

Lab #2: VHDL Components

Lab 2 is due Monday, April 2. Your lab report and source code must be submitted by **10:10 AM** before class. The late policy applies to this lab project. This lab is to be done in **teams**. Each team member should be involved in each problem, e.g., one developing implementation models while the other building test benchmarks. Get started early! The required format for lab reports can be found on the resource page.

Problem 1:

Develop a behavioral VHDL model for a 4-to-1 word multiplexer (MUX). Your model should work with arbitrary length words; i.e., you should not place hard constraints on the lengths of inputs and outputs. Develop a test bench for your MUX that demonstrates each function **for all possible permutations of inputs other than the 4 words**.

Problem 2:

(a) Develop a behavioral VHDL model for a 4-bit shift register. Your shift register should implement functions for LOAD, HOLD, RIGHT SHIFT and LEFT SHIFT. In addition to regular inputs, your shift register should provide a SHIFT LEFT INPUT and a SHIFT RIGHT INPUT that input the value shifted into the right-most and left-most bits, respectively. You can multiplex a single port I_SHIFT_IN for the purpose of SHIFT LEFT INPUT and SHIFT RIGHT INPUT, as at most one is needed at a time. Your register should also include an ENABLE input and a CLOCK input. **The LOAD, HOLD, RIGHT SHIFT and the LEFT SHIFT functions should be rising-edge triggered.** Develop a test bench for your VHDL shift register that demonstrates each function for all possible combinations of input signals of "I_SHIFT_IN", "sel", "clock", "enable", and at least two values of "I". *shift_reg.vhdl* is a sample shift_reg program that only declares the interface. A sample test bench that tests only two input cases is provided in the file *shift_reg_tb.vhdl*.

What to turn in:

- (1) A gate-level RTL circuit schematic design. That means you can only use basic gates and flip-flops in the design. You can use any diagram drawing software to illustrate the design, or you can hand-draw and submit a picture of the drawing.
- (2) Implementation VHDL files and the test-bench for the top-level implementation.

(b) Use your 4-bit shift register from (a) to implement a structural VHDL model for an 8-bit shift register. This device should perform the same functions as the 4-bit shift register, but over 8 bits. Develop a test bench for your shift register that demonstrates each function for all possible combinations of input signals of "I_SHIFT_IN", "shift", "clock", "enable", "load", and two values of "I".

What to turn in:

- (1) A gate-level RTL circuit schematic design. That means you can only use basic gates and flip-flops in the design. However, if you use the 4-bit shift-register in 3.a unmodified, you can represent the 4-bit register as a blackbox.
- (2) Implementation VHDL files and the test-bench for the top-level implementation.

Problem 3:

Design and implement a 4-bit integer adder/subtractor. The inputs are signed, that means the most significant bit represents the sign of number, 0 for positive and 1 for negative. The negative inputs are represented with two's complement format. The inputs of the components are two 4-bit signal vectors, and its outputs include a 4-bit signed signal vector for result, 1 bit carry (overflow) signal and 1 bit underflow signal. You may use any sequential or parallel digital algorithms for the add/subtract operations. Note that you can NOT use the "+" and "-" operators in your implementation.

What to turn in:

- (1) A gate-level RTL circuit schematic design. That means you can only use basic gates and flip-

- flops in the design.
- (2) Implementation VHDL files and the test-bench for the top-level implementation.
-

Use synthesizable features of VHDL (IMPORTANT)

You should only use the synthesizable features of VHDL. If your code doesn't conform to the following requirements, the score for that part of code will be zero. Note that the following requirements only apply to the implementation part. You can use ANY features in your test benches.

VHDL feature requirements:

- (1) No variables are allowed;
 - (2) Up to one "wait" statement OR a sensitivity list is allowed in a process.
 - (3) A process should wait on (using a "wait" statement or the sensitivity list) either (1) ALL signals appearing on the right-hand-side in the process body; or (2) the clock signal or with the optional reset signal.
 - (4) No "wait for" statements or time expressions in signal assignment statements.
-

How to test using a VHDL simulator

You can use any VHDL simulator for this project. Choices include GHDL and Xilinx Vivado simulator. Below is an example process with GHDL.

GHDL

If you want to use GHDL, you can download it from <http://ghdl.free.fr>. The site provides pre-compiled packages for Linux, Windows and MacOS. You can also download the source and compile/install from there.

How to use GHDL

Assume GHDL is installed under the directory /usr/local/bin. Assuming we have "shift_reg.vhdl", which is the implementation of a 4-bit shift register, and "shift_reg_tb.vhdl", which is the test bench for our implementation, there are three steps to run the test bench:

- (1) Analyze: Compile the two vhd files
- ```
"/usr/local/bin/ghdl -a shift_reg.vhdl"
"/usr/local/bin/ghdl -a shift_reg_tb.vhdl"
```

If you use any IEEE libraries, add "--ieee= standard" after "-a".

- (2) Generate the executable for the test bench:
- ```
"/usr/local/bin/ghdl -e shift_reg_tb"
```

If you use any IEEE libraries, add "--ieee= standard" after "-e".

- (3) Run the test bench:
- ```
"/usr/local/bin/ghdl -r shift_reg_tb"
```

If you want to dump waveform files, add "--vcd=shift\_reg.vcd". You can use "gtkwave" or any other waveform viewers to open the "shift\_reg.vcd" file.

---

### **What to Turn In**

For this lab project, turn in all of your circuit design and source code, including the code that implements the components, and the code that tests your implementations. Describe your testing methodology, and where applicable, explain why you select the input values that are used to test your implementation.

**Peer evaluation:** Please email your peer evaluation (guideline is on the “Files” page on CANVAS) of your teammates, including scores and justification, directly to the TA.