ECEN 248: Introduction to Digital Design
Department of Electrical and Computer Engineering
Texas A&M University
Lab Report

Laboratory Exercise #7

Introduction to Sequential Logic

ECEN-248-509

Joaquin Salas

731000141

10-31-2022

TA: Sri Hari Pada Chandanam Kodi

Objectives:

The purpose of this week's lab was to introduce us to sequential logic circuits, to begin we read in the background section for storage elements such as latches and flip-flops. We had the opportunity to describe the components using gate-level in Verilog and simulate all the code after. Secondly, the synchronous sequential circuits will be introduced, we will be combining flip-flops with combinational logic as discussed in previous labs to simulate the operations for synchronous logic. After we will begin using simulation delays in combinational logic and observe the effects it has on the clock timing.

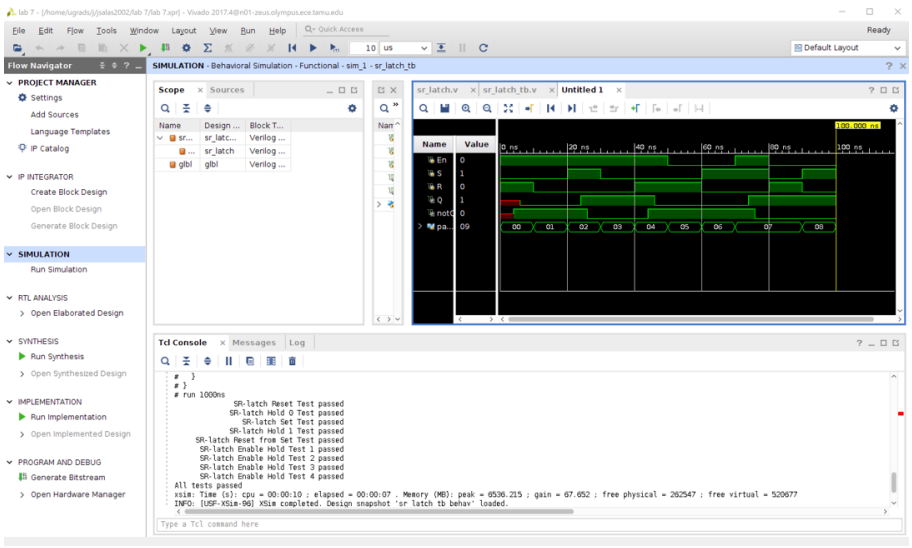
Design:

Experiment Part 1:

We began this lab by using structural Verilog to describe different memory components and simulating it in Vivado. To do this we must be introduced to delays in Verilog code. We will also use behavioral Verilog to describe a D-latch and D flip-flop to create synthesizable synchronous logic.

1. The first is to simulate the operation of the SR-latch.

Result: *SR-latch waveform*

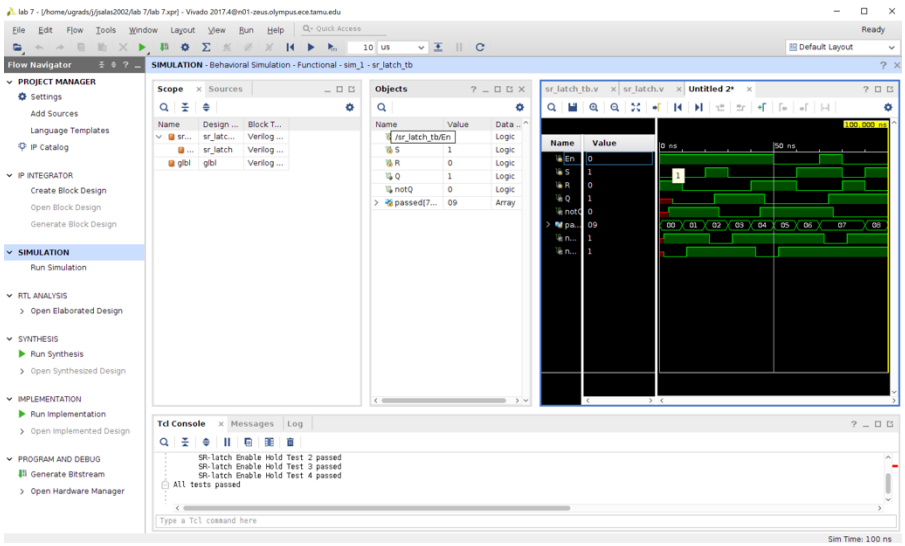


After completing this step we then had to add the internal signals nandSEN and nandREN to the waveform. After doing this step we ran the simulation again using the code shown below for both parts and resulting in the waveform shown below.

Code: *SR_latch*

```
`timescale 1ns / 1ps //delay unit is 1ns
`default_nettype none

module sr_latch(Q, notQ, En, S, R);
    /*port list*/
    output wire Q, notQ;
    input wire En, S, R;

/* intermediate nets */
    wire nandSEN , nandREN;
    nand #4 nand2(nandSEN, S, En);
    nand #4 nand3(nandREN, R, En);

    nand #4 nand0(Q, nandSEN, notQ);
    nand #4 nand1(notQ, nandREN, Q);

endmodule
```

Result: *SR-latch waveform 2*

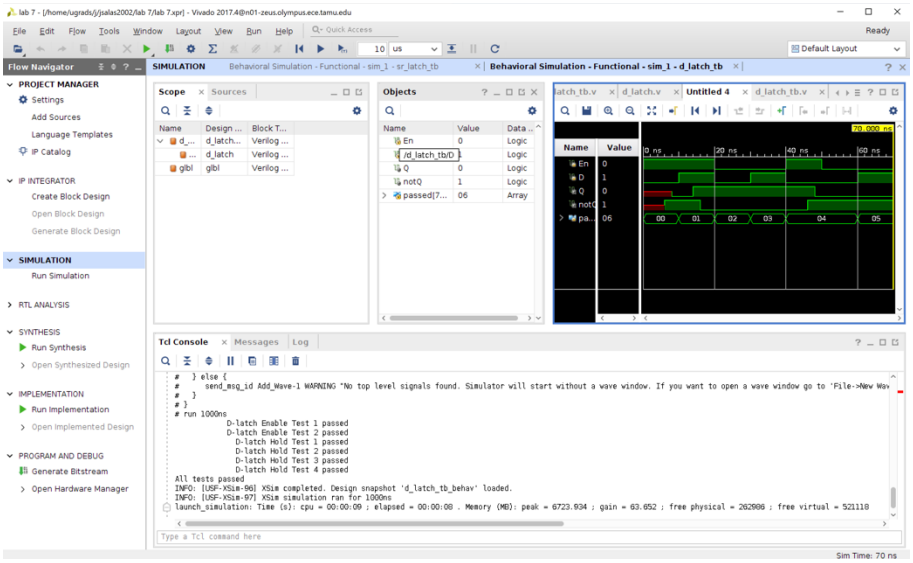


2. Next is to simulate the D-latch using Verilog.

Code: *d_latch*

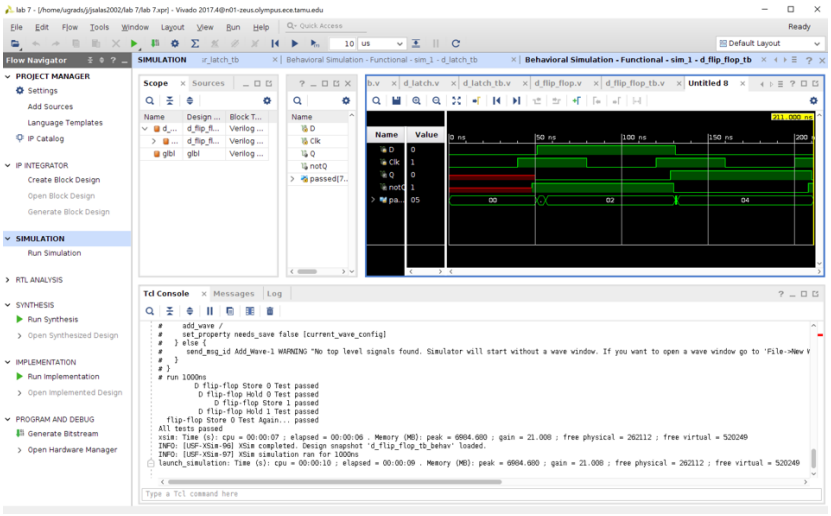
```
`timescale 1ns / 1ps
`default_nettype none

module d_latch_(
    output reg Q,
    output wire notQ,
    input wire D, En
    );

    always@(En or D)
        if(En)
            Q = D;
    assign notQ = ~Q;
endmodule
```

Result: *d_latch waveform*



Add internal signals Qm and notClk to the waveform and run the follow the same process from the last equation for the SR-latch.

Result: *d_latch waveform*



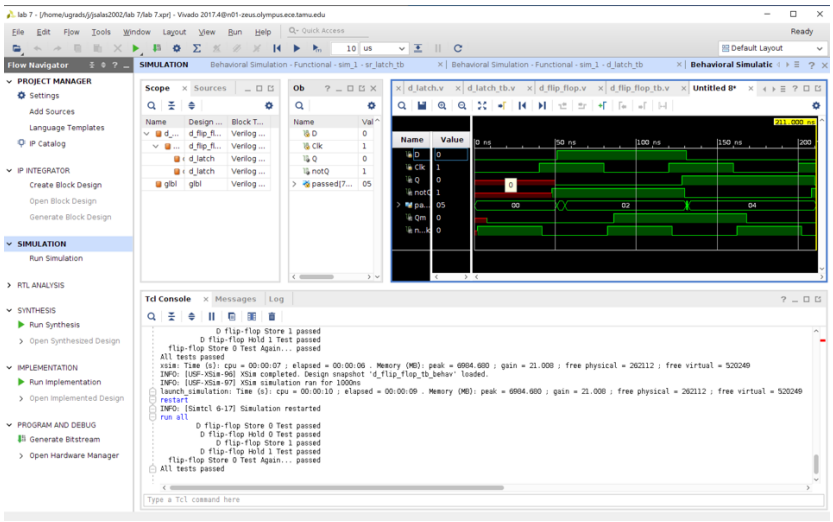
3. Using the D-Latch we just simulated we will construct a D flip-flop.

Code: *d_flip_flop*

```
`timescale 1ns / 1ps  //delay unit is 1ns
`default_nettype none

module d_flip_flop(Q, notQ, Clk, D);
    output wire Q, notQ;
    input wire Clk, D;

    wire notClk, notNotClk;
    wire Qm;
    wire notQm;

    not #2 not0(notClk, Clk);
```

```
    not #2 not1(notNotClk, notClk);

    d_latch d_latch0(Qm, notQm, notClk, D);
    d_latch d_latch1(Q, notQ, notNotClk, Qm);

endmodule
```

Result: *d_latch waveform*

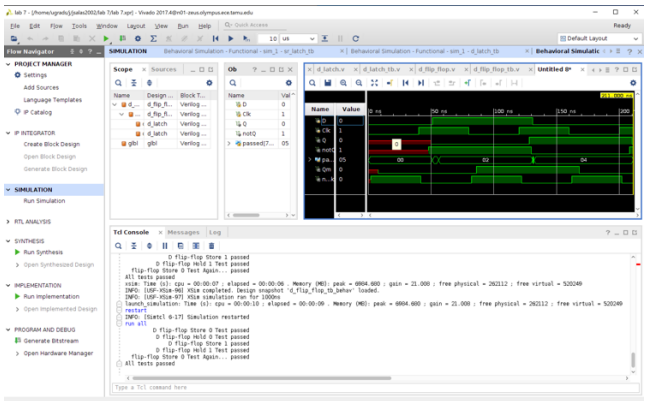


4. Here we are being introduced to the command ‘reg’

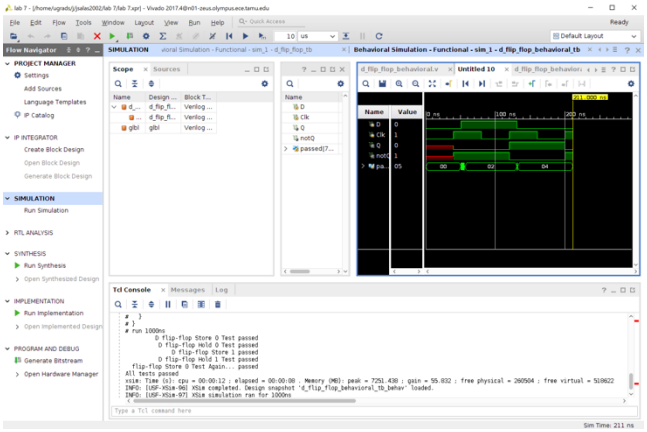
Code: *d_latch_behavioral*

```
`timescale 1ns / 1ps
`default_nettype none

module d_latch_behavioral(
    output reg Q,
    output wire notQ,
    input wire D, En
    );

    always@(En or D)
        if(En)
            Q = D;
    assign notQ = ~Q;
endmodule
```

Result: *d_latch waveform*



Result: *d_latch waveform 2*



Experiment Part 2:

For this part of the experiment, we describe a 2-bit synchronous adder and simulate it in Verilog. We will be using gate-level descriptions in Verilog, so we are able to demonstrate the effects of combinational circuit delay on a synchronous circuit. For comparison, we will show how the same circuit can be described in behavioral Verilog.

1. Use the full adder from the past lab that has already been tested to construct a 2-bit ripple-carry adder.

Code: *2-bit ripple carry adder*

```
`timescale 1ns / 1 ps
`default_nettype none
/*This module describes the gate-level model of *
*a full-adder in Verilog  */

module adder_2bit(Carry, Sum, A, B);

    /*declare output and input ports*/
    //1-bit wires
    output wire Carry;
    input wire C;
    input wire [0:1] A;
    input wire [0:1] B;
    output wire [0:1] Sum;

    full_adder full_adder0(Sum[0], C, A[0], B[0], 1'b0);
    full_adder full_adder1(Sum[1], Carry, A[1], B[1], C);
endmodule
```

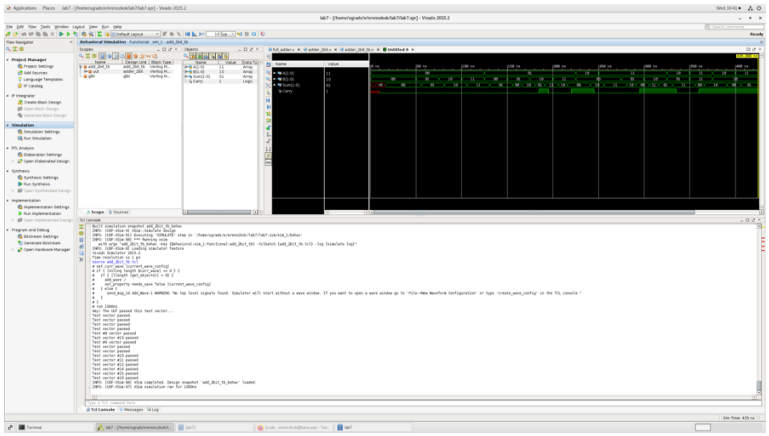
2. For this part of the experiment, we can simplify the experiment, we will use Verilog code to complete the synchronous aspect of this design, assuming flip-flops are ideal and do not contribute to our timing constraints.

Code: *My code for the 2-bit adder test bench*

```
                {A,B} = 4'b0010; // A=00 B=10
        #25;

        if({Carry, Sum} != 3'b010)
            $display("Tests Failed");
        else
            $display("Test vector passed");

                            {A,B} = 4'b0011; // A=00 B=11
    #25;

    if({Carry, Sum} != 3'b011)
        $display("Tests Failed");
     else
        $display("Test vector passed");

       {A,B} = 4'b0100; // A=01 B=00
       #25;

        if({Carry, Sum} != 3'b001)
            $display("Tests Failed");
        else
            $display("Test vector passed");
                            {A,B} = 4'b0101; // A=01 B=01
    #25;

     if({Carry, Sum} != 3'b010)
        $display("Tests #8 Failed");
     else
        $display("Test #8 vector passed");
                                    {A,B} = 4'b0110;
#25;

if({Carry, Sum} != 3'b011)
    $display("Tests #13 Failed");
else
    $display("Test vector #13 passed")
                            {A,B} = 4'b0111; // A=01 B=11
#25;

if({Carry, Sum} != 3'b100)
    $display("Tests #9 Failed");
else
    $display("Test #9 vector passed");
        {A,B} = 4'b1000; // A=10 B=00
        #25;

         if({Carry, Sum} != 3'b010)
            $display("Tests Failed");
         else
             $display("Test vector passed");
                {A,B} = 4'b1100; // A=11 B=00
        #25;

        if({Carry, Sum} != 3'b011)
             $display("Tests Failed");
         else
            $display("Test vector passed");

                {A,B} = 4'b1101; // A=11 B=01
        #25;

        if({Carry, Sum} != 3'b100)
            $display("Tests #10 Failed");
        else
            $display("Test vector #10 passed");

                            {A,B} = 4'b1110;
```

```
    #25;

     if({Carry, Sum} != 3'b101)
         $display("Tests #11 Failed");
     else
         $display("Test vector #11 passed");
                              {A,B} = 4'b1001;
    #25;

     if({Carry, Sum} != 3'b011)
         $display("Tests #12 Failed");
     else
         $display("Test vector #12 passed");

                              {A,B} = 4'b1111;
    #25;

     if({Carry, Sum} != 3'b110)
         $display("Tests #14 Failed");
     else
         $display("Test vector #14 passed");

                              {A,B} = 4'b1010;
    #25;

     if({Carry, Sum} != 3'b100)
         $display("Tests #15 Failed");
     else
         $display("Test vector #15 passed");
                                    {A,B} = 4'b1110;
     #25;

     if({Carry, Sum} != 3'b101)
         $display("Tests #16 Failed");
     else
         $display("Test vector #16 passed");

         //when we are done, let's stop the simulation
         $stop;
end
endmodule
```

Result: *2-bit adder and testbench*



Results:

After concluding this experiment, we showed our TA Sri both working testbench files and he saw that they were in fact working. All our code worked as expected as well as our waveforms and testbench tests were all passed.

Conclusion:

In this lab, we were introduced to time constraints through circuits and experimenting with what different delays do to the circuit, as well as learning to model flip flops and d-latches in behavioral Verilog and continuing to make the waveforms by running the code through the test bench file. We also learned about how testbench files work by writing part of our own.

Post-lab Deliverables:

1. For 1(f) in part 1, explain the results of the simulation.

Changing the delay in the circuit made the waveform lengthen for a longer period of time. This happened because of the delay becoming a greater number making the circuit taking longer to complete.

2. For 3 in part 1, check the waveform with the internal signals. Are the latches behaving as expected? Why or why not?

No, the waveforms did not act as they should have. The d-latch waveform is still oscillating throughout the circuit and not all tests passed through the testbench.

3. For 1.4 in part 1, compare the waveforms you captured from the behavioral Verilog to those from the structural Verilog. Are they different? If so, how?

The waveforms should be the exact same. My waveforms are exactly the same, this is because the same circuit is being simulated, the only difference is that one is in behavioral and the other is in structural Verilog.

4. For 1(e) in part 2, what is the worst-case propagation delay through the adder?

For part 1e in part 2, we had set the worst-case propagation delay to 4ns.

5. Based on the clock period you measured for your synchronous adder, what would be the theoretical maximum clock rate (frequency)? What would be the effect of increasing the width of the adder on the clock rate? How might you improve the clock rate of the design?

The lower clock rate can be controlled by us using Verilog. It can be done by simply decreasing the gate delay, this lets the frequency increase meaning a higher clock rate.

Important Student Feedback

1. What did you like most about the lab assignment and why? What did you like least about it and why?

I enjoyed learning the difference between the three types of ways that code is written in Verilog, structural, dataflow, and behavioral, the pre-lab did take me a fair amount of time but I believe it was much worth the work. I did not like how the Zybo-10 board did not work with our computer at the end of the lab.

2. Were there any sections of the lab manual that were unclear? If so, what was unclear? Do you have any suggestions for improving the clarity?

All parts of the lab manual were very clear, except for the last part on experiment 3 where there was a crucial step that was left out to get the Zybo board to work correctly.

3. What suggestions do you have to improve the overall lab assignment?

I have no suggestions for improving the overall lab assignment.