieces missing from this code: a case statement and an if-then statement. Fill in the missing code so that it implements the ALU shown in Figure 2 (Figure B.5.14). The "ALU Operation" signal in the figure is the *alu_ctrl* signal in the VHDL files. You do not need to implement overflow.



alu_ctrl	Function
0000	and
0001	or
0010	add
0011	sub
0100	slt
0101	nor

Figure 2 – MIPS ALU

Look at the if-then statement in the process. This is going to be used to implement the *slt* instruction in the ALU.

a. What does the OTHERS keyword do? ***(1)***
b. What is the variable slt_temp set to when (a < b) is true? ***(1)***
c. What is the variable slt_temp set to when (a < b) is false? ***(1)***
d. Fill in the missing sections of VHDL to complete the ALU. Upload a copy of your solution to Blackboard and paste a copy of the source code here. Be sure that your solution has a header at the top that contains, at minimum, your name and your Blackboard user name. ***(8)***

10. Once your code is complete, you will need to validate your design using the supplied testbench. Be sure that you set *mips_alu_testBench* as the top-level entity. The first step in validation is not going to be examining the wave, which is much too dense. Instead, a text file will be generated when you run the simulation. Click "Download files after run" on the left. Open the archive and file the file named "resultFile." If your implementation is correct, you should see errors reported for add and sub and nothing else. If you have other errors reported, go back and modify your implementation until you only have errors for add and sub.
    a. Are the errors reported for *add* and *sub* for all combinations? Explain. ***(3)***

11. Select "Open EPWave after run" to have the timing diagram open automatically. If you are missing signals in the diagram, add them using the "Get Signals" dialog. The scope will be *mips_alu_testBench* and then you can select "Append All." Save the project and then run the simulation. Even if the simulation failed, you will be prompted to download a zip file containing the results. Save this file somewhere on your computer.

    Assuming that your solution is correct, the diagram will be dense. You can use the magnifying class icons to zoom in and out on the diagram. You can click anywhere on the diagram to add a marker that you can use to control where the zoom is centered. Clicking "100%" will take you back to the default view.
    a. Paste a screenshot of the full timing diagram ***(1)***
    b. Paste a screenshot of the timing diagram from 200µs to 250µs. ***(1)***
    c. Paste a screenshot of the timing diagram from 400µs to 450µs. ***(1)***
    d. Paste a screenshot of the timing diagram from 750µs to 800µs. ***(1)***
    e. Paste a screenshot of the timing diagram from 1.2ms to 1.25ms. ***(1)***

Extra Credit

You can receive up to 5 points on your homework grade if you help improve this assignment. Be sure that your comments are constructive and that you provide specifics.