Digital Logic Group 69

ECSE 222

Anthony Porporino: 260863300

Tyler Watson: 260867260

# Lab Assignment 2: Written Report

**Prof. Ioannis Psaromiligkos**

**McGill Faculty of Engineering**

April 1, 2019

**Description of counter and clock divider circuit:**

The up counter circuit takes as inputs 1 bit values for the enable, reset, and clock parameters and outputs a 4 bit binary number. In terms of the architecture, the process statement starts the sequence of the counter and takes in the clock and reset parameters. If the reset has a value of 0, the count is reset to “0000” (reset is active low). If the reset value is 1, and the clock is on its rising edge, then if enable is 1, the count is incremented by 1. Otherwise, if enable is 0, the count stays the same.

The clock divider takes the same inputs, however it outputs a 1 bit value for count_out. The clock divider utilizes a down counter, which is similar to the up counter, but whose initial value is set to 499999. The count is continuously decremented at each rising edge until it reaches 0. Once this occurs, the output, which was previously set to 0, is set to 1.

Both these circuits are considered sequential designs because they follow a sequence of steps. The process block allows these sequences to be performed and values are checked and updated at each run of the sequence. Both circuits proceed and change in unison with the clock’s rising edge and the output is adjusted accordingly.

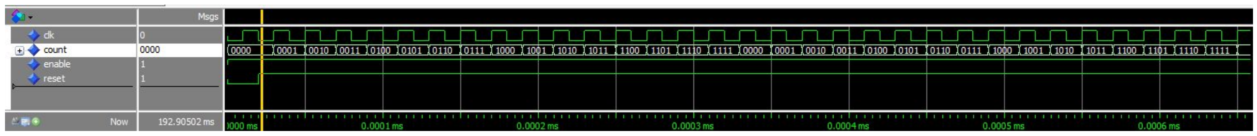
**Why we build the divider using a down counter:**

A down counter is utilized because it is slightly simpler to implement than an up counter. The up counter would require us to compare the value with a larger number than simply comparing it to 0 to know when to reset the count. The down counter is therefore slightly less time costly and also is a more accurate depiction of a countdown.

**How the counter and clock divider was tested:**

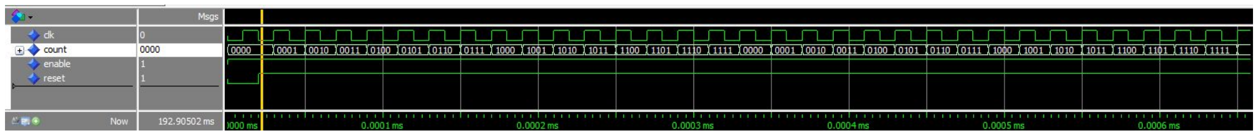
The counter circuit was tested using a testbench file. Using wait periods, the values of the enable and reset inputs were changed from 1 to 0 and the outputs were put in a wave to examine the outputs. The outputs correctly corresponded to the expected output values.

Figure 1: Counter Simulation Wave



The clock divider was also tested using a testbench file. The enable and reset started at 0 then were changed to 1 and the down counter in the clock divider circuit worked as expected. Once the count reached 0, the output was 1 and it was reinitialized to its initial value. A wave was generated to see these value changes

Figure 2: Clock Divider Simulation Wave



**Description of stopwatch circuit:**

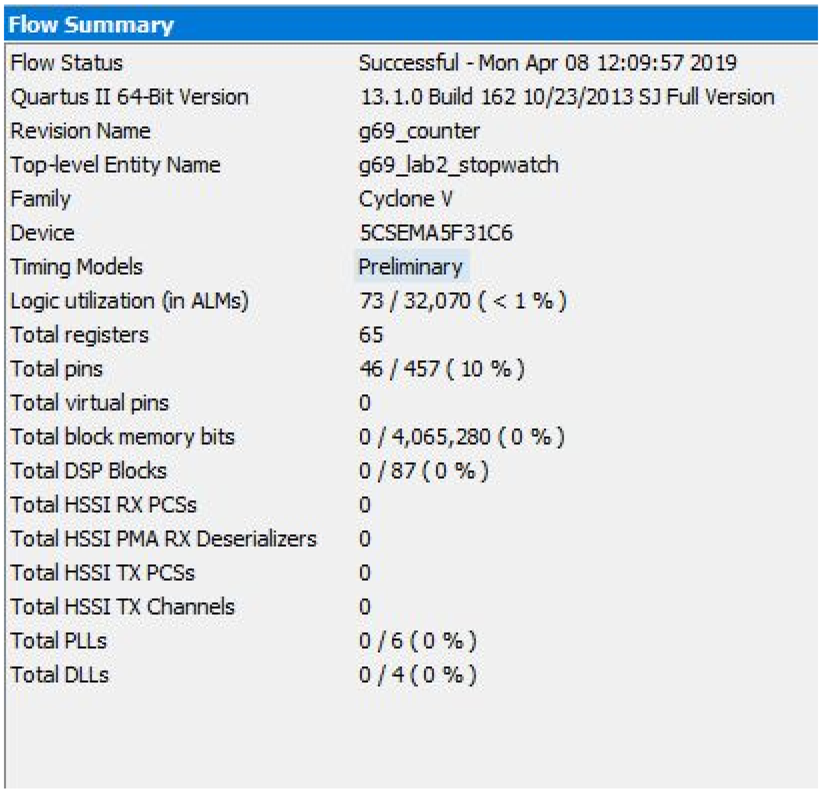
The stopwatch circuit takes as input four one bit values and outputs six seven bit logic vectors corresponding to each number to be displayed on the FPGA board. Firstly, six logic vector signals are initialized to store each undecoded value that will be decoded using the segment decoder to appear on the board. Six signals are initialized for the enable of each counter component and six signals for the resets of each counter component. Finally, a signal is made for the hundredth state and the stopwatch state. One clock divider component instance is created, six counter components instances are created, and six seven segment decoder components are created. The six counters are necessary because each value to be displayed on the board must behave independently and only count up to its maximum value (either 5 or 9). Based on the values of start and stop, the stopwatch state was set, which then allowed the hundredth state to be set. This hundredth state acts as the enable input for the first counter. This starts the stopwatch. The stopwatch runs until either the stop button or reset button are pressed. Each subsequent counter is enabled when the previous counter reaches its maximum number (either 9 or 5). The undecoded values are finally passed through the seven segment decoder so that they display correctly on the Altera Board. This is all encapsulated in a process block.

## **How the stopwatch circuit was tested:**

The circuit was tested using a ModelSim simulation wave. The inputs and output signals were added to the wave. The clock was forced to a simple clock. The scaling was adjusted to be able to see the changing hundredth seconds.The inputs of the stopwatch were each forced to different values and the values of the outputs were examined for a 100 ms simulation length. Each undecoded and decoded hex output value corresponded with the expected value for those specific inputs. Thus, the stopwatch works correctly and accurately.

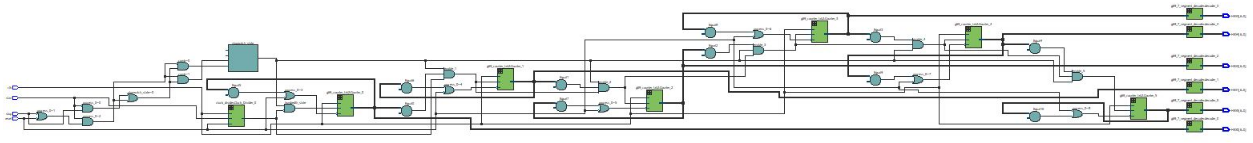
The final stopwatch test was done by directly using the board and testing the behaviour of the different buttons (start, stop and reset). The stopwatch functioned properly as it should.

Figure 3: Compilation Report Flow Summary



| Flow Summary | |
|---|---|
| Flow Status | Successful - Mon Apr 08 12:09:57 2019 |
| Quartus II 64-Bit Version | 13.1.0 Build 162 10/23/2013 SJ Full Version |
| Revision Name | g69_counter |
| Top-level Entity Name | g69_lab2_stopwatch |
| Family | Cyclone V |
| Device | 5CSEMA5F31C6 |
| Timing Models | Preliminary |
| Logic utilization (in ALMs) | 73 / 32,070 ( < 1 % ) |
| Total registers | 65 |
| Total pins | 46 / 457 ( 10 % ) |
| 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 TX Channels | 0 |
| Total PLLs | 0 / 6 ( 0 % ) |
| Total DLLs | 0 / 4 ( 0 % ) |

Figure 4: RTL Schematic for the Stopwatch



The green rectangles represent the counter components. The counters are sequential because each counter affects the behaviour of the subsequent counter. Thus, they are placed in order of the smallest units to the largests. Finally, all the values of these counters are passed through the seven segment decoder so that they correctly display on the board.