Digital Systems Design II

COEN 313 (FJ-X)

Experiment 1

Andre Hei Wang Law

4017 5600

Performed on September 20, 2021

Due on October 4, 2021

“I certify that this submission is my original work and meets the Faculty’s Expectations of Originality”.

Saturday, October 2, 2021

4017 5600

1) Objectives

For the first experiment of the course COEN 313, students have a main objective of familiarizing themselves with using VHDL simulation software tool, logic synthesis tools as well as FPGA implementation software tools. In conjunctions with using properly these tools, they will also have to learn the basics of Modelsim Do files.

2) Procedure

In order to familiarize the students to the various tools used in this course, students will begin practicing writing, simulating and running several codes such as guided in the “Digital logic Simulation and Synthesis Using Modelsim Tutorial” and “Nexys Quickstart Tutorial” documents. Afterwards, they will simulate a full adder program in which the results and outputs are displayed through the FPGA board. Given enough practice, students will complete the objective of learning the basics of using the simulation software tool, the logic synthesis tools and the FPGA implementation software tools.

3) Results

3.1) Modelsim simulation results of the full adder and the complete VHDL source code of the full adder together with DO file used to simulate the full adder.

```
[reliability] [/home/l/l_heiwan/Modelsim/Code/COEN313/LAB1] > ls -al
total 40
drwxrwx--- 4 l_heiwan l_heiwan 4096 Sep 20 16:10 .
drwxrwx--- 3 l_heiwan l_heiwan 4096 Sep 20 15:16 ..
-rw-rw---- 1 l_heiwan l_heiwan  689 Sep 20 15:23 full_adder_nexysboard.vhd
-rw-rw---- 1 l_heiwan l_heiwan  424 Sep 20 15:27 fulladder.xdc
-rw-rw---- 1 l_heiwan l_heiwan  260 Sep 20 15:18 half_adder_regular_outputs.vhd
drwxrwx--- 7 l_heiwan l_heiwan 4096 Sep 20 15:34 lab1
-rw-rw---- 1 l_heiwan l_heiwan 1939 Sep 20 15:42 vivado.jou
-rw-rw---- 1 l_heiwan l_heiwan 4098 Sep 20 16:10 vivado.log
drwxrwx--- 2 l_heiwan l_heiwan 4096 Sep 20 16:10 .Xil
[reliability] [/home/l/l_heiwan/Modelsim/Code/COEN313/LAB1] >
```

Figure 1. Directory of the VHDL Source Code of the Full Adder

```
[reliability] [/home/l/l_heiwan/Modelsim/Code/COEN313/LAB1] > cat -n full_adder_nexysboard.vhd
     1  library ieee;
     2  use ieee.std_logic_1164.all;
     3
     4  entity full_adder is
     5          port (carry_in, input1, input2 : in std_logic;
     6                  sum_out, carry_out : out std_logic);
     7  end full_adder;
     8
     9  architecture structural of full_adder is
    10
    11  component half_adder
    12          port (in1, in2 : in std_logic;
    13                  carry, sum : out std_logic);
    14  end component;
    15
    16  signal carry1, carry2 : std_logic;
    17  signal sum_int          : std_logic;
    18
    19  for ha1, ha2 : half_adder use entity
    20  WORK.half_adder(true_outputs);
    21
    22  begin
    23
    24  ha1: half_adder port map(in1 => input1, in2 => input2,
    25                                  carry => carry1, sum=> sum_int);
    26  ha2: half_adder port map(in1 => sum_int, in2 => carry_in,
    27                                  carry => carry2, sum => sum_out);
    28  carry_out <= carry1 or carry2;
    29
    30  end structural;
    31
[reliability] [/home/l/l_heiwan/Modelsim/Code/COEN313/LAB1] >
```

Figure 2. VHDL Source Code of a Full Adder

```
[reliability] [/home/l/l_heiwan/Modelsim/Code/COEN313/LAB1] > cat -n half_adder_regular_outputs.vhd
     1  library ieee;
     2  use ieee.std_logic_1164.all;
     3
     4  entity half_adder is
     5          port (in1, in2 : in std_logic;
     6                  carry, sum : out std_logic);
     7  end half_adder;
     8
     9  architecture true_outputs of half_adder is
    10  begin
    11          carry <= (in1 and in2);
    12          sum <= (in1 xor in2);
    13  end true_outputs;
[reliability] [/home/l/l_heiwan/Modelsim/Code/COEN313/LAB1] >
```

Figure 3. VHDL Source Code of a Half Adder

```
[reliability] [/home/l/l_heiwan/Modelsim/DO] > cat -n full_adder.do
     1  force in1 0
     2  force in2 0
     3  run
     4
     5  force in1 1
     6  force in2 0
     7  run
     8
     9  force in1 0
    10  force in2 1
    11  run
    12
    13  force in1 1
    14  force in2 1
    15  run
[reliability] [/home/l/l_heiwan/Modelsim/DO] >
```

Figure 4. DO File Source Code

3.2) Xilinx XCD file used in the Vivado implementation.

```
[reliability] [/home/l/l_heiwan/Modelsim/Code/COEN313/LAB1] > cat -n fulladder.xdc
     1  set_property -dict { PACKAGE_PIN J15 IOSTANDARD LVCMOS33 } [get_ports { carry_in} ];
     2  set_property -dict { PACKAGE_PIN L16 IOSTANDARD LVCMOS33 } [get_ports { input1 } ];
     3  set_property -dict { PACKAGE_PIN M13 IOSTANDARD LVCMOS33 } [get_ports { input2 } ];
     4  set_property -dict { PACKAGE_PIN K15 IOSTANDARD LVCMOS33 } [get_ports { carry_out }];
     5  set_property -dict { PACKAGE_PIN H17 IOSTANDARD LVCMOS33 } [get_ports { sum_out } ];
[reliability] [/home/l/l_heiwan/Modelsim/Code/COEN313/LAB1] >
```

Figure 7. Xilinx XDC File of the Full Adder

4) Questions

4.1) What is the advantage of using the -r option in a force command within a DO file?

In a force command, the usage of the -r option allows the code to repeat a precise number of times. In other words, the -r option allows the user to repeat an action without having the need to type that section of the code multiple times. For more details, the syntax of the -r option force command is as follows:

“force signal_name signal_value start_time -r repeat_time “

4.2) Briefly explain two methods of creating a repeating periodic signal using DO files.

A method of making a periodic signal relies on a DO file that makes use of another DO file such that the first DO file will read the second DO file multiples times. This prevents situation where the user has to use a brute force DO file where all repeated section is explicitly stated multiple times. The advantage of this method is more efficient that the brute force one as it is clearer which DO file does what and it will also be easier to perform modifications, since we won’t have to manually change the code multiple times.

The second method and the most efficient method is to use a force command with a repeat function. This is done through the force command making use of the -r option which directly repeats the action for the user. As such, the code will be much more compact and the number of iterations can be changed easier.

4.3) What is a .xdc file used for?

A “Xilinx Design Constraints” file is a type of constraint file that allows the user to bind specific inputs or outputs to specific pins on a FPGA board. In other words, an .xdc file contains the necessary information concerning the bounded relation between ports in the VHDL entity to the physical pins of the board, such as to allow interactions between the source code and the FPGA board.