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

Code Block 1: 4-Bit ALU

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    14:49:20 10/24/2016
// Design Name:
// Module Name:    sr_latch
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
```

```
//
/////////////////////////////////////////////////////////////////////////////

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;

  // woah!!! what is this #2 thing?!?
  // it's a delay (simulation only!!!)
  nand #4 nand0(Q, nandSEN, notQ); //2ns gate delay
  // 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
```

Code Block 2: 4-Bit ALU

```
`timescale 1ns / 1ps
/////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    15:27:40 10/24/2016
// Design Name:
// Module Name:    d_latch
// Project Name:
```

```
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////

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

    wire Dnot;
    wire nandDnotEN;
    wire Dn1;

            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
```

Code Block 3: 4-Bit ALU

```
`timescale 1ns / 1ps
```

```
///////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    15:44:29 10/24/2016
// Design Name:
// Module Name:    d_flip_flop
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
///////////////////////////////////////////////////////////////////////////////

// Use structural Verilog, teh buil-in gate-level
// primitives, and our D-latch to construct the D flip-flop descirbed in lc

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
  wire notQ_m; // notQ_m will be wired to the d_latch but then left unconn

  // 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
```

Code Block 4: 4-Bit ALU

```
`timescale 1ns / 1ps
///////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    16:00:39 10/24/2016
// Design Name:
// Module Name:    d_latch_behavioral
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////
module d_latch_behavioral(
          output reg Q,
          output wire notQ,
          input wire D, En
     );

           always@(En or D)
           if(En)
                    Q = D;
          else
                    Q = Q;

          assign notQ = ~Q;


endmodule
```

Code Block 5: 4-Bit ALU

```
`timescale 1ns / 1ps
////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    16:06:56 10/24/2016
// Design Name:
// Module Name:    d_flip_flop_behavioral
// Project Name:
// Target Devices:
```

```
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module d_flip_flop_behavioral(
                    output reg Q,
                    output wire notQ,
                    input wire D,
                    input wire Clk
     );

          always@(posedge Clk)
                    Q <= D;

          assign notQ = ~Q;

endmodule
```

Experiment 2

Code Block 6: 4-Bit ALU

```
`timescale 1ns / 1ps
`default_nettype none
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
```

```
//
// Create Date:    16:06:23 10/10/2016
// Design Name:
// Module Name:    full_adder
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
/////////////////////////////////////////////////////////////////////////////
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
```

Code Block 7: 4-Bit ALU

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    16:13:10 10/24/2016
// Design Name:
// Module Name:    adder_2bit
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module adder_2bit(Carry, Sum, A, B);
                output wire [1:0] Sum;
                output wire Carry;

                input wire [1:0] A;
                input wire [1:0] B;
                wire Cout;

                full_adder fa0(Sum[0], Cout, A[0], B[0], 0);
                full_adder fa1(Sum[1], Carry, A[1], B[1], Cout );

endmodule
```

Code Block 8: 4-Bit ALU

```
`timescale 1ns / 1ps

module add_2bit_tb;//a test bench does not have any ports of its own!

        /* 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)//you could put your own message here
            $display("Ah crap... something went wrong here...");
                        else
                                $display("Hey! The UUT passed this test vec
        //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!!!");

        //go through all possible input combinations (2^4 = 16)
        //cut and paste makes this task a lot easier

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

endmodule
```

Code Block 9: 4-Bit ALU

```
`timescale 1ns / 1ps
/////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    16:53:17 10/24/2016
// Design Name:
// Module Name:    adder_synchronous
// Project Name:
```

```
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
/////////////////////////////////////////////////////////////////////////////////
module adder_synchronous(Carry_reg, Sum_reg, Clk, A, B);

        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;

        adder_2bit  a2b0(Carry, Sum, A_reg, B_reg);


        always@(posedge Clk)
                begin
                        A_reg <= A;
                        B_reg <= B;
```

```
			end


		always@(posedge Clk)
			begin
				Carry_reg <= Carry;
				Sum_reg <= Sum;
			end


endmodule
```

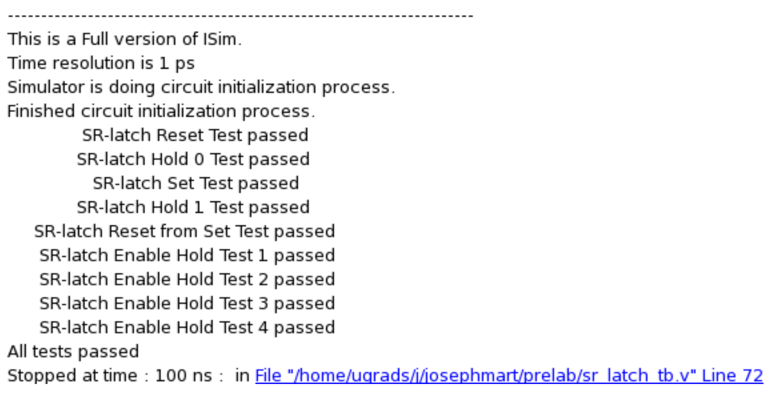
Results

Explain the 2 unit delay to 4 units and explain the results of the simulation (Experiment 1 1.e)

(Experiment 1 3.b) Do the latches behave as expected? Why or why not?

(Experiment 1 4.b) Compare the waveforms you captured from the behavioral Verilog to those captured from the structual. Are they different? If so, how?

Experiment 1



```
----------------------------------------------------------------------
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*

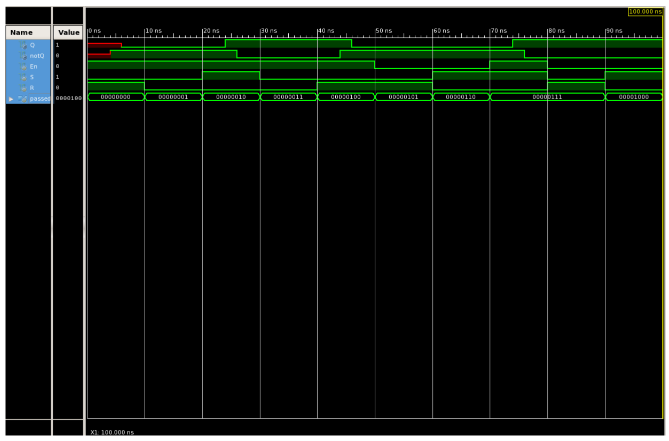
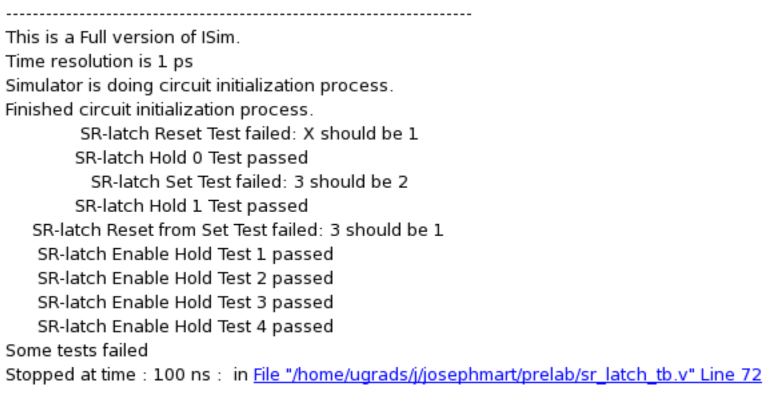


Figure 2: *SR Latch Graph*



```
----------------------------------------------------------------------
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*

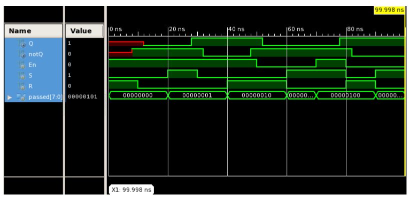


Figure 4: *SR Latch2 Graph*

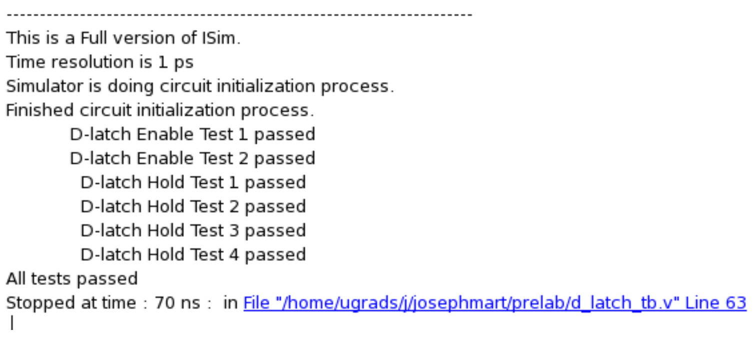

```
----------------------------------------------------------------------
This is a Full version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
          D-latch Enable Test 1 passed
          D-latch Enable Test 2 passed
            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*

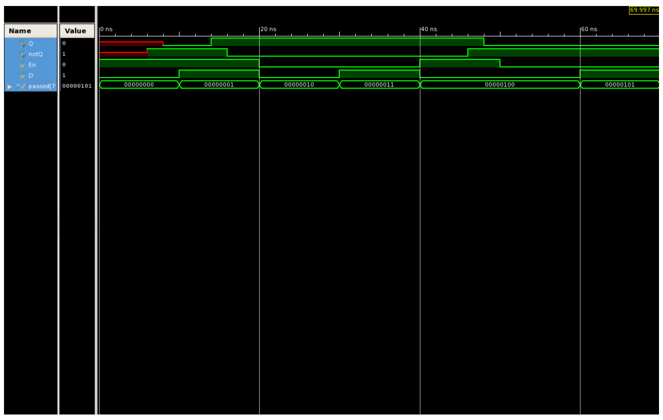
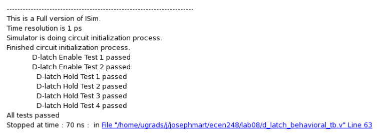


Figure 6: *D Latch Graph*



```
--------------------------------------------------------------------
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 7: *D Latch Behavioral Test Results*

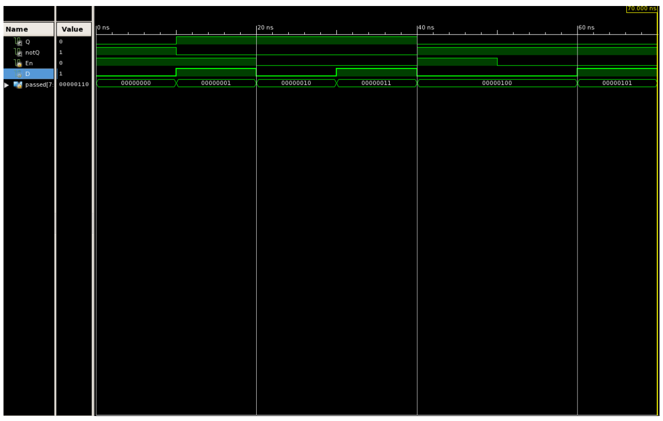
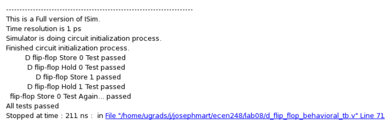


Figure 8: *D Latch Behavioral Graph*



```
--------------------------------------------------------------------
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 9: *D Flip Flop Behavioral Test Results*

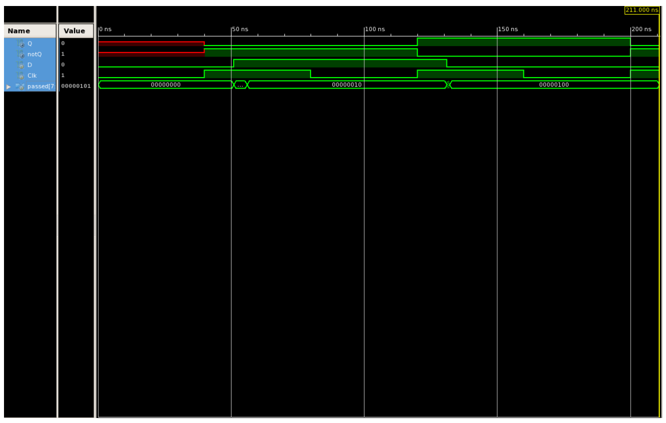


Figure 10: *D Flip Flop Behavioral Graph*

Conclutions

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 all 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?**

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?**

Student Feedback

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

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?**

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