Danny Dutton
4/1/2015
ECE 3544

Project 3: Design and Synthesis of a Counter

Objective:

The objective of this project was to design a 16-bit counter that would output to seven-segment displays. The counter would be simulated in ModelSim and then synthesized with Quartus along with a seven-segment driver.

Design:

The 16-bit counter is a finite state machine, specifically a Moore machine. This is due to how the output is only dependent on the current state, the current count. The state is dependent on the previous state and the inputs:

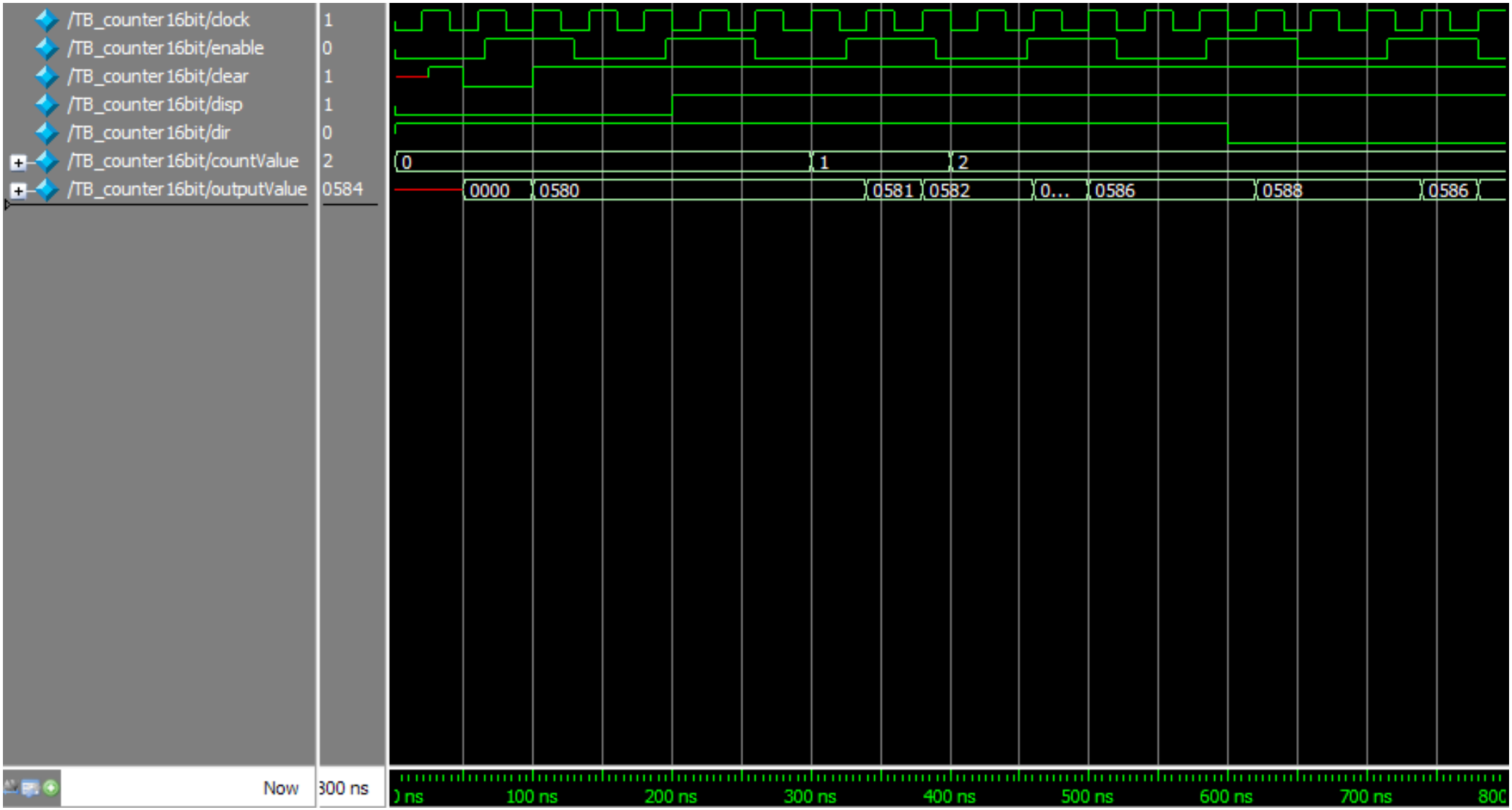
- dir
  - This input determines whether the counter will increment or decrement.
- disp
  - This input will determine whether the next count state will show my student ID number or the count.
- countValue
  - This is a 4-bit input that determines how much the count should change every iteration.
- enable
  - This input will either increment or decrement the count when disp = 1.

There is alsoa module called keypressed. This FSM will send out a pulse that will last for one clock period every time KEY[1] is pressed. This ensures that the enable button is held down long enough for the counter to register it as well as debouncing the button.

In addition, there are four seven-segment display drivers used to display the count on the LED displays. These use an always block to change the driver output depending on the digit passed to it.

Simulation Results:

The simulation of the module resulted in the expected values as per the specification sheet. The counter would increment when dir is high and disp is high and would decrement when dir is low and disp is high. When disp is low, the display would show 0580, my last four of my student ID number. The clear button would always reset the count back to zero and enable would always either decrement, increment or do nothing depending on how the switches were set. The next page shows a waveform of TB_counter16bit, the test bench used to test the counter.


/TB_counter16bit/clock
/TB_counter16bit/enable
/TB_counter16bit/clear
/TB_counter16bit/disp
/TB_counter16bit/dir
/TB_counter16bit/countValue
/TB_counter16bit/outputValue
1
0
1
1
0
2
0584
0
1
2
0000
0580
0581
0582
0...
0586
0588
0586
Now
300 ns
) ns
100 ns
200 ns
300 ns
400 ns
500 ns
600 ns
700 ns
800

Synthesis Results:
The synthesis of the design worked just as expected on the FPGA. The only noticeable issue is that during random occurrences, the enable button would cease to increment or decrement the count no matter what the countValue was. Hitting the clear button would reset the count and the enable button would work as expected again. I believe this might have something to deal with the keypressed module and how it goes through the states, possibly placing the enable button in a state of being held down and not letting it go to the released or free state. There was no pattern to this happening. Sometimes it would happen after a many button presses, sometimes less. Sometimes it would happen during large count values, sometimes during small. However, the problem seemed to happen less often the longer I played with the buttons.

Conclusions:
A logic machine should have at least two procedural blocks of code: the combinational logic block and a clocked register. The combination logic block should have inputs that include the current state. The current state along with the other inputs creates the next state. In a Mealy machine, the machine output will come from this block.

The blocks should act as clocked flip-flops that will pass the next state on to the first combinational logic block, thus becoming the current state after the clock pulse. In a Moore machine, the current state is also passed onto a second combinational logic block.

The second combinational logic block contains any logic on how the current state will affect the output of the machine. The counter used a Moore machine. The first procedural block was the flip flop, the second block was the initial combinational logic block, and the last assignment acts as the secondary combinational block, connecting the current state to the output.