**ECSE 325**

**Lab 1 Report**

Furkan Ercan, Arash Ardakani

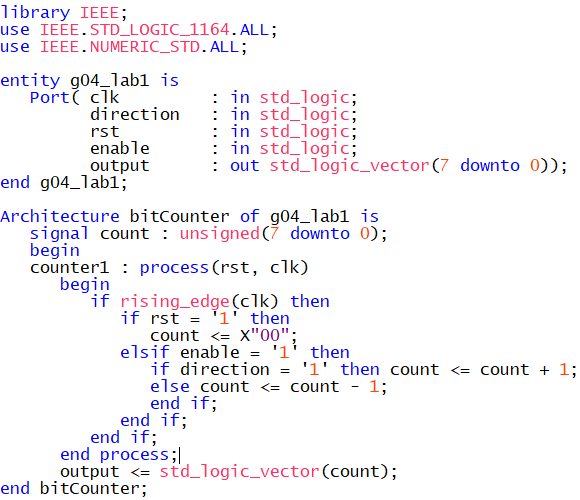
Group 04

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.

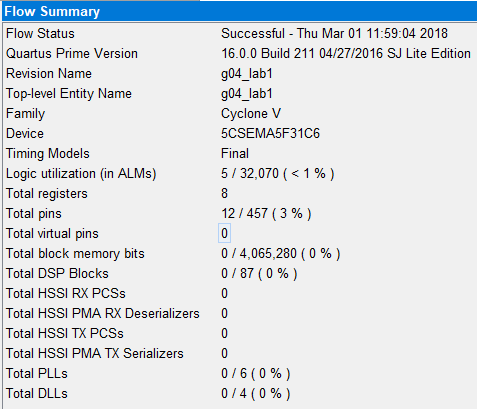
In the first block of code, the required libraries are imported.

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.

**Figure 1**

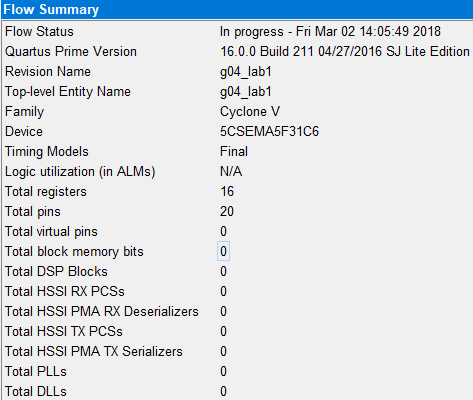
After the process block, the value of the counter is assigned to the output vector.

**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.

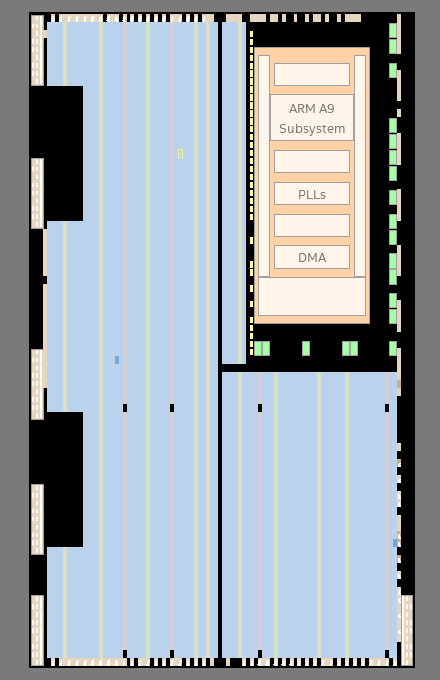
The compilation flow summary is shown in Figure 2.

**Figure 2**

**Resources**

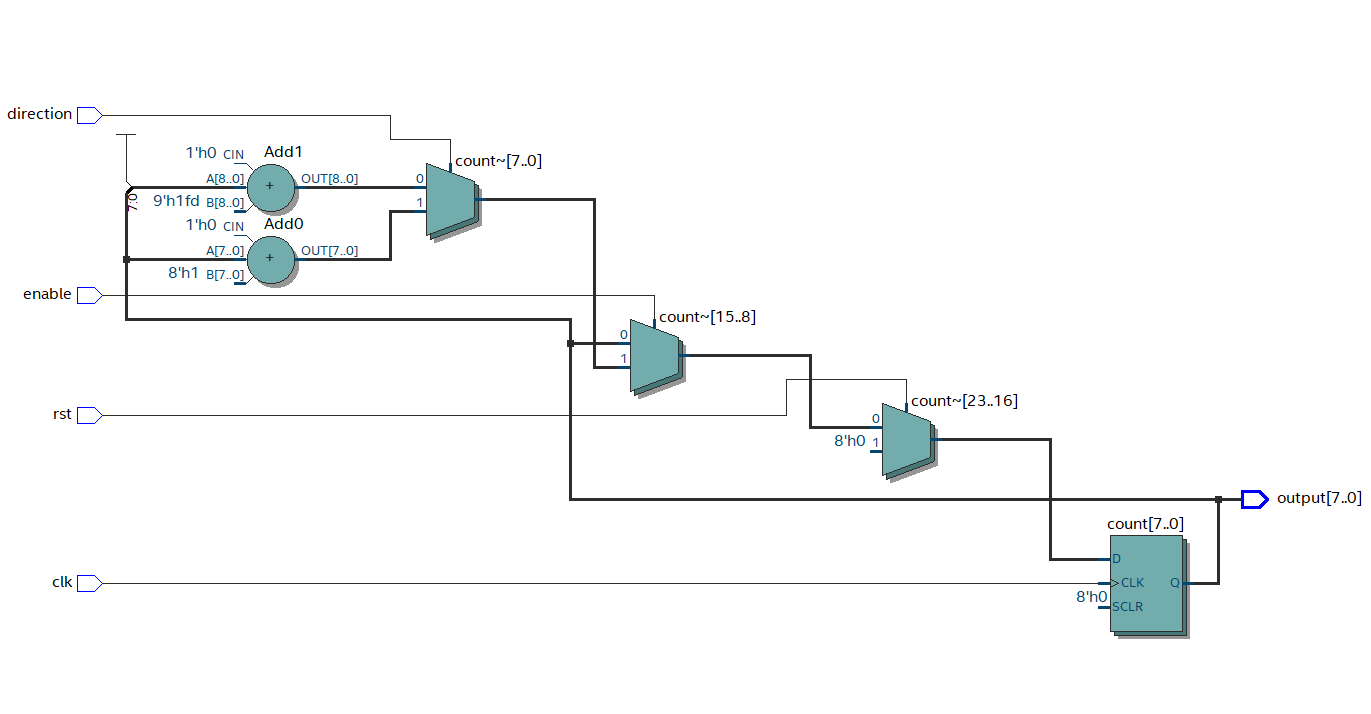
As seen in the compilation report, 8 registers are used for this design. If the counter bit size increased, the resource allocation would also increase, because more lines would be required to transmit the data. For example, if the counter is increased to 16 bits, the register use increases to 16 and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

**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).

**RTL**

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

**Figure 5**

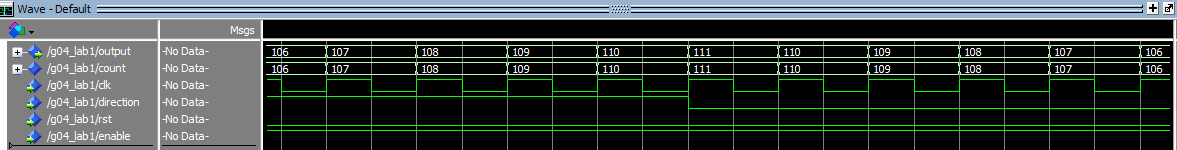
**Figure 4**

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.