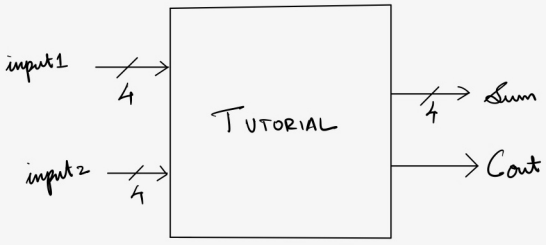
Quartus and QuestaSim - Tutorial
Reconfigurable Computing - Fall 2023
University of Windsor

Roche Christopher, Dr. Mohammed Khalid

1 Introduction

This tutorial is intended to acquaint oneself with Quartus Prime Lite, Questasim and VHDL. A 4-bit adder will be implemented and simulated using VHDL in quartus and questasim. Pictorial represenation of the hardware model that we will be implementing is given below.



The model has 2 4-bits inputs, *input1* and *input2*, a 4-bits output *sum* and a single bit output, *cout*. The model performs binary addition on the two inputs and sets the output to *sum* and if there is an overflow, the *cout* port is set to high.

2 Quartus Prime Lite Software

2.1 Setting up the project in Quartus Prime

1. Open the quartus prime software
2. Create a new project.
   (a) Click **New Project Wizard** under File
   (b) Click **Next** in the wizard, Enter the name of the project (eg. *tutorial*) and click **Next**
   (c) Select **Empty Project** and click **Next** twice
   (d) Click the **Board** tab at the top, select **MAX 10** as Family, select **MAX 10 DE10-Lite** and click **Next**

(e) Select **Questa Intel FPGA** as Simulation Tool and **VHDL** as format.

(f) Click **Next** and click **Finish**.

2.2 Coding and Compilation

1. Create a new VHDL file

   (a) Click **New** under **File**

   (b) Select **VHDL file** under **Design Files** in the new wizard.

2. Write the following code in the VHDL file and save it as *tutorial.vhdl*

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.std_logic_unsigned.all;

entity tutorial is
port(
   input1, input2: in std_logic_vector(3 downto 0);
   sum: out std_logic_vector(3 downto 0);
   cout: out std_logic
);
end entity;

architecture arch of tutorial is
   signal output_signal: std_logic_vector(4 downto 0);
   begin
      process(input1, input2)
      begin
         output_signal <= ('0' & input1) + ('0' & input2);
      end process;

      process(output_signal)
      begin
         sum <= output_signal(3 downto 0);
         cout <= output_signal(4);
      end process;
end architecture;
```

3. Click **Start Compilation** under **Processing** tab.

3 Simulation using QuestaSim

1. Compile the *vhdl* file

   (a) Make sure the **Project** tab is selected, not the **Library** tab, in the left bottom section of the window.

(b) Right click the **vhdl** file, select **compile** option and click **compile selected**.

2. Simulate using GUI

   (a) Click **Simulate** tab and click **Start Simulation**.

   (b) A wizard will open and expand the **work** section.

   (c) Select the file to simulate, in our case it is *tutorial*, and click **ok**

   (d) The ports of the entity will be listed under the objects tab. Assign values to the input ports

      i. Right click the object *input1*, click **Modify**, select **Apply Wave**.

      ii. In the wizard that opens up, set the *Start Time* as **0**, *Stop Time* as **100**, click **Next**, set the value to **0011** and click **Finish**. Now a wave window will appear with the *input1* added to it.

      iii. Right click the object *input1* in the wave window, click **Edit**, select **Wave Editor** and click **Create/Modify Waveform**.

      iv. In the wizard that opens up, set the *Start Time* as **100**, *Stop Time* as **200**, click **Next**, set the value to **1011** and click **Finish**

      v. Right click the object *input2*, click **Modify**, select **Apply Wave**.

      vi. In the wizard that opens up, set the *Start Time* as **0**, *Stop Time* as **200**, click **Next**, set the value to **1010** and click **Finish**.

   (e) Add the waves of the output port to wave window.

      i. Right click the *output* object under the *Objects* section and click **Add Wave**.

      ii. Right click the *overflow* object under the *Objects* section and click **Add Wave**.

   (f) Run the simulation

      i. Click the *Simulate* tab, Select *Run* and click **Run -All**

3. Simulate using testbench
   When the size of the project becomes bigger, simulating using graphical user interface becomes time consuming, cumbersome and virtually impossible. Testbench is preferred in scenarios such as those.

   (a) Create a file named *tutorial_tb.vhdl*

   (b) Change the VHDL version to standard *1076-2008*.

      i. Right click *tutorial_tb.vhdl* and select **Properties**.

      ii. Click **VHDL** tab, select **Use 1076-2008** and click **Ok**

   (c) Copy the following testbench code to that file and compile it the way the vhdl model file was compiled.

```
library ieee;
use ieee.std_logic_1164.all;
use std.env.all;
```

```
entity tutorial_testbench is
end entity;

architecture tut_testbench_arch of tutorial_testbench is

   component tutorial is
   port(
      input1, input2: in std_logic_vector(3 downto 0);
      sum: out std_logic_vector(3 downto 0);
      cout: out std_logic
   );
   end component;

   signal input1, input2, sum: std_logic_vector(3 downto 0);
   signal cout: std_logic;

   begin
      uut: tutorial
      port map(
      input1 => input1,
      input2 => input2,
      sum => sum,
      cout => cout
      );

      process
      begin
      input1 <= "1010";
      input2 <= "0011";
      wait for 5 ns;

      input1 <= "1000";
      input2 <= "1000";
      wait for 5 ns;

      input1 <= "1100";
      input2 <= "0011";
      wait for 5 ns;

      input1 <= "1111";
      input2 <= "0001";
      wait for 5 ns;

      stop(0);

   end process;
end architecture;
```

(d) Make sure that both the *tutorial.vhdl* and *tutorial_tb.vhdl* exist in the same

project. Then compile the programs by clicking **Compile** tab and **Compile All**.

(e) Click **Simulate** tab and click **Start Simulation**. Click **tutorial_testbench** in the wizard and click **OK**.

(f) Under *sim* tab, right click **tutorial_testbench**, click **Add to**, click **Wave** and select **All items in region**. All waves should appear in the *Wave* window.

(g) Click **Simulate** tab, select **Run** and click **Run -All**.

4 Appendix

The code attached in this document is available in the github repository, `https://github.com/rocheparadox/Reconfigurable-computing-fall-2023-quartus-tutorial`