ELEC-4200

Digital System Design

FROM: Jacob Howard

TO: Prof. Ujjwal Guin

DUE DATE: 2/18/21

Lab 4

Introduction

The goal of this lab was to familiarize ourselves with various types of latches and model them. We were also required to model flip-flops with control signals. In total, there were seven tasks for this lab.

Task 1

In Task 1, we were asked to design an SR latch by using the code that was given to us. Our main task for most of the lab was to design test-benches for each code to check if the code produced the desired output before physical testing. The testbench can be seen in *Testbench 1*, along with the simulation in *Figure 1*, and the correct LUT and IO usage in *Figure 2*. *Note that all virtual and physical tests were successful for every task and were verified by TA.*

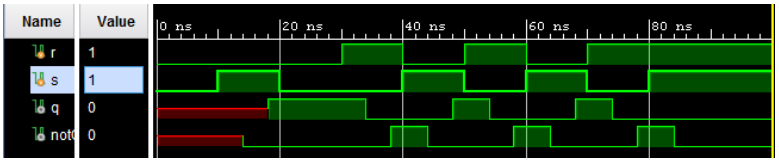

Figure 1

Resource	Utilization	Available	Utilization...
LUT	2	63400	0.01
IO	4	210	1.90

Figure 2

```
module Task1_tb(
    );
    reg s;
    reg r;
    wire q;
    wire notQ;

    Task1 DUT (.s(s), .r(r), .q(q), .notQ(notQ));

    initial begin
    #100 $finish; //runs simulation 100 times
    end

    initial begin
    s = 0;
    r = 0;
    #10
    s = 1;
    #10
    s = 0;
    #10
    r = 1;
    #10
    r = 0;
    s = 1;
    #10
    s = 0;
    r = 1;
    #10
    s = 1;
    r = 0;
    #10
    r = 1;
    s = 0;
    #10
    s = 1;
    end
endmodule
```

Testbench 1

Task 2

In Task 2, we were asked to design a gated SR latch using dataflow modeling and then develop a testbench to test and validate the design. The code for the SR latch can be seen in *Code 1,* testbench code can be seen in *Testbench 2,* Simulation can be seen in *Figure 3*, and correct LUT and IO usage in *Figure 4* below.

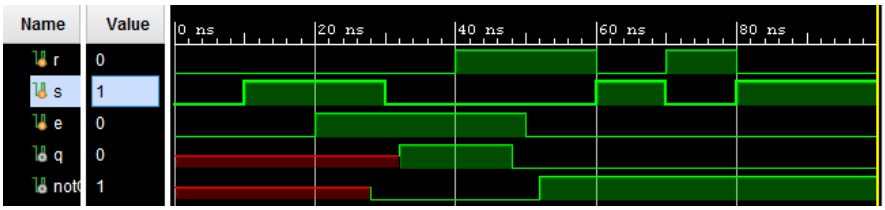

Figure 3

Resource	Utilization	Available	Utilization...
LUT	2	63400	0.01
IO	5	210	2.38

Figure 4

```
module Task2(
    input e,
    input s,
    input r,
    output q,
    output notQ
);
    wire s1;
    wire r1;

//Gated SR Latch
and #4 (s1, e, s);
and #4 (r1, e, r);

nor #4 (q, r1, notQ);
nor #4 (notQ, s1, q);

endmodule
```

Code 1

```
module Task2_tb(

    );
        reg e;
        reg s;
        reg r;
        wire q;
        wire notQ;

        Task2 DUT (.e(e), .s(s), .r(r), .q(q),
.notQ(notQ));

        initial begin
        #100 $finish; //runs simulation 100
times
        end

        initial begin
        s = 0;
        r = 0;
        e = 0;
        #10 //10
        s = 1;
        #10 //20
        e = 1;
        #10 //30
        s = 0;
        #10 //40
        r = 1;
        #10 //50
        e = 0;
        #10 //60
        s = 1;
        r = 0;
        #10 //70
        r = 1;
        s = 0;
        #10 //80
        s = 1;
        r = 0;


        end
endmodule
```

Testbench 2

Task 3

In Task 3, we were asked to design a D latch using dataflow modeling and develop a testbench to test and validate the design. The code for the D latch can be seen in *Code 2,* testbench code can be seen in *Testbench 3*, Simulation can be seen in *Figure 5*, and correct LUT and IO usage in *Figure 6* below.

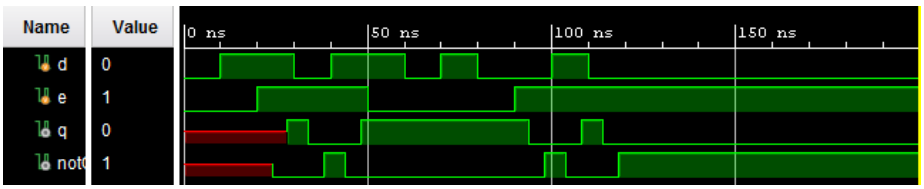

Figure 5

Resource	Utilization	Available	Utilization...
LUT	2	63400	0.01
IO	4	210	1.90

Figure 6

```
module Task3(
    input d,
    input e,
    output q,
    output notQ
    );
   wire q1;
   wire notQ1;
   wire q2;
   wire notQ2;

   assign #4 q = ~((~d & e) | notQ);
   assign #4 notQ = ~((d & e) | q);

endmodule
```

Code 2

```
module Task3_tb(

    );
    reg d;
    reg e;
    wire q;
    wire notQ;

     Task3 DUT (.d(d), .e(e), .q(q), .notQ(notQ));

          initial begin
          #200 $finish; //runs simulation 100 times
          end

          initial begin
          d = 0;
          e = 0;
          #10 //10
          d = 1;
          #10 //20
          e = 1;
          #10 //30
          d = 0;
          #10 //40
          d = 1;
          #10 //50
          e = 0;
          #10 //60
          d = 0;
          #10 //70
          d = 1;
          #10 //80
          d = 0;
          #10 //90
          e = 1;
          #10 //100
          d = 1;
          #10 //110
          d = 0;
          end
endmodule
```

Testbench 3

Task 4

In Task, we were asked to model a D flip-flop using behavioral modeling and develop a testbench to validate the model. The code for the D flip-flop can be seen in *Code 3,* testbench code can be seen in *Testbench 4* and Simulation can be seen in *Figure 7*.

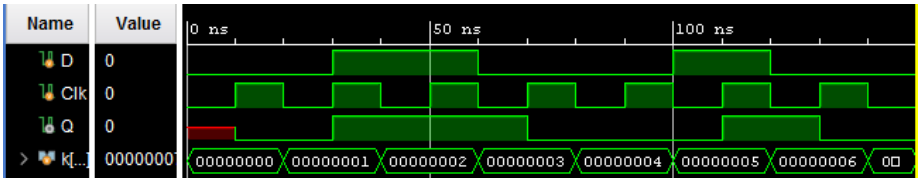

Figure 7

```
module Task4(
   input D,
   input Clk,
   output reg Q
   );
   always @ (posedge Clk)
   if(Clk)
   begin
   Q <= D;
   end
endmodule
```

Code 3

```
module Task4_tb();
   reg D;
   reg Clk;
   wire Q;
   integer k;

   Task4 DUT (.D(D), .Clk(Clk), .Q(Q));
   initial begin
   #150 $finish; //runs simulation 150 times
   end

   initial begin
   D = 0;
   #30 D = 1;
   #30 D = 0;
   #40 D = 1;
   #20 D = 0;
   end

   //clock
   initial begin
   for (k = 0; k <= 6; k = k+1)
   begin
      Clk = 0;
      #10 Clk = 1;
      #10;
      Clk = 0;
      end
   end
endmodule
```

Testbench 4

Task 5

In Task 5, we were asked to Model the circuit, as shown in Figure 8 below, using behavioral modeling. The code for the circuit can be seen in *Code 4,* testbench code can be seen in *Testbench 5* and Simulation can be seen in *Figure 9*.

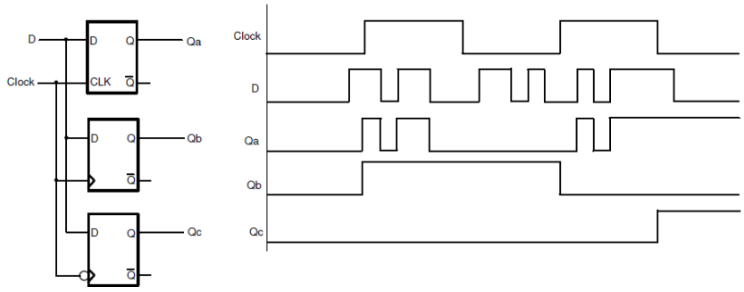

Figure 8

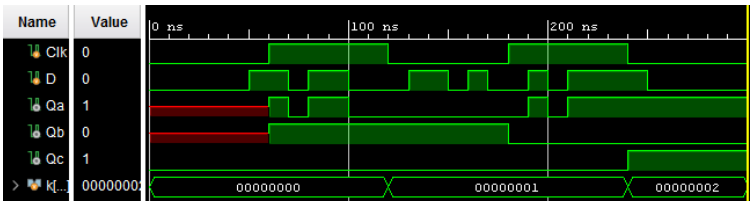

Figure 9

```
module Task5(
    input Clk,
    input D,
    output reg Qa,
    output reg Qb,
    output reg Qc
    );

    reg Qbar;

    always @ (D or Clk)
    if(Clk)
    begin
    Qa <= D;
    Qbar <= ~D;
    end

    always @ (posedge Clk)
        if(Clk)
        begin
        Qb <= D;
        end

        always @ (negedge Clk)
            begin
            Qc <= D;
            end

endmodule
```

Code 4

```
module Task5_tb(
    );
    reg Clk;
    reg D;
    wire Qa;
    wire Qb;
    wire Qc;
    integer k;

    Task5 DUT (.D(D), .Clk(Clk), .Qa(Qa), .Qb(Qb), .Qc(Qc));
     initial begin
       #300 $finish; //runs simulation 300 times
       end

       initial begin
       D = 0;
       #50
       D = 1;
       #20 //70
       D = 0;
       #10 //80
       D = 1;
       #20 //100
       D = 0;
       #30 //130
       D = 1;
       #20 //150
       D = 0;
       #10 //160
       D = 1;
       #10 //170
       D = 0;
       #20 //190
       D = 1;
       #10 //200
       D = 0;
       #10 //210
       D = 1;
       #40 //250
       D = 0;
       end

      initial begin
      for (k = 0; k <= 6; k = k+1)
      begin
         Clk = 0;
         #60 Clk = 1;
         #60;
         Clk = 0;
         end
      end
endmodule
```

Testbench 5

Task 6

In task 6, we were asked to model the D flip-flop with synchronous reset using behavioral modeling and develop a testbench to test and validate the design. The code for the D flip-flop can be seen in *Code 5,* testbench code can be seen in *Testbench 6*, Simulation can be seen in *Figure 10*, and correct BUFG and IO usage in *Figure 11* below.

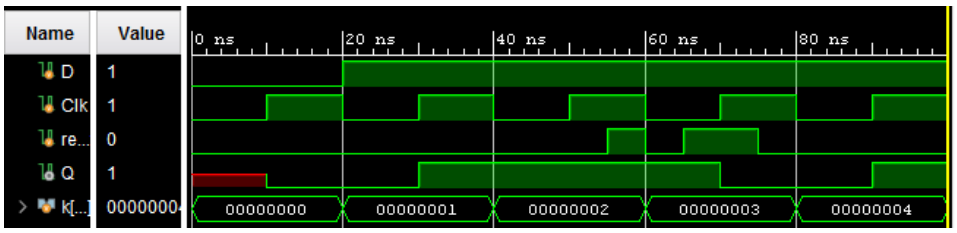

Figure 10

Resource	Utilization	Available	Utilization...
FF	1	126800	0.01
IO	4	210	1.90
BUFG	1	32	3.13

Figure 11

```
module Task6(
    input D,
    input Clk,
    input reset,
    output reg Q
    );

    always @(posedge Clk)
    if (reset
    )begin
    Q <= 1'b0;
    end else
     begin
     Q <= D;
     end
endmodule
```

Code 5

```
module Task6_tb();

reg D;
reg Clk;
reg reset;
wire Q;
integer k;

Task7 DUT (.Clk(Clk), .reset(reset), .D(D), .Q(Q));
    initial begin
      #100 $finish; //runs simulation 100 times
      end

      initial begin
      D = 0;
      reset = 0;
      #20 //20
      D = 1;
      #35 //55
      reset = 1;
      #5 //40
      reset = 0;
      #5 //45
      reset = 1;
      #10 //55
      reset = 0;
      end

   //Clock
    initial begin
       for (k = 0; k <= 5; k = k+1)
       begin
          Clk = 0;
          #10 Clk = 1;
          #10;
          Clk = 0;
          end
      end
endmodule
```

Testbench 6

Task 7

In task 7, we were asked to model the D flip-flop with synchronous reset and clock enable using behavioral modeling and develop a testbench to test and validate the design. The code for the D flip-flop can be seen in *Code 6,* testbench code can be seen in *Testbench 7*, Simulation can be seen in *Figure 12*, and correct BUFG and IO usage in *Figure 13* below.

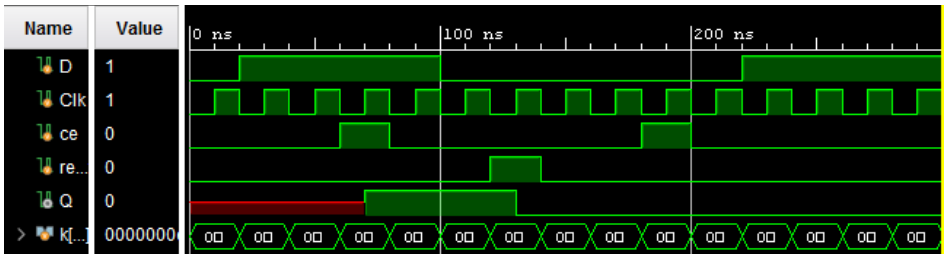

Figure 12

Resource	Utilization	Available	Utilization...
FF	1	126800	0.01
IO	5	210	2.38
BUFG	1	32	3.13

Figure 13

```
module Task6(
    input D,
    input Clk,
    input reset,
    output reg Q
    );

    always @(posedge Clk)
    if (reset
    )begin
    Q <= 1'b0;
    end else
     begin
     Q <= D;
     end
endmodule
```

Code 6

```
module Task7_tb();
reg D;
reg Clk;
reg ce;
reg reset;
wire Q;
integer k;

Task7 DUT (.Clk(Clk), .reset(reset), .D(D), .ce(ce),
.Q(Q));
    initial begin
      #300 $finish; //runs simulation 300 times
      end

      initial begin
      D = 0;
      ce = 0;
      reset = 0;
      #20 //20
      D = 1;
      #40 //60
      ce = 1;
      #20 //80
      ce = 0;
      #20 //100
      D = 0;
      #20 //120
      reset = 1;
      #20 //140
      reset = 0;
      #40 //180
      ce = 1;
      #20 //200
      ce = 0;
      #20 //220
      D = 1;
      end

      //Clock
          initial begin
             for (k = 0; k <= 15; k = k+1)
             begin
                Clk = 0;
                #10 Clk = 1;
                #10;
                Clk = 0;
                end
            end
endmodule
```

Testbench 7

Task 8

In task 8, we were asked to model a T flip-flop with synchronous negative-logic reset and clock enable using the above given to us in *Code 7.* All we had to do for this task was physically verify the design on the board. The design ran as expected.

```
module Task8(
    input Clk,
    input reset_n,
    input T,
    output reg Q
    );

    always @(negedge Clk)
    if (!reset_n)
    Q <= 1'b0;
    else if (T)
    Q <= ~Q;
endmodule
```

Code 7

Conclusion

In conclusion, this lab was very helpful for us to understand different types of latches and flip-flops. This lab was also an introduction to coding workbenches. While I wish the lab went into more detail on how to correctly write workbenches and provided examples, it was overall a simple process. The lab manual was well written and gave plenty of coding examples for latches and flip-flops and latches. Overall, I would say the lab was fairly easy but did take a very long time to write all the code for since there were so many tasks. Other than the length, I would say this lab was a good intro to writing our own testbenches, latches, and flip-flops.