ECE M16 and CS M51A

Winter 2019 Section 2 Logic Design of Digital Systems

Dr. Yutao He

Verilog Lab #3 - Design of Sequential Systems

Due: March 10, 2019

Team ID: P33

Team Member: Zhiheng Ma (404950730)

Date: Feb 20, 2019

Result	
Correctness	
Creativity	
Report	
Total	

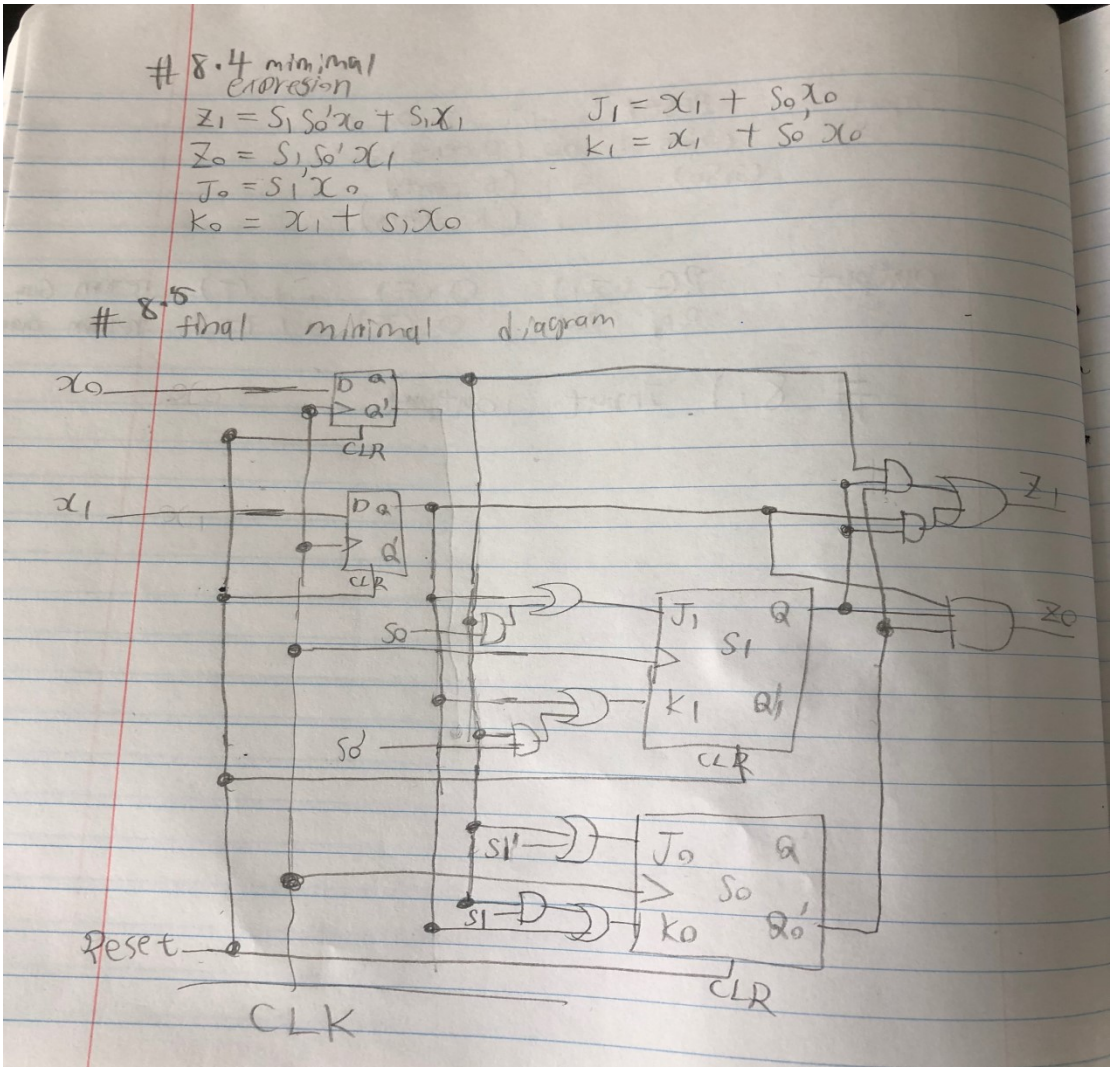
Abstract:

This project requires build a circuit for a vending machine by using JK flip flops. This vending machine has three states (0cents, 5cents, 10cents, 15cents). After the vending machine has 20 cents, it will return a gum (Z1). If the vending machine has 25 cents, it will return 5cents (Z0). The user will input 0 cents, 5 cents or 10 cents. Moreover, the vending machine can reset the status.

Switching Function of Circuit:

#8.4 minimal OR-AND expression

#8.5 schematic of the circuit composed of logic gates and flip-flops



Verlog Code:

```
////////////////////////////////////
// school: ucla
 // class: ee M16
// name:Zhiheng Ma 404950730
//
```

```
21  module dff(
22   input wire d_in,
23   input wire Clk,
24   input wire reset,
25   output reg q_out);
26
27   always@(posedge Clk) begin
28   if(reset)
29   q_out <= 0;
30  else
31   q_out <= d_in;
32   end
33  endmodule
34
```

```
35  module eem16_proj3(
36        input wire [1:0]x,
37        input wire reset,
38        input wire clk,
39        output reg [1:0]z
40      );
41      reg j1,k1,j0,k0;
42      wire x1,x0,s1,s0;
43
44      jkff jkff1(
45  .J (j1),
46  .K (k1),
47  .CLK (clk),
48  .CLR (reset),
49  .Q (s1));
50
51  jkff jkff0(
52  .J (j0),
```

```
51  jkff jkff0(
52  .J (j0),
53  .K (k0),
54  .CLK (clk),
55  .CLR (reset),
56  .Q (s0));
57
58  dff in_1(
59  .d_in (x[1]),
60  .Clk (clk),
61  .reset (reset),
62  .q_out (x1));
63
64  dff in_0(
65  .d_in (x[0]),
66  .Clk (clk),
67  .reset (reset),
68  .q_out (x0));
69
```

```
70
71  always @(*) begin
72   j1 <=x1|(s0&x0);
73   k1 <=x1|(~s0&x0);
74   j0 <=~s1&x0;
75   k0 <=x1|(s1&x0);
76   z[1] <=(s1&~s0&x0)|(s1&x1);
77   z[0] <=s1&~s0&x1;
78  end
79
80  endmodule
```

#1 .V code

```
////////////////////////////////////
// school: ucla
 // class: ee M16
// name:Zhiheng Ma 404950730
//
```

```
module ee_m16_proj3_tb;

   // Inputs
   reg [1:0] x;
   reg reset;
   reg clk;

   // Outputs
   wire [1:0] z;

   // Instantiate the Unit Under Test (UUT)
   eem16_proj3 uut (
      .x(x),
      .reset(reset),
      .clk(clk),
      .z(z)
```

```
44     initial begin
45        // Initialize Inputs
46        x=0;
47        reset=1;
48        clk=1;
49
50      #10 reset = 0;
51
52      #10
53  //test1: insert five nickels
54      x=1;
55      #10
56      x=1;
57      #10
58      x=1;
59      #10
60      x=1;
61      #10
62      x=1;
```

```
   #10//set to orginal state
   x=0;

   #10
   //test2: insert one nickels, two dime
   x=3;
   #10
   x=3;
   #10
   x=1;

   #10
   x=0;//set to orginal state

   ---
```

```
78      #10
79      //test3: insert one nickel, and rese
80      x=1;
81      #10
82      reset = 1;
83
84      #10
85      reset = 0;
86      #10
87      x=0; //set to orginal state
88
```

```
90      //test4: insert two nickel, and reset
91       x=1;
92      #10
93      x=1;
94      #10
95      reset = 1;
96      #10
97      reset = 0;
98      #10
99      x=0; //set to orginal state
100
101     #10
102     //test5: insert one nickle, and two dime
103     x=1;
104     #10
105     x=3;
106     #10
107     x=3;
108     #10
109
110     $finish;
```

```
111
112
113 end
114       always begin
115     #5 clk = ~clk;
116 end
117
118
119 endmodule
120
```

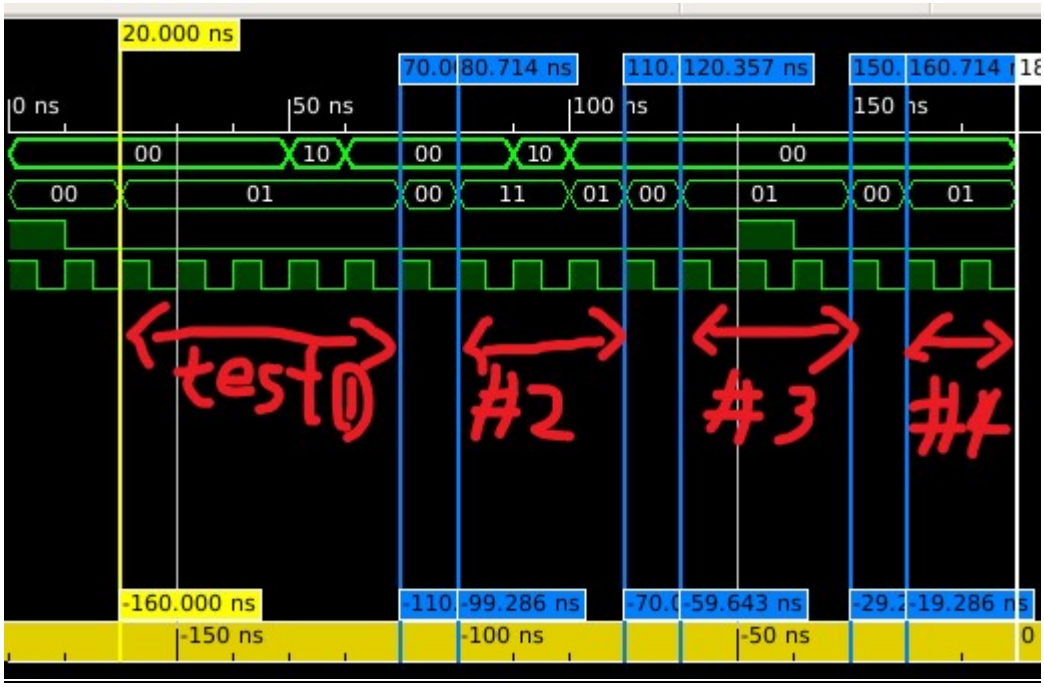
#2 test bench code

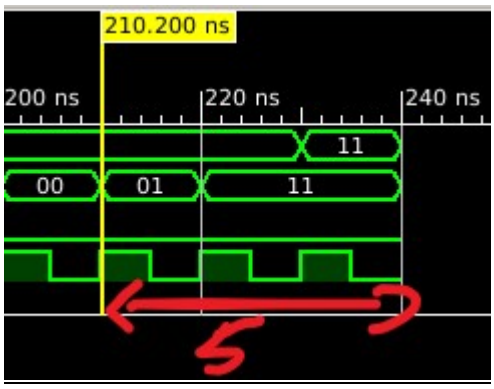
```
module jkff(
    input wire J,
    input wire K,
    input wire CLK,
    input wire CLR,
    output reg Q);

always @(posedge CLK) begin
  if (!CLR)
    if(J==1'b0 && K==1'b1) begin
      Q <= 'b0;
    end
    else if(J==1'b1 && K==1'b0) begin
      Q <= 1'b1;
    end
    else if(J==1'b1 & K==1'b1) begin
      Q <= ~Q;
    end
  else
     Q<=0;
  end
endmodule
////////////////////////////////////
// school: ucla
 // class: ee M16
// name:Zhiheng Ma 404950730
//
```

#3 JK Flip-flop

Stimulation Result:





#1 input five nickels, the output should only be return gum.

#2 input two dimes and one nickels, the output should be return gum.

#3 input one nickels, and use the reset, the output should be 00.

#4 input a dimes, and use the reset, the output should be 00

#5 input a nickel, and two dimes, return gum and return nickel

Design Review:

The most important thing I learn is the skill is using k-map to converting status table to flip-flop. This skill might help me to design the circuit more efficiently. The most difficult problem I found is converting the status table and using K-map to find the minimal expression for the JK flip-flop. I spend most of time on it. After I wrote down the truth table, I finished the k-map and minimal expression and computed the code very quick.

Team Member Contribution:

Zhiheng Ma(100%): I guess I need to do all the work ☺

Appendix:

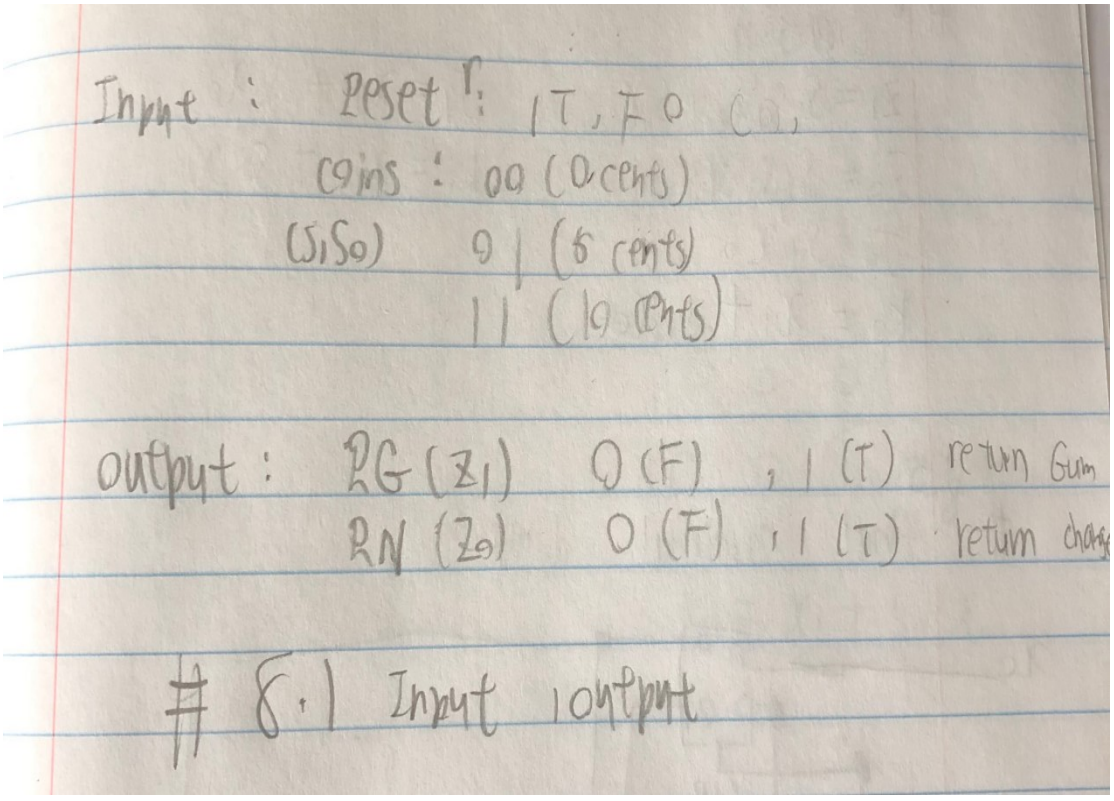
8.1 inputs, outputs, and states of the system.

8.2 encoding schemes of inputs, output, and states.

8.3 the state diagram and the state table.

8.4 minimization procedure for state and output variables by means of the K-map.

8.5 final minimal expressions of the logic functions in forms of ORAND switching expressions and the final schematic of the circuit.



Rest

PS	Input				
State	E0	N5	D10	0	1
S0	(S0,00)	(S1,00)	(S2,00)	(S0,00)	(S0,01)
S1	(S1,00)	(S2,00)	(S3,00)	(S1,00)	(S0,01)
S2	(S2,00)	(S3,00)	(S0,10)	(S2,00)	(S0,01)
S3	(S3,00)	(S0,10)	(S0,11)	(S3,00)	(S0,01)

NS, output (Z, Z0)

2G RN

#8.2 ↑ scheme of input

PS S1 S0	Input x1 x0		
	00	01	11
00	00,00	01,00	11,00
01	01,00	11,00	10,10
11	11,00	10,00	00,10
10	10,00	00,10	00,11

← #8.3 State diagram

NS(S1,S0), output (Z, Z0)

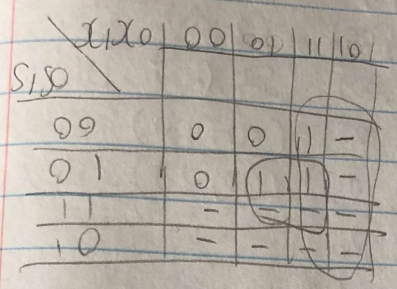
PS S1 S0	Input 00	01	11
00	0-,0-	0-,1-	1-,1-
01	0-,-0	1-,-0	1-,-1
11	-0,-0	-0,-1	-1,-1
10	-0,0-	-1,0-	-1,0-

← #8.3 state table

(J1K1, J0K0)

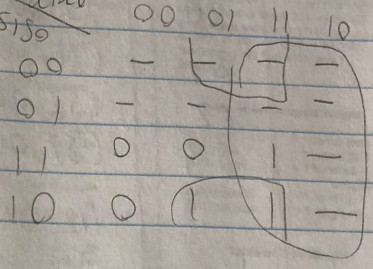
#8.4 kmap

J1:



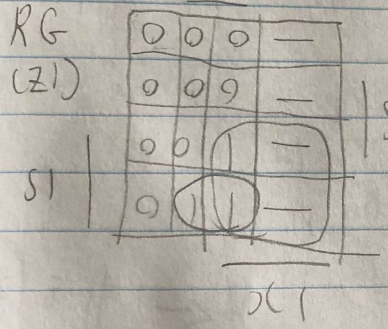
$J_1 = x_1 + S_0 x_0$

K1



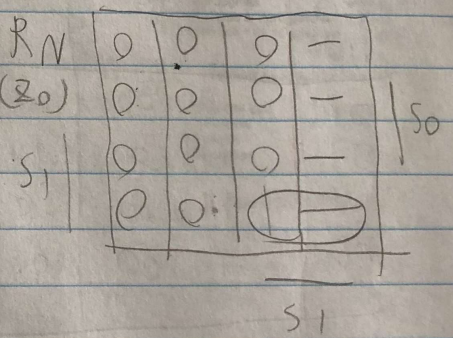
$K_1 = x_1 + S_0' x_0$

RG (z1)



$RG = S_1 S_0' x_0 + S_1 x_1$

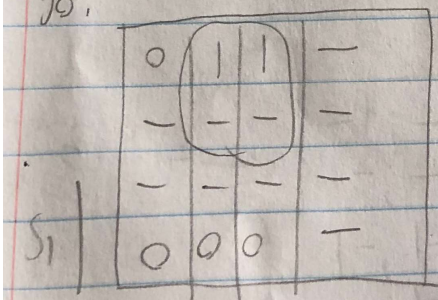
RN (z0)



$RN = S_1 S_0' x_1$

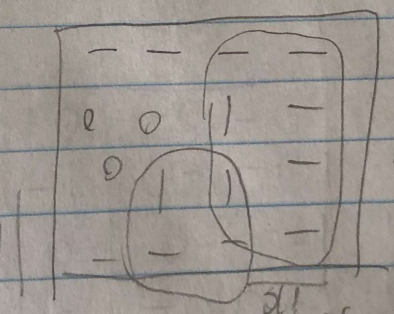
(z1)

J0:



$J_0 = S_1' x_0$

K0



$K_0 = S_1 x_0 + x_1$

8.4 minimal expression

$Z_1 = S_1 S_0' x_0 + S_1 x_1$

$Z_0 = S_1 S_0' x_1$

$J_0 = S_1 x_0$

$K_0 = x_1 + S_1 x_0$

$J_1 = x_1 + S_0 x_0$

$K_1 = x_1 + S_0 x_0$

8.5 final minimal diagram