ECSE 325

Lab 1 Report

Furkan Ercan, Arash Ardakani

Group 04

Alexander Hale, Anna Bieber

260672475, 260678856

March 02, 2018

Introduction

This laboratory served to teach us the basics of Quartus FPGA software and how to compile synchronous circuit VHDL declarations to a target FPGA. To accomplish this, the lab instructions provided a step-by-step instruction list to create a project, configure the selection of FPGA, create a new file, and enter code. More specifically, the code implements a synchronous 8-bit counter.

Design

The VHDL description of the synchronous 8-bit counter can be seen in Figure 1.

In the first block of code, the required libraries are imported.

In the second block, the counter entity is declared. The entity has inputs for the clock, counting direction, reset bit, and enable bit, as well as one output bit vector.

The final block is the architecture declaration, which contains a process block with the counting logic. First, since the clock is synchronous, a conditional statement ensures that the following logic only executes on the rising clock edge. Next, the reset bit is checked: if it is high, the counter is reset to zero. Then, it is verified that the enable bit is high, before checking the counting direction. If the direction bit is high the counter increases, and if the direction bit is low the counter decreases.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity g04_lab1 is
   Port( clk         : in std_logic;
         direction   : in std_logic;
         rst         : in std_logic;
         enable      : in std_logic;
         output      : out std_logic_vector(7 downto 0));
end g04_lab1;

Architecture bitCounter of g04_lab1 is
   signal count : unsigned(7 downto 0);
   begin
   counter1 : process(rst, clk)
      begin
         if rising_edge(clk) then
            if rst = '1' then
               count <= X"00";
            elsif enable = '1' then
               if direction = '1' then count <= count + 1;
               else count <= count - 1;
               end if;
            end if;
         end if;
      end process;
      output <= std_logic_vector(count);
end bitCounter;
```

Figure 1

After the process block, the value of the counter is assigned to the output vector.

Compilation

To compile the VHDL description, the *Compile* button is pressed. The quick compilation runs relatively quickly, while the full compilation takes longer. During compilation, the compiler is assigning the description's logic to different gates on the FPGA.

The compilation flow summary is shown in Figure 2.

Flow Summary	
Flow Status	Successful - Thu Mar 01 11:59:04 2018
Quartus Prime Version	16.0.0 Build 211 04/27/2016 SJ Lite Edition
Revision Name	g04_lab1
Top-level Entity Name	g04_lab1
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	5 / 32,070 (< 1 %)
Total registers	8
Total pins	12 / 457 (3 %)
Total virtual pins	0
Total block memory bits	0 / 4,065,280 (0 %)
Total DSP Blocks	0 / 87 (0 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0 / 6 (0 %)
Total DLLs	0 / 4 (0 %)

Figure 2

Resources

As seen in the compilation report, 8 registers are used for this design. If the counter bit size increased, the resource allocation would also increase, because more lines would be required to transmit the data. For example, if the counter is increased to 16 bits, the register use increases to 16 and the pin use increases to 20 (Figure 3).

Flow Summary	
Flow Status	In progress - Fri Mar 02 14:05:49 2018
Quartus Prime Version	16.0.0 Build 211 04/27/2016 SJ Lite Edition
Revision Name	g04_lab1
Top-level Entity Name	g04_lab1
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	N/A
Total registers	16
Total pins	20
Total virtual pins	0
Total block memory bits	0
Total DSP Blocks	0
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0
Total DLLs	0

Figure 3

Chip Planner

The chip planner is shown in Figure 4. The resources used are highlighted in dark blue on the figure. We can observe that a very small part of the resources are utilized which aligns with the code written (very few resources demanded).

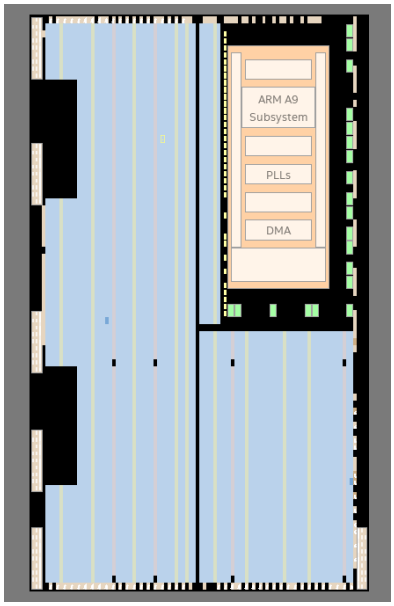

Figure 4

RTL

The RTL view of the circuit is shown in Figure 5.

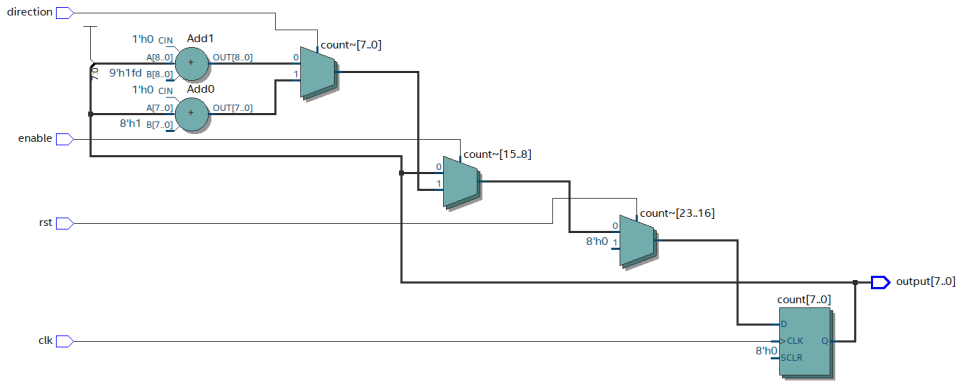

Figure 5

We observe the usage of 2 adders, 3 multiplexers and a register. These components allow us to implement an 8-bit counter.

The multiplexer is a combinational circuit designed to route one of several input lines through to a single output line by the application of a control signal. In this case, the control signals are the direction, enable, and reset signals. Each multiplexer selects between 2 inputs, depending on the control signal.

The register at the end of the circuit is used to hold the data so that it can be output.

Simulation

The VHDL design was simulated in Modelsim. The clock, direction, and reset signals were manually manipulated to verify that the counter functioned properly (Figure 6).

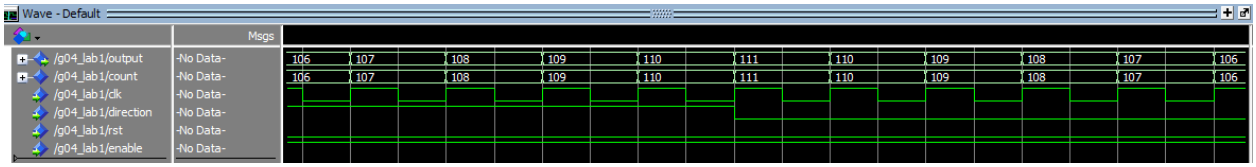

Figure 6

Conclusion

This lab allowed us to learn the basics of Quartus-II FPGA software through the implementation of an 8-bit counter. We were able to create a framework for a new project as well as writing the description of the counter in VHDL code. Furthermore, this allowed us to understand the logic utilization within the FPGA board, as well as the floorplan and RTL viewer of a circuit.