System Synthesis and Modeling

Project 4

Zack Fravel

9/29/16

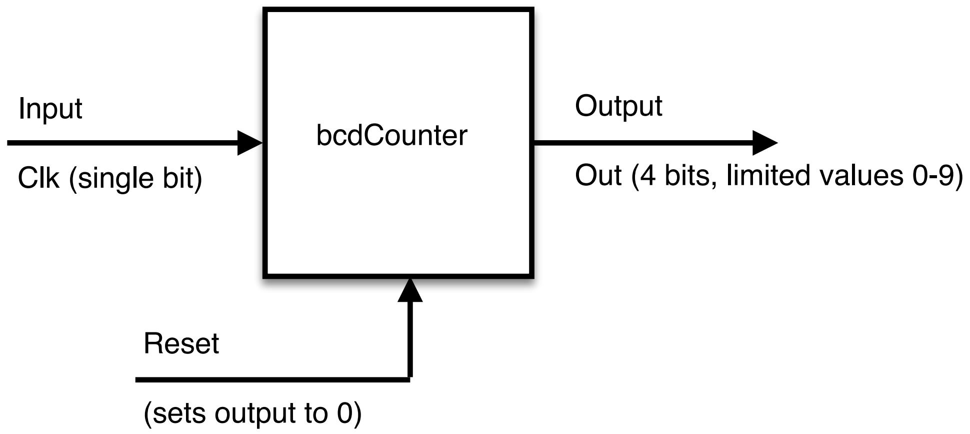
zpfravel@uark.edu

Project Objective

The objective of project 4 was to design a Binary-Coded-Decimal (BCD) counter whose initial count is 5. Binary coded decimals refer to all the single digit integers, or numbers that can be displayed on a single seven segment display for instance (0 to 9 / 0000 to 1001). This project is different from our last projects in that this is a sequential circuit, meaning we have to account for the passing of time and have our circuit respond accordingly. After designing our BCD counter, we're also tasked with writing a testing solution, in my case a testbench file with simulation waveform.

Design Methodology

When beginning the design process, we're tasked this time with fairly generic instructions and are left to make most assumptions ourselves about the full functionality of the circuit. In my case, I designed a bcdCounter which takes two inputs and has one output. The two inputs are the clock source and a reset, allowing the counter to go back to 0 when a value of '1' is asserted. The output is four bits, because our maximum value, 9 (1001), requires four bits to store its value. Below is a circuit diagram for my bcdCounter.



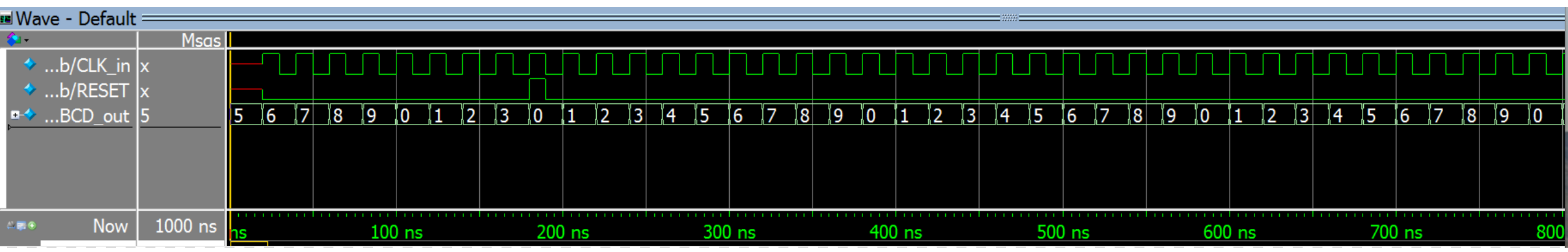
The way this is implemented in verilog is fairly straight forward. I initialize my two inputs as wires and my output as a register as well as have my initial statement set the output to 5, as requested in the assignment details. Below that, I created an always block that is sensitive to the positive edge of the clock signal and contains a nested if statement. If reset goes high, the output is set to zero; otherwise, the circuit checks if the output is less than 9 and increments the output value. Once the output reaches 9, it hits the final else condition that resets the count back to zero. I use nonblocking assignments for all three since we are dealing with sequential logic circuits we don't want any assignment timing mistakes. The source code is appended to the end of the report.

Simulation

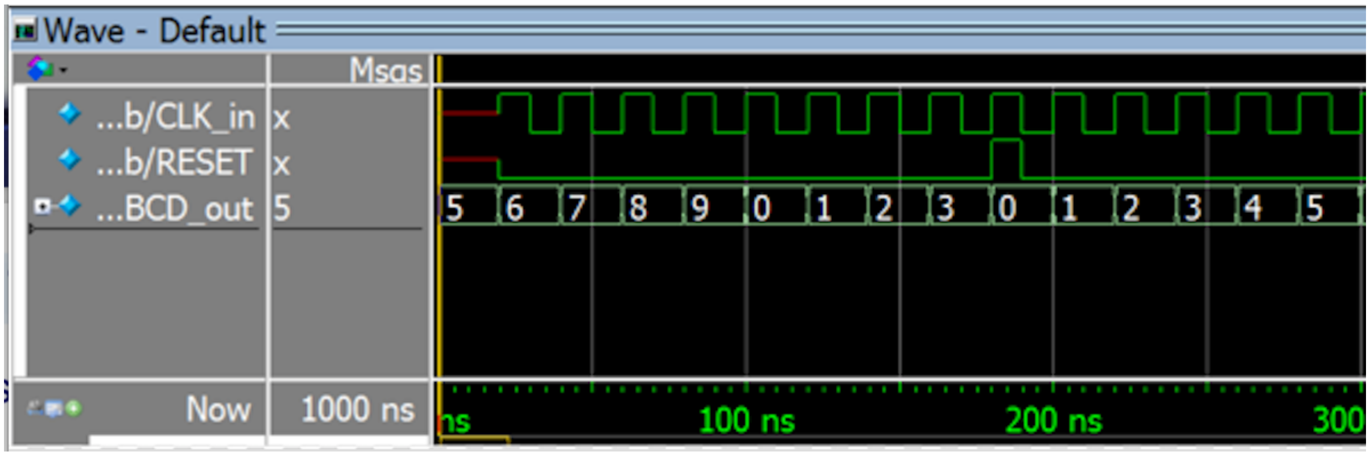
For my simulation method I chose to write a verilog testbench file. I initialize my timescale to 1ns/1ns at the top of the module and declare my inputs CLK_in and RESET as registers and my BCD_out as a wire[3:0]. Once the signals are declared, I instantiated my bcdCounter test with the declared ports above. In my testbench I created two initial blocks. One waits 20 ns, sets CLK_in and RESET to 0, and waits another 20 ns. The other initial block waits for the 180 ns and sets RESET to 1 for 10 ns. To complete the testbench, I instantiate a 20 ns clock cycle with an always statement that waits 10 ns and sets CLK_in to its compliment.

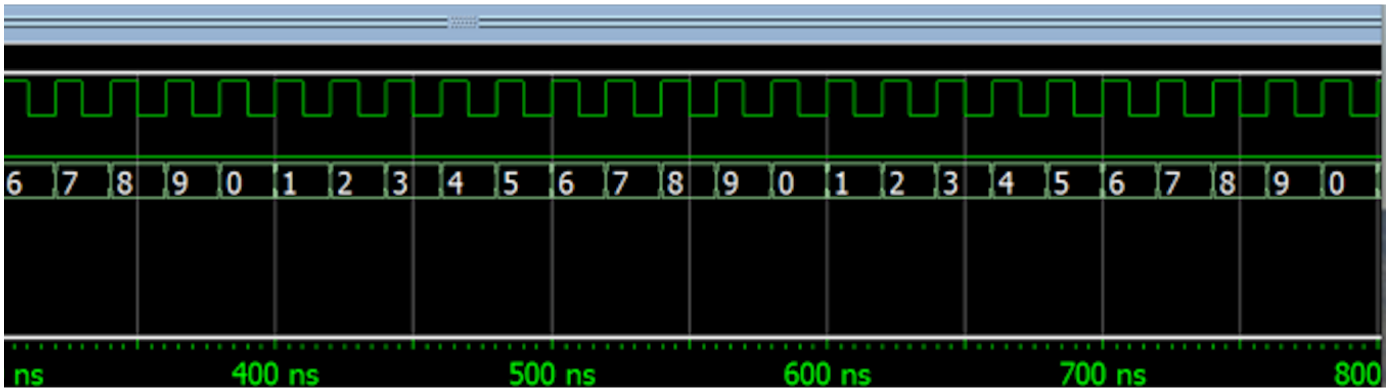
Result

Below is a screenshot of 800 ns of runtime from the testbench described above.



It can be seen in the simulation screenshots that the circuit successfully is initialized with the value 5, increments by 1 on each clock cycle, and goes back to 0 when the output reaches 9. The functionality of the reset can also be seen. Below are two zoomed in snapshots of the above full runtime. It can be seen that the testbench successfully simulates the designed circuit for all possible output patterns. All outputs are shown in hexadecimal.



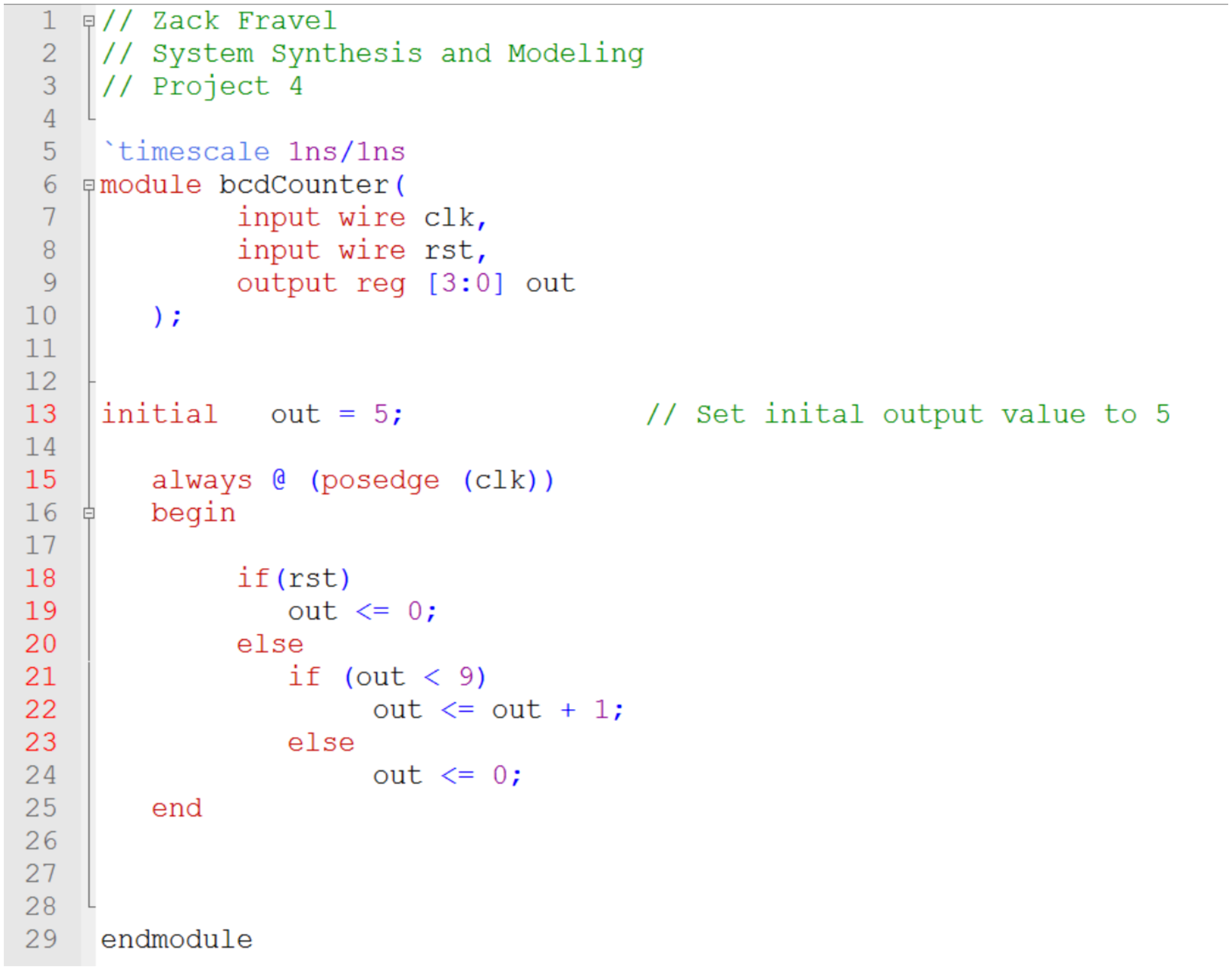


Analysis and Conclusion

The main hurdle I had to overcome when designing the module was the correct syntax and order for my if statements to get the desired behavior, otherwise the circuit was fairly straight forward to design as described in the assignment. For example, for the first time I tried to simulated I had a circuit that counted to the hex value A, or 10 and never reset. Once I corrected my if statement conditions and restructured them, the counter worked properly. In conclusion, what we are left with now is what was assigned originally, we have a verilog module that takes in a clock source as well as a reset input and outputs a BCD up counter with an initial value of 5.

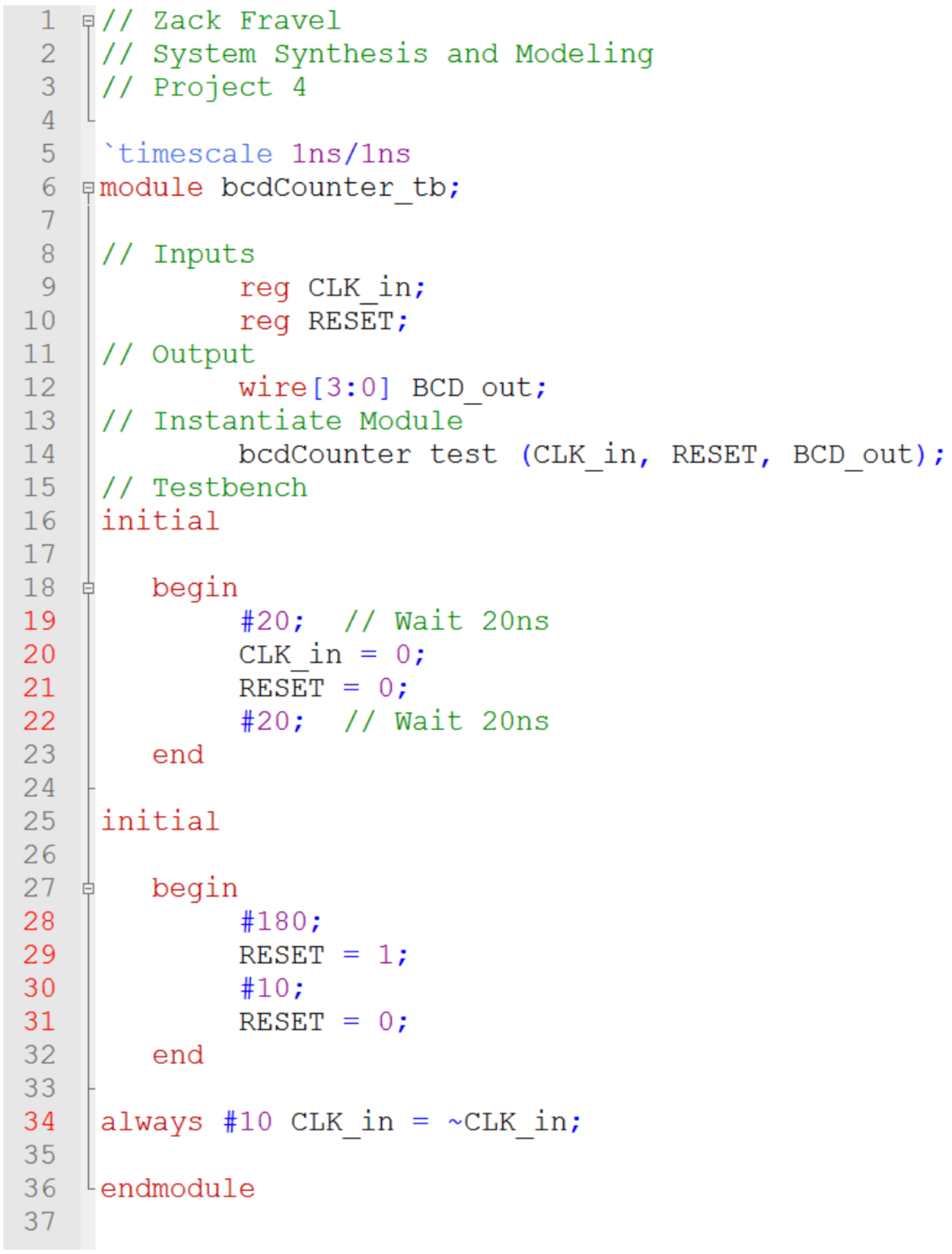
Source Code

bcdCounter.v



```verilog
// Zack Fravel
// System Synthesis and Modeling
// Project 4

`timescale 1ns/1ns
module bcdCounter(
        input wire clk,
        input wire rst,
        output reg [3:0] out
   );


initial   out = 5;              // Set inital output value to 5

   always @ (posedge (clk))
   begin

        if(rst)
           out <= 0;
        else
           if (out < 9)
                out <= out + 1;
           else
                out <= 0;
   end



endmodule
```

bcdCounter_tb.v



```verilog
// Zack Fravel
// System Synthesis and Modeling
// Project 4

`timescale 1ns/1ns
module bcdCounter_tb;

// Inputs
        reg CLK_in;
        reg RESET;
// Output
        wire[3:0] BCD_out;
// Instantiate Module
        bcdCounter test (CLK_in, RESET, BCD_out);
// Testbench
initial

   begin
        #20;  // Wait 20ns
        CLK_in = 0;
        RESET = 0;
        #20;  // Wait 20ns
   end

initial

   begin
        #180;
        RESET = 1;
        #10;
        RESET = 0;
   end

always #10 CLK_in = ~CLK_in;

endmodule
```