

VHDL Assignment #1: Getting Started with the EDA Playground Online HDL Simulator

1 Instructions

- Due date: Friday, September 25, 2020 by 5pm.
- submission is in teams using myCourses (only one team member submits). In the report, provide the names and McGill IDs of the team members.
- Late submissions will incur penalties as described in the course syllabus.

2 Introduction

In this assignment you will be required to write simple logic functions in VHDL by following a step-by-step tutorial.

3 Learning Outcomes

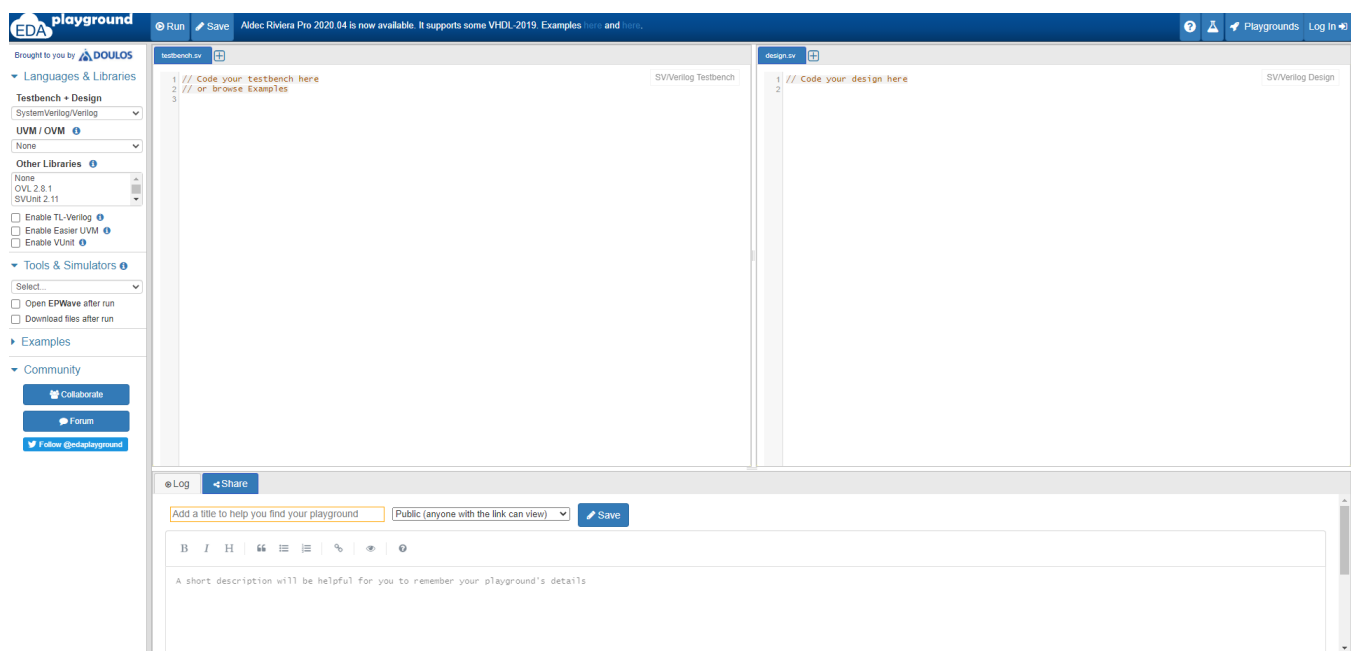
After completing this lab you should know how to:

- Synthesize logic functions
- Use EDA Playground and VHDL to implement logic functions
- Perform functional simulation

4 Getting Familiar with EDA Playground

In this course you will be using the **EDA Playground HDL simulator**: EDA Playground supports immediate hands-on exposure to simulating SystemVerilog, Verilog, VHDL, C++/SystemC, and other programming languages. All you need is a web browser. The goal is to accelerate learning of design/testbench development with easier code sharing and simpler access to EDA tools and libraries.

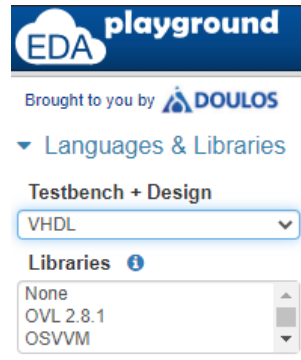
To begin, go to the EDA Playground [website](#) and wait until the following window appears:



To access all available tools and simulators, sign up (or log in) by clicking the log in icon on the top right corner.

5 Creating a New Project

Under **Language & Libraries** select VHDL for “*Testbench + Design*”:



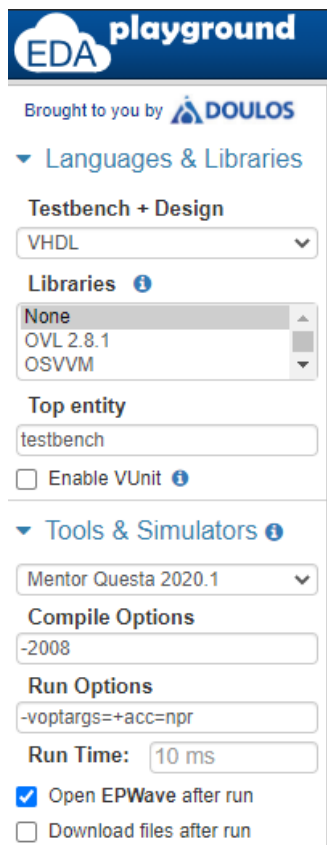
You will now see two windows named **testbench.vhd** and **design.vhd**. The design.vhd window is used to describe logic circuits with VHDL and the testbench.vhd is used to perform simulations. Type the following VHDL code in the design.vhd window. This simple VHDL code describes a single logic OR gate.

```
design.vhd +  
1  -- Simple OR gate design  
2  library IEEE;  
3  use IEEE.std_logic_1164.all;  
4  
5  entity my_OR_gate is  
6  port(  
7      a: in std_logic;  
8      b: in std_logic;  
9      q: out std_logic);  
10 end my_OR_gate;  
11  
12 architecture RTL of my_OR_gate is  
13 begin  
14     q <= a OR b;  
15 end RTL;
```

Type the following VHDL code in the testbench.vhd window. The testbench is used to test all possible binary values (“0” and “1”) for the inputs *a* and *b*.

```
testbench.vhd
1  -- Testbench for OR gate
2  library IEEE;
3  use IEEE.std_logic_1164.all;
4
5  entity testbench is
6  -- empty
7  end testbench;
8
9  architecture tb of testbench is
10
11  -- DUT component
12  component my_OR_gate is
13  port(
14    a: in std_logic;
15    b: in std_logic;
16    q: out std_logic);
17  end component;
18
19  signal a_in, b_in, q_out: std_logic;
20
21  begin
22
23  -- Connect DUT
24  DUT: my_OR_gate port map(a_in, b_in, q_out);
25
26  process
27  begin
28    a_in <= '0';
29    b_in <= '0';
30    wait for 1 ns;
31
32    a_in <= '0';
33    b_in <= '1';
34    wait for 1 ns;
35
36    a_in <= '1';
37    b_in <= 'X';
38    wait for 1 ns;
39
40    a_in <= '1';
41    b_in <= '1';
42    wait for 1 ns;
43
44    -- Clear inputs
45    a_in <= '0';
46    b_in <= '0';
47
48    wait;
49  end process;
50  end tb;
51
```

To perform (run) the simulation, type *testbench* in the box located under the **Top entity**. Now, select *Mentor Questa 2020.1* under **Tools & Simulators** and check the **Open EPWave after run** box.



EDA playground

Brought to you by DOULOS

▼ Languages & Libraries

Testbench + Design

VHDL

Libraries

None

OVL 2.8.1

OSVVM

Top entity

testbench

☐ Enable VUnit

▼ Tools & Simulators

Mentor Questa 2020.1

Compile Options

-2008

Run Options

-voptargs=+acc=npr

Run Time: 10 ms

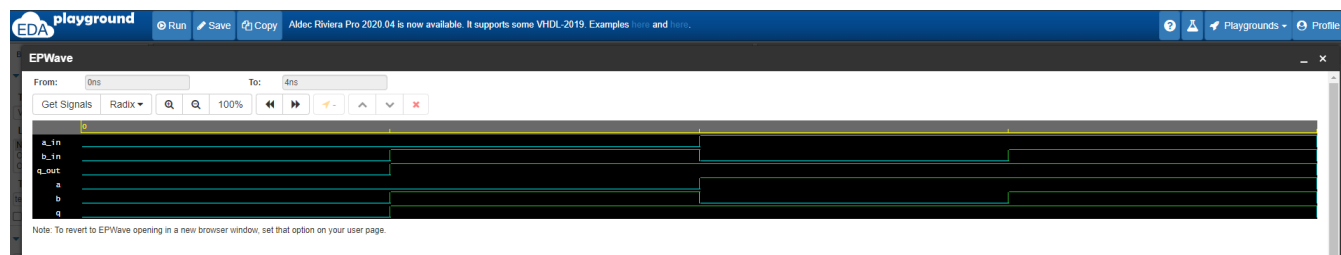
☒ Open EPWave after run

☐ Download files after run

Save your progress and **Run** your simulation.



The following window will pop up (**EPWave**). If not, make sure that your browser does not block pop ups for the EDA Playground website. The result of your simulation in the form of a waveform is plotted in this window. You can use **Get Signals** to add or remove signals from your waveform.



6 Synthesize your Circuit

In order to synthesize your work, you need to add a new file to your project. Create a new file by clicking on the **plus (+)** icon next to your testbench.vhd file and name it **run.do**. Then, add the following lines to the run.do file:

```
setup_design -manufacturer Xilinx -family Artix-7 -part 7A100TCSG324

foreach arg $::argv add_input_file $arg

compile

synthesize
```

```

auto_write precision.v
report_output_file_list
report_area
report_timing
exec cat precision.v

```

Now, select *Mentor Precision 2019.2* under **Tools & Simulators** and **Run** EDA Playground. The resulting parameters of the synthesized circuit, will be shown in the Log window:

The screenshot shows the EDA Playground interface. On the left, the 'Languages & Libraries' sidebar is visible, with 'VHDL' selected under 'Testbench + Design'. The 'Tools & Simulators' section shows 'Mentor Precision 2019.2' selected. The main editor displays a testbench.vhd file with the following code:

```

1 setup_design -manufacturer xilinx -family Artix-7 -part 7A100TCSG324
2 foreach arg $::argv {add_input_file $arg}
3 compile
4 synthesize
5 auto_write precision.v
6 report_output_file_list
7 report_area
8 report_timing
9 exec cat precision.v
10

```

Below the editor, the 'Log' window shows the synthesis results. The output includes information about the device (7A100TCSG324) and a table of resource utilization:

```

# Info: 4 /home/runner/impl_1/or_gate_env.ntm
# Info: 5 /home/runner/impl_1/or_gate.edf
# Info: 6 /home/runner/impl_1/or_gate.v
# Info: 7 /home/runner/impl_1/or_gate.xdc
# Info: 8 /home/runner/impl_1/or_gate.tcl
# Info: *****
# Info: Device Utilization for 7A100TCSG324
# Info: *****
# Info: Resource Used Avail utilization
# Info: -----
# Info: I/Os 3 210 1.43%
# Info: Global Buffers 0 32 0.00%
# Info: LUTs 1 63400 0.00%
# Info: CLB Slices 1 15850 0.01%
# Info: DFFs or Latches 0 126800 0.00%
# Info: Block RAMs 0 135 0.00%
# Info: DSP48E1s 0 240 0.00%
# Info: -----
# Info: *****
# Info: Library: work Cell: or_gate View: rtl
# Info: *****
# Info: Number of ports : 3

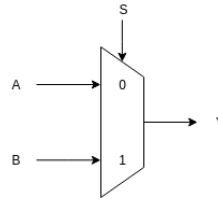
```

Note that for this Lab you only need the results of **Device Utilization**.

Students are encouraged to visit the EDA Playground [VHDL Examples](#) page to practice additional examples and improve their VHDL skills.

7 VHDL Implementation of a 2-to-1 MUX

A multiplexer is a circuit that selects between several inputs and forwards the selected input to a single output. In general, a 2^n -to-1 MUX has 2^n inputs with n selectors. A schematic diagram of a 2-to-1 multiplexer is shown below.



According to this schematic, the 2-to-1 multiplexer sends the input signal A to the output when the selector signal S is equal to '0', alternatively, the input signal B is sent to the output. In this lab, you will implement a 2-to-1 multiplexer in VHDL using structural and behavioral styles. In the structural modeling of the 2-to-1 multiplexer in VHDL, the multiplexer is implemented using AND, OR, and/or NOT gates only. More specifically, the structural description of the multiplexer literally realizes its Boolean function. Describe the Boolean function of the 2-to-1 MUX in VHDL using AND, OR, and/or NOT gates only. Use the following entity declaration for your VHDL description of the 2-to-1 multiplexer:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity MUX_structural is
    Port (A      : in  std_logic;
          B      : in  std_logic;
          S      : in  std_logic;
          Y      : out std_logic);
end MUX_structural;
```

Side note: You can create/add .vhd files to your project by clicking on the **plus (+)** icon next to your design.vhd file. Leave the design.vhd file empty as you will not use it in this assignment. Name your .vhd files the same as the entity name. Do not forget to include the file extension “.vhd” when naming your .vhd files.

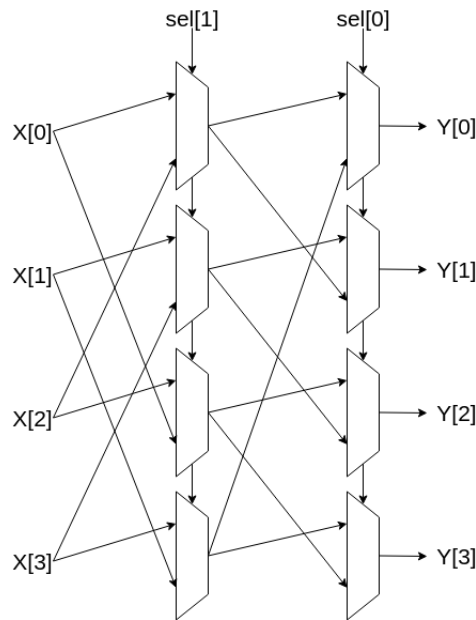
Once completed, you will describe the architecture of the 2-to-1 multiplexer using the behavioral style. In the behavioral description, you describe the behavior of your target circuit and the synthesis tool generates a gate-level layout of your design. Use (only) a single VHDL select assignment and the entity declaration below to implement a behavioral description of the 2-to-1 multiplexer.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity MUX_behavioral is
    Port (A      : in  std_logic;
          B      : in  std_logic;
          S      : in  std_logic;
          Y      : out std_logic);
end MUX_behavioral;
```

Once both behavioral and structural styles of the 2-to-1 multiplexer are described in VHDL, you are required to test your circuits. Write a testbench code and perform an exhaustive test for both VHDL descriptions of the 2-to-1 multiplexer. An exhaustive test is a test that simulates the circuit for all possible input patterns.

8 VHDL Implementation of a 4-bit circular barrel shifter

In this part of the lab, you will design a circuit that implements an 4-bit circular barrel shifter. The circular barrel shifter is a circuit that can shift its input by a given number of bits using combinational logic. The 4-bit circular barrel shifter is usually implemented by stacking two layers of 2-to-1 multiplexers as shown below. All four multiplexers in the leftmost layer use `sel[1]` as the selector signal. Similarly, all four multiplexers in the rightmost layer use `sel[0]` as the selector signal. Make sure that you familiarize yourself with how the barrel shifter works by calculating its output for several input combinations by hand.



Similar to the first part of the assignment, you will implement the 4-bit barrel shifter in VHDL using both structural and behavioral styles. To obtain a structural description of the 4-bit barrel shifter, you are required to use the structural description of the 2-to-1 multiplexer. Write a structural VHDL description for the 4-bit circular barrel shifter by instantiating the structural description of the 2-to-1 multiplexer eight times. Use the following entity declaration for your structural VHDL description of the 4-bit circular barrel shifter:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity barrel_shifter_structural is
    Port (X      : in  std_logic_vector (3 downto 0);
          sel    : in  std_logic_vector (1 downto 0);
          Y      : out std_logic_vector (3 downto 0));
end barrel_shifter_structural;
```

Once completed, you are required to implement the 4-bit circular barrel shifter using the behavioral style. One way to obtain a behavioral description of the 4-bit circular barrel shifter is the use of VHDL select statements. Write a behavioral VHDL code for the 4-bit circular barrel shifter using (only) a single VHDL select statement. Use the following entity declaration for your behavioral VHDL description of the 4-bit circular barrel shifter:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity barrel_shifter_behavioral is
    Port (X      : in  std_logic_vector (3 downto 0);
          sel    : in  std_logic_vector (1 downto 0);
          Y      : out std_logic_vector (3 downto 0));
end barrel_shifter_behavioral;
```

Hint: You may want to use the VHDL concatenate operator *ampersand* (&). It can be used to combine two or more signals together. Note that all input signals to the concatenation must be of the same type and the result of the concatenation needs to fit the **exact width** of the concatenated input signals.

Once you have your circuit described in VHDL using both structural and behavioral styles, write a testbench code and perform an exhaustive test for both VHDL descriptions of the 4-bit circular barrel shifter.

9 Deliverables

Once completed, you are required to submit a report explaining your work in detail, including answers to the questions related to this lab. You will find the questions in the next section. You must also submit all the design

files (*i.e.*, your VHDL code and testbench files) along with Log content of the synthesized circuits. If you fail to submit any part of the required deliverables, you will incur grade penalties as described in the syllabus.

10 Questions

1. Explain your VHDL code.
2. Report the number of pins and logic modules used to fit your designs on the FPGA board.

	2-to-1 MUX		4-bit circular barrel shifter	
	Structural	Behavioral	Structural	Behavioral
Logic Utilization (in LUTs)				
Total pins				

3. Show representative simulation plots of the 2-to-1 MUX circuits for all the possible input values (exhaustive test results). Note that you can simply include snapshots from the waveform that you obtained from EPWave. In order to fully capture all the signals from the waveform, you can adjust the display range using the magnifier icons. Make sure that all signal names and axes are properly visible.
4. Show representative simulation plots of the 4-bit circular shift register circuits for a given input sequence, *e.g.*, “1011” ($X = \text{“1011”}$), for all the possible shift amounts.