Engineering 378 Digital Systems Design

Exp. 3: Latches, Flip-flops and Sequential Circuits

Submitted by:

NoraldAlejo
Farnam Adelkhani

September 30, 2016

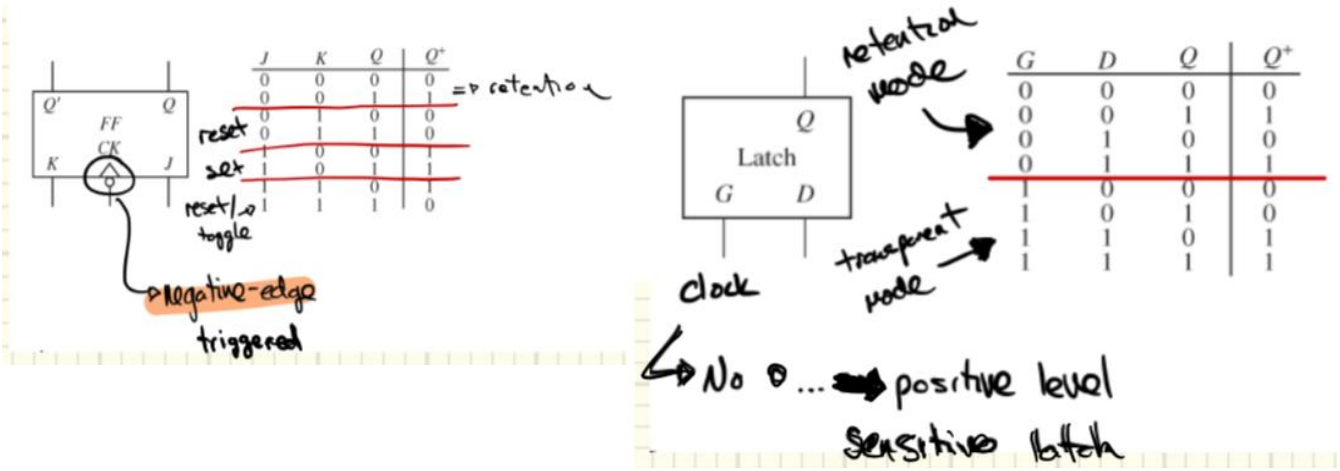
Demonstration	_/5
Report: Presentation	_/2
Problem: analysis	_/2
Design: work	_/5
Results	_/5
Conclusion/Discussion	_/1
Total	_/20

Problem Analysis

The primary objective in this laboratory assignment is again familiarity with Verilog design code of both basic sequential circuit components, such as latches and flip-flops, as well as complete sequential circuits. Another goal of this section is to design and implement a Test Bench for the sequential circuit design.

Hardware Design

The hardware design portion of this laboratory assignment begins in the pre-lab with a quick review on writing do you latch and JK flip-flops that are pictured below:

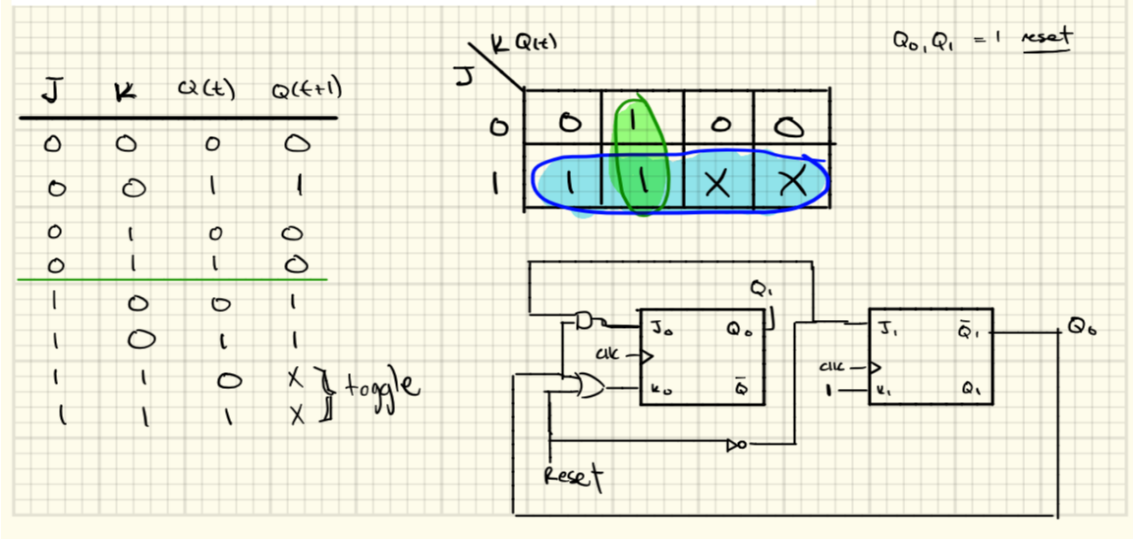


J	K	Q	Q+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

G	D	Q	Q+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

The next section in the design process requires the user to implement a 2-bit counter with an active high reset signal using a maximum of two JK flip-flops and no other registers. There are no restrictions put on the approach to the design process, the user is free to implement any design technique including K-maps and diagrams. The design needs to be implemented keeping in mind that the next step is to interpret the work to Verilog code. So the counter being designed should be able to run from zero 23 and then back down to zero, with an active high reset.

- ***Final Design***



The final design was implemented using 2 JK flip flops one or gate and one and gate. Q not is the output with an active high reset.

Verilog Modeling

Part 1

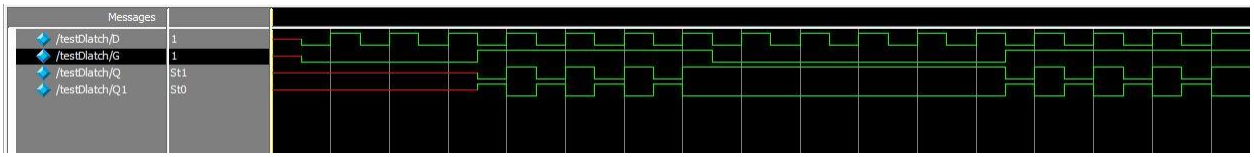
The first part of the procedure is the design and simulation of the D latch. The user is required to write a Verilog module for the D latch as well as a Test Bench and then to carry out and implement the simulation. The code for the Verilog module is given below:

```
Ln#
 1  module Dlatch (G, D, Q, Q1);
 2  input G, D;
 3  output Q1, Q;
 4
 5  reg Q1, Q;
 6
 7  always @(D or G)
 8    if (G)
 9      begin
10        Q <= D;
11        Q1 <= ~D;
12      end
13  endmodule
14
```

The test bench for the D latch design is given below:

```
module testDlatch();

  wire Q, Q1;
  reg G, D;
  Dlatch d1(G, D, Q, Q1);

  initial
  begin
    #10 G = 0; D = 0;
    #10 G = 0; D = 1;
    #10 G = 0; D = 0;
    #10 G = 0; D = 1;
    #10 G = 0; D = 0;
    #10 G = 0; D = 1;

    #10 G = 1; D = 0;
    #10 G = 1; D = 1;
    #10 G = 1; D = 0;
    #10 G = 1; D = 1;
    #10 G = 1; D = 0;
    #10 G = 1; D = 1;
    #10 G = 1; D = 0;
    #10 G = 1; D = 1;

    #10 G = 0; D = 0;
    #10 G = 0; D = 1;
    #10 G = 0; D = 0;
    #10 G = 0; D = 1;
    #10 G = 0; D = 0;
    #10 G = 0; D = 1;
    #10 G = 0; D = 0;
    #10 G = 0; D = 1;
    #10 G = 0; D = 0;
    #10 G = 0; D = 1;

    #10 G = 1; D = 0;
    #10 G = 1; D = 1;
    #10 G = 1; D = 0;
    #10 G = 1; D = 1;
    #10 G = 1; D = 0;
    #10 G = 1; D = 1;
    #10 G = 1; D = 0;
    #10 G = 1; D = 1;
end
endmodule
```

Lastly in order to verify that the module and test bench are working as intended the waveforms to the truth table are compared to the one generated to find there are no errors:



Part 2

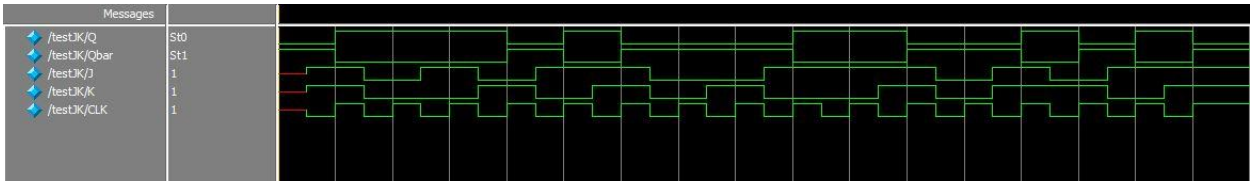
The next portion of this laboratory assignment Will show the Verilog module for the JK flip-flop design. This design is shown below:

```
1   module jkFlipFlop (J, K, CLK, Q, Qbar);
2     input J, K, CLK;
3     output Q, Qbar;
4
5     reg Q, Qbar;
6
7     initial
8       begin
9         Q = 0;
10        Qbar = 1;
11    end
12
13    always @ (posedge CLK)
14    begin
15      case ({J, K})
16        {1'b0, 1'b0}:
17        begin
18          Q = Q;
19          Qbar = Qbar;
20        end
21
22        {1'b0, 1'b1}:
23        begin
24          Q = 0;
25          Qbar = 1;
26        end
27
28        {1'b1, 1'b0}:
29        begin
30          Q = 1;
31          Qbar = 0;
32        end
33
34        {1'b1, 1'b1}:
35        begin
36          Q = ~Q;
37          Qbar = ~Qbar;
38        end
39      endcase
40    end
41  endmodule
```

The test bench for the JK flip-flop design and simulation is shown here:

```
module testJK();

  wire Q, Qbar;
  reg J, K, CLK;

  jkFlipFlop jk(J, K, CLK, Q, Qbar);

  initial
  begin
    #10 CLK = 0; J = 1; K = 1;
    #10 CLK = 1; J = 1; K = 1;
    #10 CLK = 0; J = 0; K = 0;
    #10 CLK = 1; J = 0; K = 0;
    #10 CLK = 0; J = 1; K = 0;
    #10 CLK = 1; J = 1; K = 0;
    #10 CLK = 0; J = 0; K = 1;
    #10 CLK = 1; J = 0; K = 1;
    #10 CLK = 0; J = 1; K = 0;
    #10 CLK = 1; J = 1; K = 0;
    #10 CLK = 0; J = 1; K = 1;
    #10 CLK = 1; J = 1; K = 1;

    #10 CLK = 0; J = 0; K = 0;
    #10 CLK = 1; J = 0; K = 0;
    #10 CLK = 0; J = 0; K = 1;
    #10 CLK = 1; J = 0; K = 1;
    #10 CLK = 0; J = 1; K = 0;
    #10 CLK = 1; J = 1; K = 0;
    #10 CLK = 0; J = 1; K = 0;
    #10 CLK = 1; J = 1; K = 0;
    #10 CLK = 0; J = 1; K = 1;
    #10 CLK = 1; J = 1; K = 1;

    #10 CLK = 0; J = 0; K = 0;
    #10 CLK = 1; J = 0; K = 0;
    #10 CLK = 0; J = 1; K = 1;
    #10 CLK = 1; J = 1; K = 1;
    #10 CLK = 0; J = 0; K = 1;
    #10 CLK = 1; J = 0; K = 1;
    #10 CLK = 0; J = 1; K = 0;
    #10 CLK = 1; J = 1; K = 0;
    #10 CLK = 0; J = 1; K = 1;
    #10 CLK = 1; J = 1; K = 1;
  end
endmodule
```

Lastly in order to verify that the design and simulation are working properly we take a look at the waveforms for comparison to the truth table:



Part 3

The Verilog design for the two big counter using JK flip-flops with an active high reset signal is shown below:

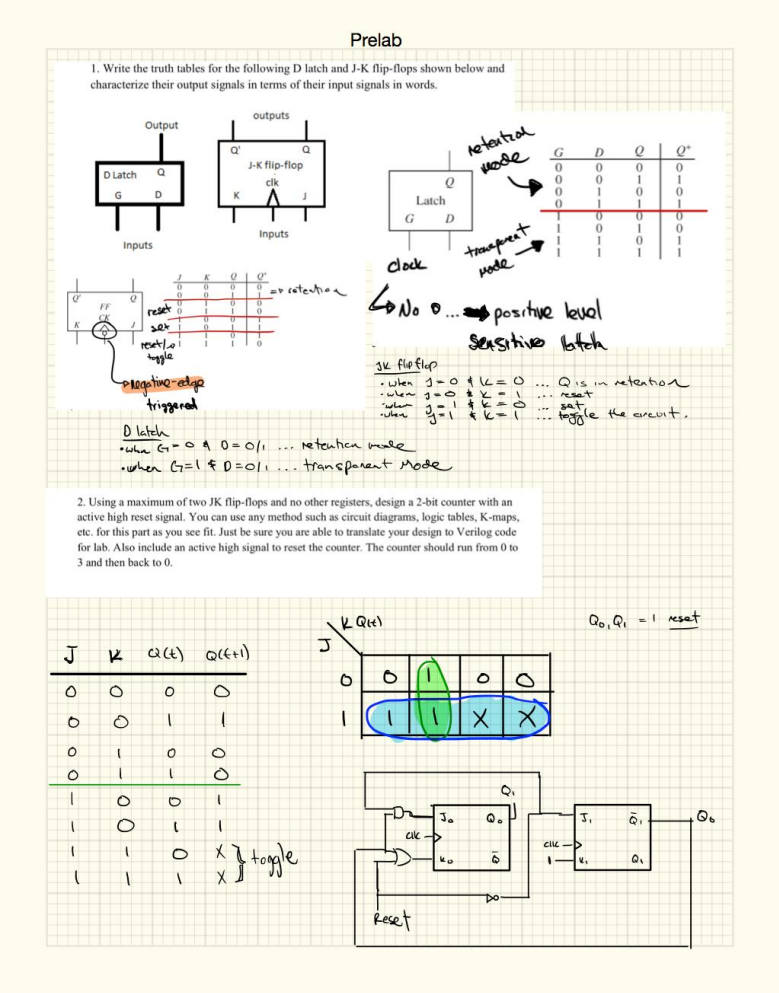
Results/Verification

Conclusion and Discussion

Work Breakdown

- Pre-lab assignments were completed individually.
- We worked together to construct a 4-bit circuit for test-bench implementation but we would have likely taken different approaches. In the end, Norald had some fantastic ideas on how to construct the system with a sequence of ‘if’ statements. Quite brilliant.
- Writing a test bench was a simple procedure that we worked together on.
- We split up the work and used a google doc to complete the lab report write-up.

Prelabs



HDL Source Code