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
1
  // Company:
3
  // Engineer:
5
                   14:49:20 10/24/2016
6
  // Create Date:
  // Design Name:
7
  // Module Name:
8
                   sr_{-}latch
9
  // Project Name:
  // Target Devices:
10
  // Tool versions:
11
  // Description:
12
13
  // Dependencies:
14
15
  // Revision:
16
  // Revision 0.01 - File Created
17
 // Additional Comments:
```

```
19
  20
21
  module sr_latch (Q, notQ, En, S, R);
22
23
    // All ports should be wires
24
    output wire Q, notQ;
    input wire En, S, R;
25
26
    // Intermediate nets
27
28
    wire nandSEN, nandREN;
29
30
    // woah!!! what is this #2 thing?!?
    // it's a delay (simulation only!!!)
31
    nand #4 nand0(Q, nandSEN, notQ); //2ns gate delay
32
    // finish things up here...
33
34
    nand #4 nand1 (notQ, nandREN, Q);
35
    nand #4 nand2(nandSEN, S, En);
36
    nand #4 nand3(nandREN, R, En);
37
38
  endmodule // sr_latch
```

Code Block 2: 4-Bit ALU

```
// Target Devices:
  // Tool versions:
11
  // Description:
12
13
  // Dependencies:
14
15
16
  // Revision:
  // Revision 0.01 - File Created
  // Additional Comments:
18
19
  //
  20
21
22
  module d_latch(Q, notQ, En, D);
23
     output wire Q;
24
     output wire notQ;
25
     input wire En;
26
     input wire D;
27
     wire Dnot;
28
29
     wire nandDnotEN;
30
     wire Dn1;
31
32
          not #2 (Dnot, D);
33
          nand #2 nand0(nandDEN, D, En); //2ns gate delay
34
          nand #2 nand1(Q, nandDEN, notQ);
          nand #2 nand2 (nandDnotEN, Dnot, En);
35
36
          nand #2 nand3 (notQ, Q, nandDnotEN);
37
  endmodule // d_l a t c h
```

Code Block 3: 4-Bit ALU

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

```
// Company:
3
4 | // Engineer:
  //
5
  // Create Date:
                  15:44:29 10/24/2016
6
7 // Design Name:
  // Module Name:
                   d_-flip_-flop
  // Project Name:
  // Target Devices:
10
  // Tool versions:
11
12
  // Description:
13
  //
  // Dependencies:
14
  //
15
  // Revision:
16
  // Revision 0.01 - File Created
17
  // Additional Comments:
18
  //
19
  20
21
22
  // Use structual Verilog, teh buil-in gate-level
  // primitives, and our D-latch to construct the D flip-flop descirbed in la
23
24
  module d_flip_flop (Q, notQ, Clk, D);
25
26
    output wire Q, notQ;
    input wire Clk, D;
27
28
    // Internal nets
29
30
    wire notClk, notNotClk;
    wire Qm; // Output of master latch
31
    wire notQ_m; // notQ_m will be wired to the d_latch but then left unconn
32
```

```
33
     // Strcutual level wiring
34
     // Instantiate and wire up the not gates here...
35
     not #2 not0(notClk, Clk);
36
     not #2 not1(notNotClk, notClk);
37
38
39
     // Instantiate and wire up the d latches based on schematic in lab
40
     d_latch dlMaster(Q_m, notQ_m, notClk, D);
41
     d_latch dlSlave(Q, notQ, notNotClk, Q.m);
42
43
44
   endmodule // d_-flip_-flop
```

Code Block 4: 4-Bit ALU

```
'timescale 1ns / 1ps
1
  // Company:
3
  // Engineer:
4
5
  // Create Date:
                   16:00:39 10/24/2016
6
7 // Design Name:
  // Module Name:
                   d_{-}latch_{-}behavioral
  // Project Name:
  // Target Devices:
10
  // Tool versions:
11
12 | // Description:
  //
13
14 | // Dependencies:
15
  16 \ // \ Revision:
17 // Revision 0.01 - File Created
```

```
// Additional Comments:
19
  20
  module d_latch_behavioral(
22
         output reg Q,
23
         output wire notQ,
24
         input wire D, En
25
      );
26
27
          always@(En or D)
28
          if(En)
               Q = D;
29
         else
30
               Q = Q;
31
32
         assign notQ = ^{\sim}Q;
33
34
35
  endmodule
36
```

Code Block 5: 4-Bit ALU

```
// Tool versions:
  // Description:
12
13
  // Dependencies:
14
15
16
  // Revision:
  // Revision 0.01 - File Created
  // Additional Comments:
18
19
  //
20
  module d_flip_flop_behavioral(
21
22
                 output reg Q,
23
                 output wire notQ,
24
                 input wire D,
25
                 input wire Clk
      );
26
27
          always@(posedge Clk)
28
29
                 Q \leq D;
30
          assign notQ = ^{\sim}Q;
31
32
33
  endmodule
```

Experiment 2

Code Block 6: 4-Bit ALU

```
6
   // Create Date:
                    16:06:23 10/10/2016
8 | // Design Name:
   // Module Name:
                        full_a adder
   // Project Name:
10
11
   // Target Devices:
   // Tool versions:
   // Description:
13
14 //
   // Dependencies:
15
16 //
   // Revision:
17
   // Revision 0.01 - File Created
18
   // Additional Comments:
   //
20
21
   module full_adder(S, Cout, A, B, Cin);
22
23
            // Declare input and output ports
            input wire A, B, Cin;
24
25
            output wire S, Cout;
26
27
            // Declare wi res
            \mathbf{wire} \ \ \mathrm{andBCin} \,, \ \ \mathrm{andACin} \,, \ \ \mathrm{andAB} \,; \ \ // \ \ \mathit{add} \ \ \mathit{more}
28
29
30
            // Use dataflow to create gatelevel commands
            assign #6 S = A \hat{} B \hat{} Cin; // \hat{} is XOR
31
32
            assign #4 and AB = A \& B;
33
            assign #4 and BCin = B \& Cin;
34
            assign #4 and ACin = A \& Cin;
            assign #6 Cout = andAB | andBCin | andACin;
35
36
```

```
37 | 38 | 39 | endmodule
```

Code Block 7: 4-Bit ALU

```
1
  'timescale 1ns / 1ps
2
  3 // Company:
4 | // Engineer:
  //
5
6 // Create Date:
                   16:13:10 10/24/2016
7 // Design Name:
  // Module Name:
                   adder_{-}2bit
8
9
  // Project Name:
10
  // Target Devices:
  // Tool versions:
11
  // Description:
12
13
  // Dependencies:
  //
15
  // Revision:
16
  // Revision 0.01 - File Created
17
  // Additional Comments:
18
  //
19
  20
  {\bf module}\ {\tt adder\_2bit}\,(\,{\tt Carry}\,,\ {\tt Sum},\ A,\ B\,)\,;
21
22
                output wire [1:0] Sum;
23
                output wire Carry;
24
                input wire [1:0] A;
25
26
                input wire [1:0] B;
```

Code Block 8: 4-Bit ALU

```
'timescale 1ns / 1ps
1
2
   module add_2bit_tb; //a test bench does not have any ports of its own!
3
4
           /* Input nets */
5
           reg [1:0] A; //these are regs because they are modified in
6
           reg [1:0] B; //a behavioral block
7
8
           /* Output nets */
9
10
           wire [1:0] Sum; //these are wires because they will be driven
           wire Carry; //by the inantiated module
11
12
           /* Instantiate the Unit Under Test (UUT) */
13
           adder_2bit uut ( //this is a different way
14
                                 //to instantiate a module.
                    A(A),
15
                                 //the nice thing about this style
16
                    B(B),
                                 //is that the order does not matter!
17
                    . Sum(Sum),
18
                    . Carry (Carry) // notice the ports are in a different order!
19
           );
20
21
       /*-this is a behavioral block which is executed only once!
22
```

 $*-the\ statements\ within\ this\ behavioral\ block\ are\ executed\ *$

23

```
24
        *-sequentially because we are using blocking statements
        *-an '= ' sign within a behavioral construct is considered a*
25
        * blocking statement. We will talk more about this later...*/
26
            initial
27
         begin
28
29
           /* Initialize inputs*/
30
           A = 0;
31
32
           B = 0;
33
34
            #25; //just delay 25 ns
35
            \{A,B\} = 4'b0000; //stimulate the inputs
            #25; //wait a bit for the result to propagate
36
            //here is where we could put a check to see if the results
37
            //are as expected!
38
39
            if(\{Carry, Sum\} = 3'b000)//you could put your own message here
40
                $\display("Ah_crap \ldots something went wrong here \ldots");
                             else
41
42
                                      $\display("Hey! \_The\_UUT\_passed \_this\_test\_vectores
            //let's do it again with a different input...
43
44
            \{A,B\} = 4'b0001; //stimulate the inputs
            #25; //wait a bit for the result to propagate
45
            //check output
46
            if ({ Carry, Sum} != 3'b001)
47
48
                $display("You_are_garbage!");
                             else
49
                                      $display("Test_vector_passed!!!");
50
            //okay this is fun... you try it now...
51
52
                       //let's do it again with a different input...
53
            \{A,B\} = 4'b0010; //stimulate the inputs
54
```

```
#25; //wait a bit for the result to propagate
55
            //check output
56
            if ({ Carry, Sum} != 3'b010)
57
                $display("You_are_garbage!");
58
                             else
59
                                      $display("Test_vector_passed!!!");
60
61
62
63
                             //let's do it again with a different inp\psi t...
            \{A,B\} = 4'b0011; //stimulate the inputs
64
65
            #25; //wait a bit for the result to propagate
            //check output
66
            if ({ Carry, Sum} != 3'b011)
67
                $display("You_are_garbage!");
68
                             else
69
                                      $display("Test_vector_passed!!!");
70
71
72
                             //let's do it again with a different input...
73
74
            \{A,B\} = 4'b0100; //stimulate the inputs
            #25; //wait a bit for the result to propagate
75
            //check output
76
77
            if ({ Carry, Sum} != 3'b001)
                $display("You_are_garbage!");
78
79
                             else
                                      $display("Test_vector_passed!!!");
80
81
82
83
                             //let's do it again with a different inp\psi t...
            \{A,B\} = 4'b0101; //stimulate the inputs
84
            #25; //wait a bit for the result to propagate
85
```

```
86
             //check output
             if({Carry, Sum} != 3'b010)
87
                 $display("You_are_garbage!");
88
                              else
89
90
                                       $display("Test_vector_passed!!!");
91
92
                              //let's do it again with a different inp\psi t...
93
             \{A,B\} = 4'b0110; //stimulate the inputs
94
             #25; //wait a bit for the result to propagate
95
             //check output
96
             if ({ Carry, Sum} != 3'b011)
97
                 $display("You_are_garbage!");
                              else
98
                                       $display("Test_vector_passed!!!");
99
100
101
                              //let's do it again with a different input...
102
             \{A,B\} = 4'b0111; //stimulate the inputs
103
104
             #25; //wait a bit for the result to propagate
105
             //check output
             if ({ Carry, Sum} != 3'b100)
106
                 $display("You_are_garbage!");
107
108
                              else
                                       $display("Test_vector_passed!!!");
109
110
111
112
                              //let 's do it again with a different inp\psi t \dots
             \{A,B\} = 4'b1000; //stimulate the inputs
113
114
             #25; //wait a bit for the result to propagate
            //check output
115
             if ({ Carry, Sum} != 3'b010)
116
```

```
117
                 $display("You_are_garbage!");
118
                               else
                                       $display("Test_vector_passed!!!");
119
120
121
122
                              //let's do it again with a different input...
             \{A,B\} = 4'b1001; //stimulate the inputs
123
124
             #25; //wait a bit for the result to propagate
125
             //check output
126
             if ({ Carry, Sum} != 3'b011)
127
                 $display("You_are_garbage!");
128
                               else
                                       $display("Test_vector_passed!!!");
129
130
131
132
                              //let 's do it again with a different inp\psi t \dots
             \{A,B\} = 4'b1010; //stimulate the inputs
133
             #25; //wait a bit for the result to propagate
134
             //check output
135
             if ({ Carry, Sum} != 3'b100)
136
                 $display("You_are_garbage!");
137
                              else
138
                                       $display("Test_vector_passed!!!");
139
140
141
142
                              //let 's do it again with a different inp\psi t \dots
             \{A,B\} = 4'b1011; //stimulate the inputs
143
144
             #25; //wait a bit for the result to propagate
145
             //check output
             if ({ Carry, Sum} != 3'b101)
146
                 $display("You_are_garbage!");
147
```

```
148
                               else
                                       $display("Test_vector_passed!!!");
149
150
151
                              //let's do it again with a different input...
152
             \{A,B\} = 4'b1100; //stimulate the inputs
153
             #25; //wait a bit for the result to propagate
154
             //check output
155
156
             if ({ Carry, Sum} != 3'b011)
157
                 $display("You_are_garbage!");
158
                               else
                                       $display("Test_vector_passed!!!");
159
160
161
162
                              //let 's do it again with a different inp\psi t \dots
             \{A,B\} = 4'b1101; //stimulate the inputs
163
164
             #25; //wait a bit for the result to propagate
             //check output
165
             if ({ Carry, Sum} != 3'b100)
166
167
                 $display("You_are_garbage!");
168
                               else
                                        $display("Test_vector_passed!!!");
169
170
171
172
                              //let 's do it again with a different inp\psi t \dots
             \{A,B\} = 4'b1110; //stimulate the inputs
173
             #25; //wait a bit for the result to propagate
174
175
             //check output
176
             if ({ Carry, Sum} != 3'b101)
                 $display("You_are_garbage!");
177
                               else
178
```

```
$display("Test_vector_passed!!!");
179
180
181
182
             //let's do it again with a different input...
183
             \{A,B\} = 4'b1111; //stimulate the inputs
             #25; //wait a bit for the result to propagate
184
185
             //check output
186
             if ({ Carry, Sum} != 3'b110)
187
                 $display("You_are_garbage!");
188
                              else
189
                                       $display("Test_vector_passed!!!");
190
191
            //go\ through\ all\ possible\ input\ combinations\ (2^4 = 16)
192
             //cut and paste makes this task a lot easier
193
194
             //when we are done, let's stop the simulation
195
             $stop;
196
             end
197
198
    endmodule
```

Code Block 9: 4-Bit ALU

```
10 // Target Devices:
  // Tool versions:
11
  // Description:
12
13
  // Dependencies:
14
15
  //
  // Revision:
16
17
  // Revision 0.01 - File Created
  // Additional Comments:
18
19
  //
  20
  module adder_synchronous(Carry_reg, Sum_reg, Clk, A, B);
21
22
23
          output reg Carry_reg;
          output reg [1:0] Sum_reg;
24
25
          input wire Clk;
26
          input wire [1:0] A,B;
27
28
29
          reg [1:0] A_reg, B_reg;
30
          wire Carry;
31
          wire [1:0] Sum;
32
33
          adder_2bit a2b0(Carry, Sum, A_reg, B_reg);
34
35
36
          always@(posedge Clk)
37
38
                  begin
39
                         A_reg \ll A;
40
                         B_reg \ll B;
```

```
41
                      end
42
43
            always@(posedge Clk)
44
45
                      begin
46
                               Carry_reg <= Carry;
47
                               Sum_reg <= Sum;
48
                      end
49
50
   end module\\
51
```

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 Hold 1 Test passed
SR-latch Hold 1 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
SR-latch Enable Hold Test 4 passed
SR-latch Enable Hold Test 4 passed
SR-latch Enable Hold Test 5 passed
SR-latch Enable Hold Test 6 passed
SR-latch Enable Hold Test 7 passed
SR-latch Enable Hold Test 8 passed
SR-latch Enable Hold Test 8 passed
```

Figure 1: SR Latch Test Results

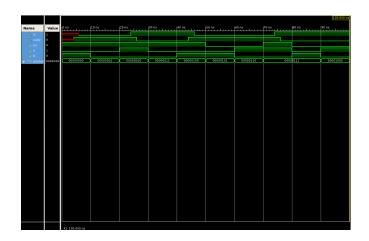


Figure 2: SR Latch Graph

```
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 2 passed

SR-latch Enable Hold Test 3 passed

SR-latch Enable Hold Test 4 passed

SR-latch Enable Hold Test 4 passed

Some tests failed

Stopped at time: 100 ns: in File "/home/ugrads/i/josephmart/prelab/sr_latch_tb.v" Line 72
```

Figure 3: SR Latch2 Test Results

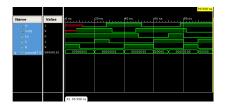


Figure 4: SR Latch2 Graph

```
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 2 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/i/josephmart/prelab/d_latch_tb.v" Line 63
```

Figure 5: D Latch Test Results

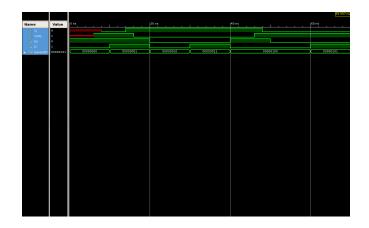


Figure 6: D Latch Graph

This is a full version of time.
Time resolution is 1, as
Simulator is droit over thirduzation process.
Friended creat intellization process.
Details insight for 127 passed
All rests passed.

Figure 7: D Latch Behavioral Test Results



Figure 8: D Latch Behavioral Graph

The, is a full version of film
Time resolution is 1 ps
Ginulator is denign contributation process.
Finding contributation process.

Of the full version of the full ve

Figure 9: D Flip Flop Behavioral Test Results

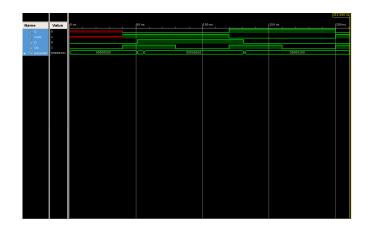


Figure 10: D Flip Flop Behavioral Graph

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.

- 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? 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?