Objective

In this lab, students were introduced to latches and flip-flops. These sequential logic circuits will be described in first structual Verilog and then behavioral Verilog. Synchronous sequential circuits will also be introduced towards the end of the lab. Delays will also be added to logic gates for the first time while implementing a clock signal. Finally, students will design one block of code that will include both flip-flops and combinational logic (full adder) to simulate synchronous logic.

Design

Experiment 1

To start off the experiment, a structual Verilog SR-Latch was designed. Two nanoseconds delays were added to the gates. After that, the delay was increased to four nanoseconds. Below is the code for which the four nanosecod delay was designed for. For a two nanosecond design, instead of #4, there was #2.

Code Block 1: SR-Latch

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
// Create Date: 14:49:20 10/24/2016
// Module Name: sr_latch
// Author Name: Joseph Martinsen
//
//////////////////////////////////////////////////////////////////

module sr_latch (Q, notQ, En, S, R);
  // All ports should be wires
  output wire Q, notQ;
  input wire En, S, R;

  // Intermediate nets
```

```
  wire nandSEN, nandREN;

  // 4ns delay
  nand #4 nand0(Q, nandSEN, notQ);
  // finish things up here...
  nand #4 nand1(notQ, nandREN, Q);
  nand #4 nand2(nandSEN, S, En);
  nand #4 nand3(nandREN, R, En);
endmodule // sr_latch
```

Next, a structual D-Latch was designed.The design took into consideration that NOT and NAND gates had a two nanosecond delay each. Below is the design for the structual D-Latch.

Code Block 2: D-Latch

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
// Create Date: 15:27:40 10/24/2016
// Module Name: d_latch
// Author Name: Joseph Martinsen
//
//////////////////////////////////////////////////////////////////

module d_latch(Q, notQ, En, D);
  // Declare Wires
    output wire Q;
    output wire notQ;
    input wire En;
    input wire D;

    // Declare Intermidiate Wires
    wire Dnot;
```

```
    wire nandDnotEN;
    wire Dn1;

    // Structual Logic for D-Latch
        not #2 (Dnot, D);
        nand #2 nand0(nandDEN, D, En); //2ns gate delay
        nand #2 nand1(Q, nandDEN, notQ);
        nand #2 nand2 (nandDnotEN, Dnot, En);
        nand #2 nand3 (notQ, Q, nandDnotEN);
endmodule // d_latch
```

The next part of **Experiment 1** consisted of designing a D flip-flop in structual Verilog. The design instantiated only two inverter gates and two D-latch modules that were designed earlier in this lab. The design for the structual D flip-flop.

Code Block 3: D FLip-FLop

```
`timescale 1ns / 1ps
////////////////////////////////////////////////////////////////
// Create Date: 15:44:29 10/24/2016
// Module Name: d_flip_flop
// Author Name: Joseph Martinsen
//
////////////////////////////////////////////////////////////////
module d_flip_flop (Q, notQ, Clk, D);
  output wire Q, notQ;
  input wire Clk, D;

  // Internal nets
  wire notClk, notNotClk;
  wire Q_m; // Output of master latch
  // notQ_m will be wired to the d_latch but then left
  // unconnected from there
```

```
    wire notQ_m;

    // Strcutual level wiring
    // Instantiate and wire up the not gates here...
    not #2 not0(notClk, Clk);
    not #2 not1(notNotClk, notClk);

    // Instantiate and wire up the d latches based
    // on schematic in lab
    d_latch dlMaster(Q_m, notQ_m, notClk, D);
    d_latch dlSlave(Q, notQ, notNotClk, Q_m);
endmodule // d_flip_flop
```

Next, a behavioral version of the D-latch and D flip-flop was copied and edited from the lab manual. The following designs are below. *Compare the waveforms you captured from the behavioral Verilog to those captured from the structural Verilog. Are they different? If so, how?*

Code Block 4: D-Latch Behavioral

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
// Create Date: 16:00:39 10/24/2016
// Module Name: d_latch_behavioral
// Author Name: Joseph Martinsen
//
//////////////////////////////////////////////////////////////////
module d_latch_behavioral(
        // Declare Input/Output Wires
        output reg Q,
        output wire notQ,
        input wire D, En
    );
```

```

        // Logic for D-Latch
         always@(En or D)
         if(En)
                        Q = D;
                else
                        Q = Q;
        // Catch All
        assign notQ = ~Q;
endmodule
```

Code Block 5: D Flip-Flop Behavioral

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
// Create Date: 16:06:56 10/24/2016
// Module Name: d_flip_flop_behavioral
// Author Name: Joseph Martinsen
//
//////////////////////////////////////////////////////////////////
module d_flip_flop_behavioral(
                // Declare wires
                output reg Q,
                output wire notQ,
                input wire D,
                input wire Clk
    );

        // Logic for Positive Edge of Clock
        always@(posedge Clk)
                Q <= D;

```

```
        // Catch all Case
        assign notQ = ~Q;
endmodule
```

Experiment 2

To start off **Experiment 2**, the full adder designed in **Lab 6** was modified. Gate delays were added such that 3-input AND, OR, and XOR gates have a delay of six nanoseconds, while 2-input AND, OR, and XOR gates have a delay of four nanoseconds. Finally, NOT, NAND, and NOR gates were changed in order to have a delay of two nanoseconds.

Code Block 6: Full Adder Edited

```
`timescale 1ns / 1ps
`default_nettype none
//////////////////////////////////////////////////////////////////
// Create Date: 16:06:23 10/10/2016
// Module Name: full_adder
// Author Name: Joseph Martinsen
//
//////////////////////////////////////////////////////////////////
module full_adder(S, Cout, A, B, Cin );
        // Declare input and output ports
        input wire A, B, Cin;
        output wire S, Cout;

        // Declare wi res
        wire andBCin, andACin, andAB; // add more

        // Use dataflow to create gatelevel commands
        assign #6 S = A ^ B ^ Cin; // ^ is XOR
        assign #4 andAB = A & B;
        assign #4 andBCin = B & Cin;
```

```
        assign #4 andACin = A & Cin;
        assign #6 Cout = andAB | andBCin | andACin;
endmodule
```

The edited full adder was then used to create a simple 2-Bit Adder. The design is below.

Code Block 7: 2-Bit Adder

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
// Create Date: 16:13:10 10/24/2016
// Module Name: adder_2bit
// Author Name: Joseph Martinsen
//
//////////////////////////////////////////////////////////////////
module adder_2bit(Carry, Sum, A, B);
        // Initialize Wires
        output wire [1:0] Sum;
        output wire Carry;
        input wire [1:0] A;
        input wire [1:0] B;
        wire Cout;

        // Call 2 Full adders to make it 2-Bit
        full_adder fa0(Sum[0], Cout, A[0], B[0], 0);
        full_adder fa1(Sum[1], Carry, A[1], B[1], Cout );

endmodule
```

The 2-Bit adder test bench was incomplete in order to check the soundness of the newly designed two-bit adder. After adding a test case for every output, the 2-Bit adder was then tested. The modified test bench is below. *Use the simulation waveform to* *determine the worst case propagation delay through the ripple- carry adder.* 20***ns***

Code Block 8: 2-Bit Adder Test Bench

```
`timescale 1ns / 1ps

module add_2bit_tb;

/* Input nets */
reg [1:0] A; //these are regs because they are modified in
reg [1:0] B; //a behavioral block

/* Output nets */
wire [1:0] Sum; //these are wires because they will be driven
wire Carry;//by the inantiated module

/* Instantiate the Unit Under Test (UUT) */
adder_2bit uut ( //this is a different way
.A(A),          //to instantiate a module.
.B(B),          //the nice thing about this style
.Sum(Sum),      //is that the order does not matter!
.Carry(Carry)//notice the ports are in a different order!
);


/*-this is a behavioral block which is executed only once! *
 *-the statements within this behavioral block are executed *
 *-sequentially because we are using blocking statements    *
 *-an '=' sign within a behavioral construct is considered a*
 * blocking statement. We will talk more about this later...*/
initial
  begin
```

```

  /* Initialize inputs*/
  A = 0;
  B = 0;

  #25; //just delay 25 ns
  {A,B} = 4'b0000; //stimulate the inputs
  #25; //wait a bit for the result to propagate
  //here is where we could put a check to see if the results
  //are as expected!
  if({Carry, Sum} != 3'b000)
      $display("Ah crap... something went wrong here...");
            else
                    $display("Hey! The UUT passed");
  //let's do it again with a different input...
  {A,B} = 4'b0001; //stimulate the inputs
  #25; //wait a bit for the result to propagate
  //check output
  if({Carry, Sum} != 3'b001)
    $display("You are garbage!");
            else
                    $display("Test vector passed!!!");
  //okay this is fun... you try it now...

//let's do it again with a different input...
  {A,B} = 4'b0010; //stimulate the inputs
  #25; //wait a bit for the result to propagate
  //check output
  if({Carry, Sum} != 3'b010)
    $display("You are garbage!");
            else
```

```
                    $display("Test vector passed!!!");

    //let's do it again with a different input...
  {A,B} = 4'b0011; //stimulate the inputs
  #25; //wait a bit for the result to propagate
  //check output
  if({Carry, Sum} != 3'b011)
    $display("You are garbage!");
            else
                    $display("Test vector passed!!!");

    //let's do it again with a different input...
  {A,B} = 4'b0100; //stimulate the inputs
  #25; //wait a bit for the result to propagate
  //check output
  if({Carry, Sum} != 3'b001)
    $display("You are garbage!");
            else
                    $display("Test vector passed!!!");

    //let's do it again with a different input...
  {A,B} = 4'b0101; //stimulate the inputs
  #25; //wait a bit for the result to propagate
  //check output
  if({Carry, Sum} != 3'b010)
    $display("You are garbage!");
            else
                    $display("Test vector passed!!!");

    //let's do it again with a different input...
  {A,B} = 4'b0110; //stimulate the inputs
```

```
    #25; //wait a bit for the result to propagate
    //check output
    if({Carry, Sum} != 3'b011)
      $display("You are garbage!");
                else
                        $display("Test vector passed!!!");

        //let's do it again with a different input...
    {A,B} = 4'b0111; //stimulate the inputs
    #25; //wait a bit for the result to propagate
    //check output
    if({Carry, Sum} != 3'b100)
      $display("You are garbage!");
                else
                        $display("Test vector passed!!!");

        //let's do it again with a different input...
    {A,B} = 4'b1000; //stimulate the inputs
    #25; //wait a bit for the result to propagate
    //check output
    if({Carry, Sum} != 3'b010)
      $display("You are garbage!");
                else
                        $display("Test vector passed!!!");

        //let's do it again with a different input...
    {A,B} = 4'b1001; //stimulate the inputs
    #25; //wait a bit for the result to propagate
    //check output
    if({Carry, Sum} != 3'b011)
      $display("You are garbage!");
```

```
                    else
                            $display("Test vector passed!!!");

        //let's do it again with a different input...
    {A,B} = 4'b1010; //stimulate the inputs
    #25; //wait a bit for the result to propagate
    //check output
    if({Carry, Sum} != 3'b100)
      $display("You are garbage!");
                    else
                            $display("Test vector passed!!!");

        //let's do it again with a different input...
    {A,B} = 4'b1011; //stimulate the inputs
    #25; //wait a bit for the result to propagate
    //check output
    if({Carry, Sum} != 3'b101)
      $display("You are garbage!");
                    else
                            $display("Test vector passed!!!");

        //let's do it again with a different input...
    {A,B} = 4'b1100; //stimulate the inputs
    #25; //wait a bit for the result to propagate
    //check output
    if({Carry, Sum} != 3'b011)
      $display("You are garbage!");
                    else
                            $display("Test vector passed!!!");

        //let's do it again with a different input...
```

```
        {A,B} = 4'b1101; //stimulate the inputs
        #25; //wait a bit for the result to propagate
        //check output
        if({Carry, Sum} != 3'b100)
           $display("You are garbage!");
                      else
                                $display("Test vector passed!!!");

            //let's do it again with a different input...
        {A,B} = 4'b1110; //stimulate the inputs
        #25; //wait a bit for the result to propagate
        //check output
        if({Carry, Sum} != 3'b101)
           $display("You are garbage!");
                      else
                                $display("Test vector passed!!!");

        //let's do it again with a different input...
        {A,B} = 4'b1111; //stimulate the inputs
        #25; //wait a bit for the result to propagate
        //check output
        if({Carry, Sum} != 3'b110)
           $display("You are garbage!");
                      else
                                $display("Test vector passed!!!");

        $stop;
end

endmodule
```

In the final part of the lab, Verilog code was created to complete the synchronous aspect of the design. The flip-flops were assumed tob be ideal thus it will not contribute to any time delay.

Code Block 9: Adder Synchronous

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////
// Create Date: 16:53:17 10/24/2016
// Module Name: adder_synchronous
// Author Name: Joseph Martinsen
//
//////////////////////////////////////////////////////////////////
module adder_synchronous(Carry_reg, Sum_reg, Clk, A, B);
	// Declare Wires
	output reg Carry_reg;
	output reg [1:0] Sum_reg;
	input wire Clk;
	input wire [1:0] A,B;
	reg [1:0] A_reg, B_reg;
	wire Carry;
	wire [1:0] Sum;

	// Instantiate 2-Bit Adder
	adder_2bit a2b0(Carry, Sum, A_reg, B_reg);

	// Logice for A and B
	always@(posedge Clk)
		begin
			A_reg <= A;
			B_reg <= B;
		end
```

```
        // Logice for Sum and Carry
        always@(posedge Clk)
                begin
                        Carry_reg <= Carry;
                        Sum_reg <= Sum;
                end
endmodule
```

In the end, ‘define CLOCK PERIOD in the test bench code was changed to 18, 19, and finally 20.

Results

Experiment 1

The first test was for the SR-Latch that had a two second delay for each gate.

```
------------------------------------------------------------------
This is a Full version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
              SR-latch Reset Test passed
             SR-latch Hold 0 Test passed
               SR-latch Set Test passed
             SR-latch Hold 1 Test passed
     SR-latch Reset from Set Test passed
      SR-latch Enable Hold Test 1 passed
      SR-latch Enable Hold Test 2 passed
      SR-latch Enable Hold Test 3 passed
      SR-latch Enable Hold Test 4 passed
All tests passed
Stopped at time : 100 ns :  in File "/home/ugrads/j/josephmart/prelab/sr_latch_tb.v" Line 72
```

Figure 1: *SR Latch Test Results*

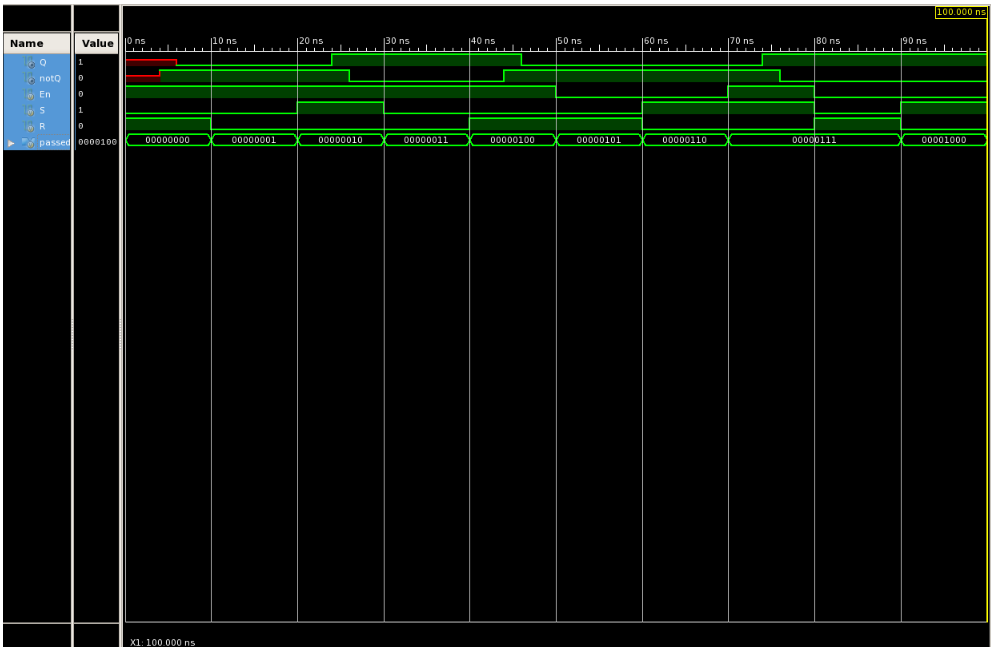

Figure 2: *SR Latch Graph*

The next test was again for the SR-Latch but this time, the gates had a four second delay instead of two. The test results are below.

```
--------------------------------------------------------------------
This is a Full version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
           SR-latch Reset Test failed: X should be 1
          SR-latch Hold 0 Test passed
            SR-latch Set Test failed: 3 should be 2
          SR-latch Hold 1 Test passed
    SR-latch Reset from Set Test failed: 3 should be 1
     SR-latch Enable Hold Test 1 passed
     SR-latch Enable Hold Test 2 passed
     SR-latch Enable Hold Test 3 passed
     SR-latch Enable Hold Test 4 passed
Some tests failed
Stopped at time : 100 ns :  in File "/home/ugrads/j/josephmart/prelab/sr_latch_tb.v" Line 72
```

Figure 3: *SR Latch2 Test Results*

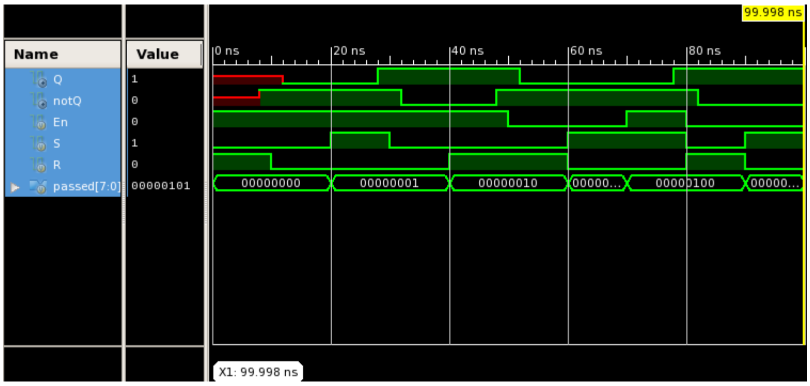

Figure 4: *SR Latch2 Graph*

The test for the four nanosecond delay failed the first four reset , hold, and set tests. For instances where the four delay was in line with the two delay, the test passed

Next, the D-Latch was tested against the provided test bench. The operation matched the given table in the lab. The test results are below.

```
----------------------------------------------------------------------
This is a Full version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
          D-latch Enable Test 1 passed
          D-latch Enable Test 2 passed
            D-latch Hold Test 1 passed
            D-latch Hold Test 2 passed
            D-latch Hold Test 3 passed
            D-latch Hold Test 4 passed
All tests passed
Stopped at time : 70 ns :  in File "/home/ugrads/j/josephmart/prelab/d_latch_tb.v" Line 63
|
```

Figure 5: *D-Latch Test Results*

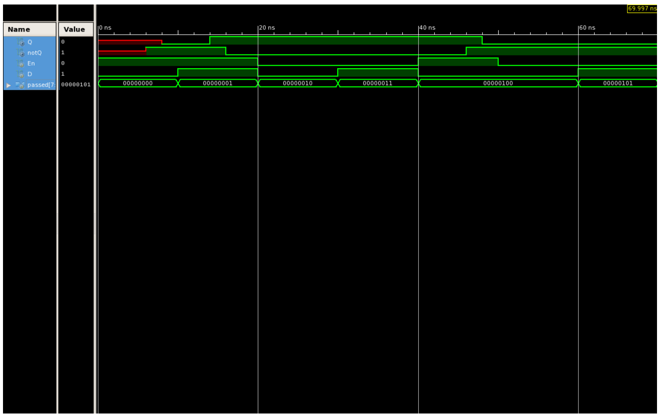

Figure 6: *D-Latch Graph*

The D flip-flop was then tested against the appropriate test bench with internal nets in the waveform.

```
-----------------------------------------------------------------------
This is a Full version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
        D flip-flop Store 0 Test passed
         D flip-flop Hold 0 Test passed
             D flip-flop Store 1 passed
         D flip-flop Hold 1 Test passed
 flip-flop Store 0 Test Again... passed
All tests passed
Stopped at time : 211 ns :  in File "/home/ugrads/j/josephmart/prelab/d_flip_flop_tb.v" Line 71
```

Figure 7: *D Flip-Flop Tests*

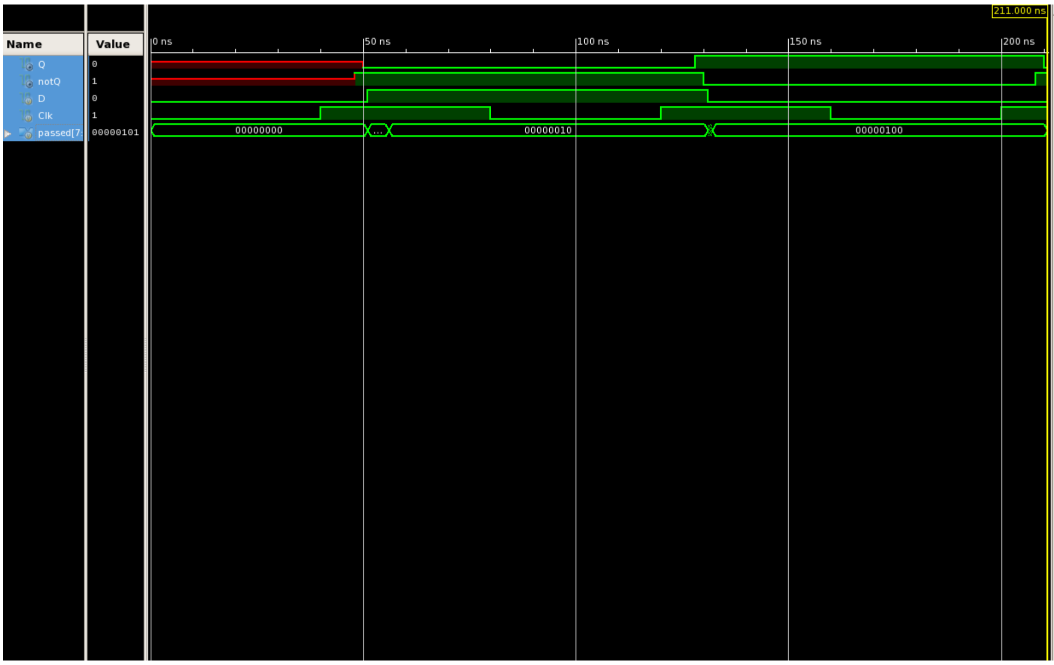

Figure 8: *D Flip-Flop Graph*

The internal nets behaved as expected.

In the final part of **Experiment 1**, the behavioral D-Latch and D flip-flop was tested against their appropriate test benches.

```
--------------------------------------------------------------------
This is a Full version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
        D-latch Enable Test 1 passed
        D-latch Enable Test 2 passed
          D-latch Hold Test 1 passed
          D-latch Hold Test 2 passed
          D-latch Hold Test 3 passed
          D-latch Hold Test 4 passed
All tests passed
Stopped at time : 70 ns :  in File "/home/ugrads/j/josephmart/ecen248/lab08/d_latch_behavioral_tb.v" Line 63
```

Figure 9: *D Latch Behavioral Test Results*

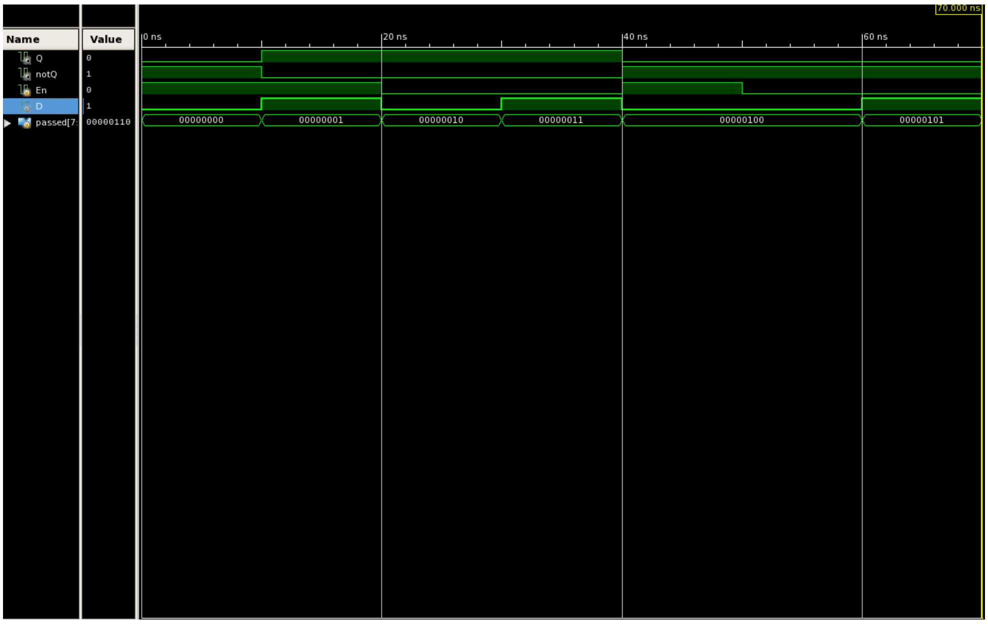

Figure 10: *D Latch Behavioral Graph*

```
--------------------------------------------------------------------
This is a Full version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
        D flip-flop Store 0 Test passed
         D flip-flop Hold 0 Test passed
             D flip-flop Store 1 passed
         D flip-flop Hold 1 Test passed
  flip-flop Store 0 Test Again... passed
All tests passed
Stopped at time : 211 ns :  in File "/home/ugrads/j/josephmart/ecen248/lab08/d_flip_flop_behavioral_tb.v" Line 71
```

Figure 11: *D Flip Flop Behavioral Test Results*

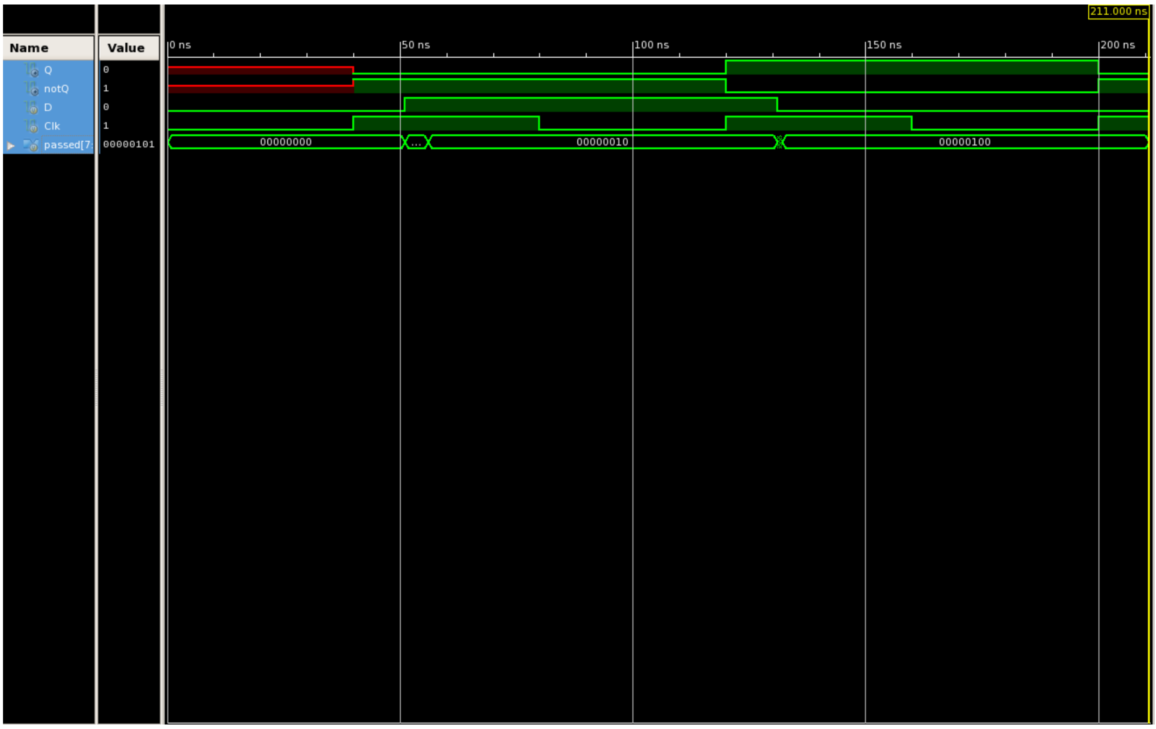

Figure 12: *D Flip Flop Behavioral Graph*

The behavioral and structual D flip-flop design looked identical. The structual and behavioral D-latch differed for the first eight nanoseconds where Q and notQ were switched.

Experiment 2

In **Experiment 2**, the 2-Bit adder was tested against my written test bench. It passed all test vectors.

```
---------------------------------------------------------------------
This is a Full version of ISim.
WARNING:  For instance uut/fa0/, width 1 of formal port Cin is not equal to width 32 of actual constant.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
Hey! The UUT passed this test vector...
Test vector passed!!!
Test vector passed!!!
Test vector passed!!!
Test vector passed!!!
Test vector passed!!!
Test vector passed!!!
Test vector passed!!!
Test vector passed!!!
Test vector passed!!!
Test vector passed!!!
Test vector passed!!!
Test vector passed!!!
Test vector passed!!!
Test vector passed!!!
Test vector passed!!!
Stopped at time : 425 ns :  in File "/home/ugrads/j/josephmart/ecen248/lab08/adder_2bit_tb.v" Line 195
```

Figure 13: *2-bit Adder Test Results*

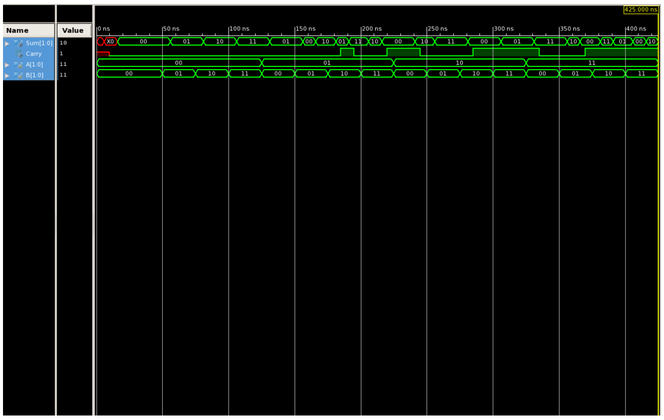

Figure 14: *2-Bit Adder Graph*

Finally, the synchronous adder circuit was tested against the test bench define CLOCK PERIOD being set to 40. Then with 18, 19, and 20. 40, 18, and 20 are shown below.

```
----------------------------------------------------------------------
This is a Full version of ISim.
WARNING:  For instance a2b0/fa0/, width 1 of formal port Cin is not equal to width 32 of actual constant.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test passed
All tests passed
Stopped at time : 660 ns(1) :  in File "/home/ugrads/j/josephmart/ecen248/lab08/adder_synchronous_tb.v" Line 78
```

Figure 15: *Asyc 40 Tests*

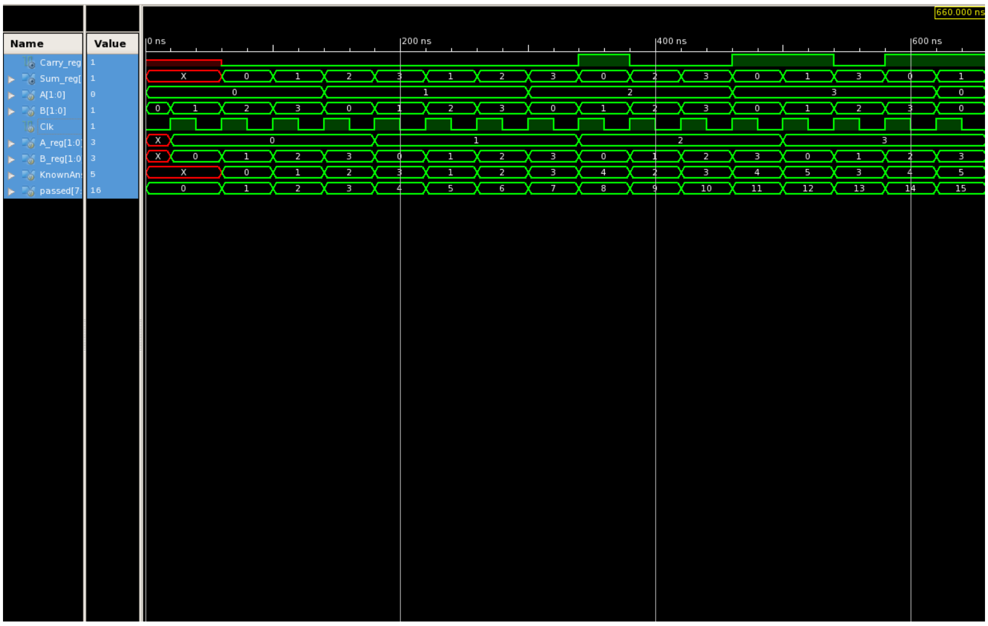

Figure 16: *Asyc 40 Graph*

```
This is a Full version of ISim.
WARNING:  For instance a2b0/fa0/, width 1 of formal port Cin is not equal to width 32 of actual constant.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test failed: 07 should be 03
        Synchronous Adder Test failed: 00 should be 04
        Synchronous Adder Test failed: 06 should be 02
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test failed: 00 should be 04
        Synchronous Adder Test passed
Some tests failed
Stopped at time : 297 ns(1) :  in File "/home/ugrads/j/josephmart/ecen248/lab08/adder_synchronous_tb.v" Line 78
```

Figure 17: *Asyc 18 Tests*

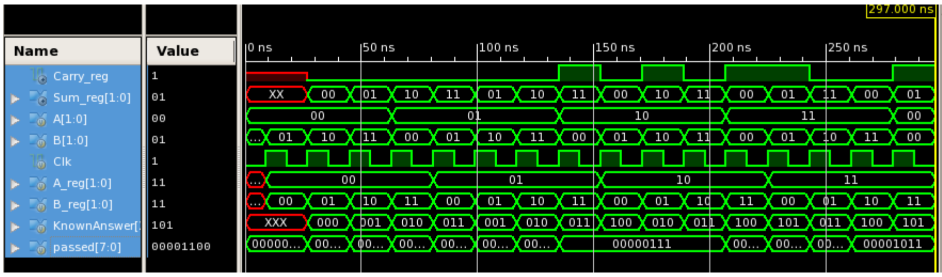

Figure 18: *Asyc 18 Graph*

```
This is a Full version of ISim.
WARNING: For instance a2b0/fa0/, width 1 of formal port Cin is not equal to width 32 of actual constant.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test passed
        Synchronous Adder Test passed
All tests passed
Stopped at time : 330 ns(1) :  in File "/home/ugrads/j/josephmart/ecen248/lab08/adder_synchronous_tb.v" Line 78
```

Figure 19: *Asyc 20 Tests*

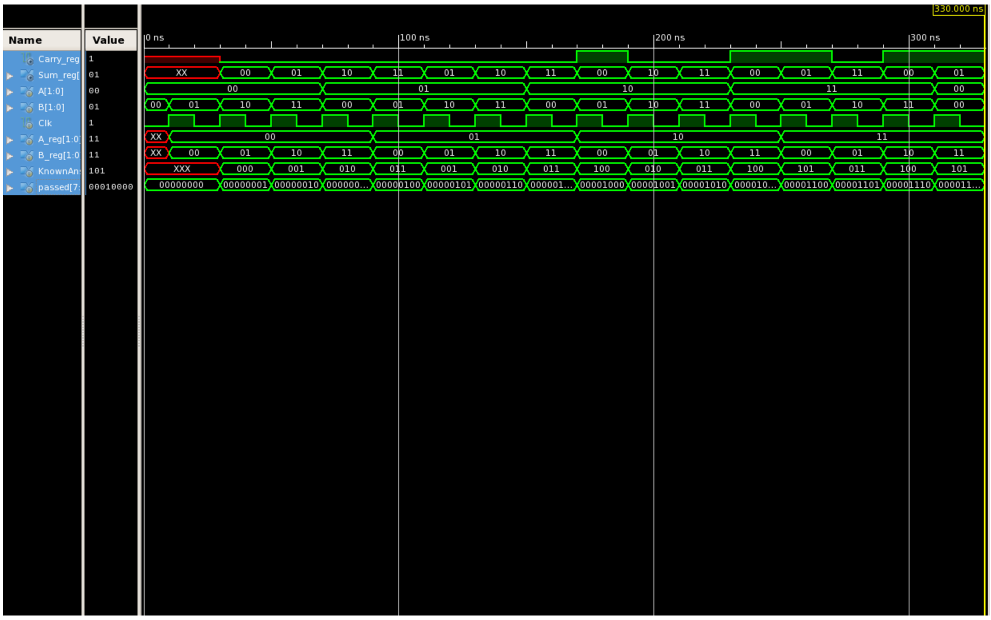

Figure 20: *Asyc 20 Graph*

The test passed for 40 and 20 and failed for 18 and 19. Thus the shortest ‘$defineCLOCKPERIOD = 20ns$

Conclution

In conclution, this lab exposed me to test bench code by allowing me to create my own for the first time. Also, this lab introduced me to three different type of latches, SR latch, D latch, and D flip-flop latch.

Questions

1. **Include the source code with comments for all modules you simulated. You do not have to include test bench code that was provided; however, you must supply the test bench code that you wrote! Code without comments will not be accepted.**

   In the report.

2. **Include screenshots of all waveforms captured during simulatio in addition to the test bench console output for each test bench simulation.**

   In the report.

3. **Answer <u>all</u> questions throughout the lab manual.**

   In the report.

4. **Compare the behavioral description of the synchronous adder found in the test bench code with the combination of stuctual and dataflow Verilog you used in the lab assignment. What are the advantages and disadvantages of each? Which do you perfer and why?**

   The advantage of utilizing a behavioral description is that it is very simple and to the point because it uses one statement that then decides what kind of output it is going to be. In structural and dataflow Verilog it is precise, but not all that concise. Efficiency is lost because it must look for each specific output rather than just paying attention to the output behavior as a whole. I prefer behavioral Verilog because it seems like it is a whole lot more efficient.

5. **Based on the clock period you measured for your synchronous adder, what would be the theoretical maximum clock rate? 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 smallest ‘define CLOCK PERIOD that the circuit could of been reduced to and still function was 20 ns which works at a 50MHz clock rate. Increasing the width of the adder would cause the clock rate to drop because it has to work with more inputs. This would result in a greater time delay which would result in a drop in clock rate. Improving the clock rate of the design would be done by decreasing the time delay in the circuit by simplifying the circuit have to use less or by using gates that have a smaller gate delay.

Student Feedback

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

   I enoyed learning about test bench workings. I also liked practicing more with Verilog.

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

   This lab was very straight forward and provided more guidance than most of the previose labs.

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

   Synthesising the designs onto the Spartan board would of been cool.