Laboratory Session 1
Introduction to VHDL: Behavioral VHDL

Objective

- Understanding the basics of VHDL descriptions
- Use Quartus IDE
- Use ModelSim Simulator

Software Packages

- Altera Quartus II
- ModelSim

1. Introduction

VHSIC Hardware Description Language (VHDL) is a programming language used to describe hardware. Unlike other programming languages such C or Java which are sequential languages, VHDL is concurrent (parallel) language. VHDL statements run at the same time

2. Samples

2.1. Half Adder

For this exercise we will implement a Half Adder using behavioral VHDL. A Half Adder takes two 1-bit inputs and outputs the sum and carry of the addition (See Figure 1).

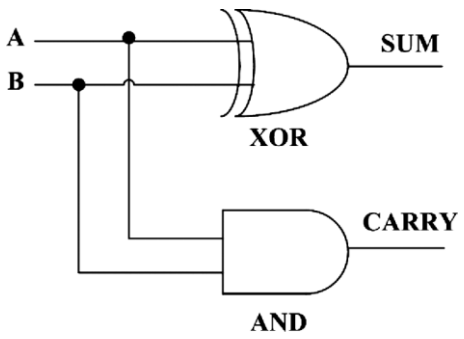

Figure 1. Half Adder Diagram

First start by creating the project in Quartus II using the following steps:

1. Start by creating a folder (eg. Half_Adder) to save your project.
2. Run Quartus II.
3. Click on Create a New Project and then click on Next
4. On the right of the first Textbox click on, to select the location of the folder created previously.

5. In the second Textbox, write a name for the project name (eg. Half_Adder) and then click on Finish.
Next create a new VHDL file
1. File → New → VHDL File → OK

2. Or click on → VHDL File → OK

Paste the following code into the new VHDL file created:

```
library ieee;
use ieee.std_logic_1164.all;
entity Half_Adder is port(
        A, B: in std_logic;
        SUM, CARRY: out std_logic
);
end entity;

architecture beh of Half_Adder is
begin

SUM <= A xor B;
CARRY <= A and B;

end architecture;
```

3. Save the file using the entity name (Half_Adder)

4. Press the start compilation button and wait until it finishes.

5. If the compilation is successful a popup window will appear stating that "Full compilation was successful"

6. The flow summary tab will appear showing the total logic elements used.

7. Generate the RTL diagram: Tools → Netlist Viewer → RTL viewer

8. Now, to check if the schematic of the Half Adder is the same as the previous figure we need to get the RTL view.

9. Verify both designs.

Test the functionality of your implementation using ModelSim as follows:

1. Create a new folder (eg. Half_AdderSim) to save the ModelSim project.

2. Run ModelSim-Altera.

3. Click on Jumpstart, then Create a Project

4. Provide a project name (eg. Half_Adder) and select the location of your new folder then click OK.

5. A new popup window will appear, select Add Existing File.

6. Select browse and go to the location of the Quartus project folder then select the .vhd file created (eg. Half_Adder.vhd)

7. Click OK then Close.

8. The vhd file is added to ModelSim with a question mark (?) under its status, this means that the file was not compiled yet under ModelSim.

9. Compile → Compile All, you will see a green check mark next to the file

10. To start the simulation, click on Simulate → Start Simulation.

11. A new popup window will appear. Click on + next to work and select the entity you want to simulate (eg. half_adder) and click OK.

12. Several windows will appear. The important ones are Objects and Wave. If you can find them select View → Objects or View → Wave.

13. Select all objects (a, b, sum, carry), right click on them then select Add → to Wave → selected signals.

14. The signals will appear in the Wave window.

15. There two methods to specify the input value: Force and Clock. Force sets a single value for the input whereas Clock alternates the value of the input between high and low. Right click on a in the wave form and select Clock, a popup window will appear with the Tile Define Clock, keep the Period 100 which means each clock cycle is 100ps, select Falling for First Edge and click OK

16. Repeat the same step for b but make the period twice that of a.

17. Click on Run in the toolbar and repeat 4 times to cover all combination of a and b.

18. Verify the results of sum and carry.

2.2. Full Adder

A Full Adder takes two 1-bit inputs and a carry in and outputs the sum and carry of the addition (See Figure 2). Apply the steps from the previous exercise to implement and verify a Full Adder. Start by creating a new folder named Full_Adder then create a new project in Quartus and call it Full_Adder.

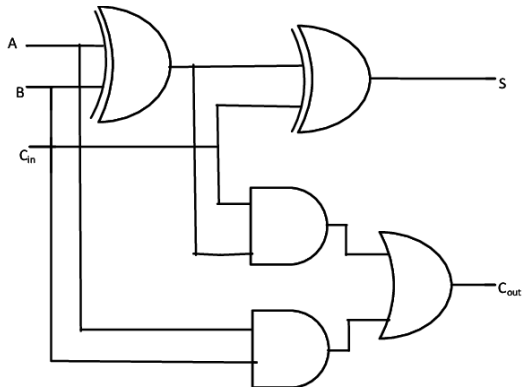

Figure 2. Full Adder Diagram

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Full_Adder is
Port ( a,b,cin : in STD_LOGIC;
s,cout : out STD_LOGIC);
end entity;

architecture beh of Full_Adder is
begin

s <= a xor b xor cin;
cout <= (a and b) or (b and cin) or (cin and a);

end architecture;
```

After completing the steps of ModelSim simulation, do the following:

1. Repeat the same step for ***cin*** but make the period twice that of ***b***.
2. Click on Run in the toolbar and repeat 8 times to cover all combination of ***a***, ***b*** and ***cin***.
3. Verify the results of sum and carry.

3. Practice Exercises

3.1. Recognizer:

Design, implement and verify a 4-bit recognizer. If the *input* is greater than 8 it should output 1 otherwise 0.

3.2. Odd Parity Generator

Design, implement and verify an odd parity generator. The generator takes a 4-bit binary value. The parity generator outputs the odd parity bit for the 4-bit input.

4. Deliverables

Practical demonstration of the sample exercises will be requested during the lab. Keep in mind that a combined report that presents the analysis and testing of the sample exercises as well as the practice exercises of lab 1 and 2 will be requested, so maintain your project file for the snapshots of the work done.