CS M51A and EE M16 Summer 2016 Section 1

Logic Design of Digital Systems

Dr. Yutao He

Verilog Lab #3 - Design of Sequential Systems

Due: August 9, 2016

Team ID: _____

(1) Name: Wu Michael
Last First

Student ID: 404751542

Signature: Michael Wu

(2) Name: Zhou Minghong
Last First

Student ID: 004424670

Signature: MingHong

Date: 08/07/16

Result	
Correctness	
Creativity	
Report	
Total Score	

Abstract

In this project we designed a vending machine controller, which is a finite state machine. It produces a set of signals as it traverses through its states. These signals, along with the inputs into the controller, are used to determine its outputs and its next state. The vending machine performs the following functions. It delivers a package of gum after receiving 20 cents or more in coins. The machine has a single coin slot that accepts only nickels and dimes. It should be able to detect the value of the coin that was inserted. Also, the vending machine performs two actions when a reset button is pressed. It first sets the controller to the initial state, and then triggers another mechanism to return all the coins inserted. When the sum of the inserted coins is greater than or equal to 20 cents, the vending machine dispenses a pack of gum and returns 5 cents of change if necessary.

The Functions of the Circuit

Switching expressions

Z1 = S1S'0X0 + S1X1

Z0 = S1S0'X1

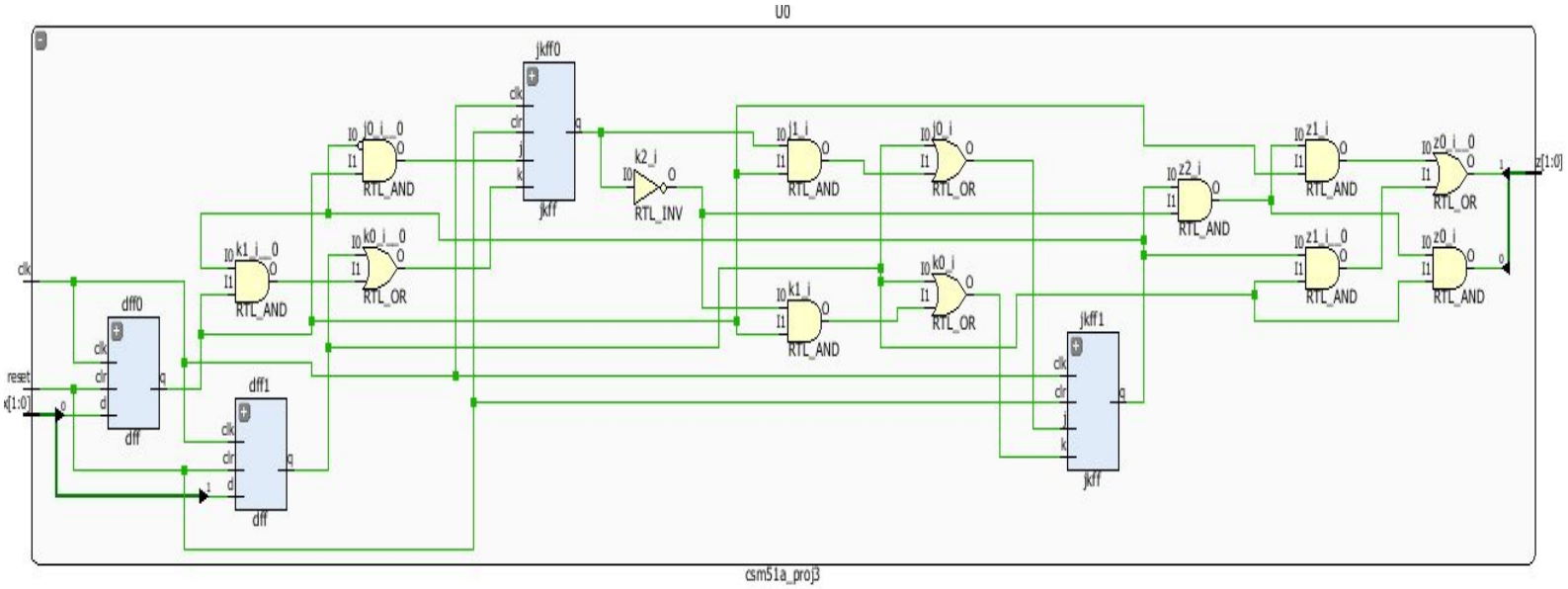
J1 = X1 + S0X0

K1 = X1 + S'X0

J0 = S0'X0

K0 = X1 + S2X0

Schematic of the Circuit:

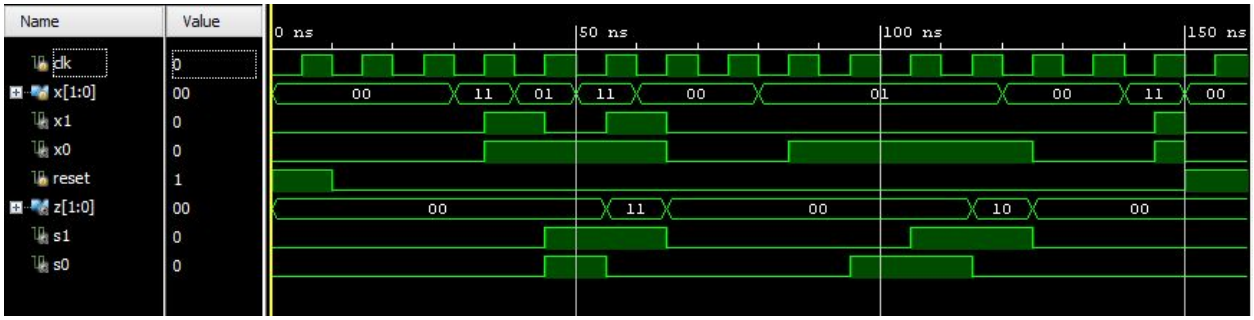


The Verilog Code

```
//csm51a_proj3.v
//Michael Wu ID: 404751542
//Minghong Zhou ID: 004424670
module dff(
   input wire d,
   input wire clk,
   input wire clr,
   output reg q);
always @(posedge clk) begin
     if (!clr)
     q<=d;
end
always @(posedge clr) begin
   q<=0;
end
endmodule

module csm51a_proj3(
     input wire [1:0]x,
     input wire reset,
     input wire clk,
     output reg [1:0] z
);
reg j1;
reg k1;
reg j0;
reg k0;
wire x1;//synchronized inputs
wire x0;
wire s1;
wire s0;
jkff jkff1(
     .j (j1),
     .k (k1),
     .clk (clk),
     .clr (reset),
     .q (s1));
jkff jkff0(
     .j (j0),
     .k (k0),
     .clk (clk),
     .clr (reset),
```

```
      .q (s0));
dff dff1(
      .d (x[1]),
      .clk (clk),
      .clr (reset),
      .q (x1));
dff dff0(
      .d (x[0]),
      .clk (clk),
      .clr (reset),
      .q (x0));

always @(*) begin
      j1<=x1|(s0&x0);
      k1<=x1|(~s0&x0);
      j0<=~s1&x0;
      k0<=x1|(s1&x0);
      z[1]<=(s1&~s0&x0)|(s1&x1);
      z[0]<=s1&~s0&x1;
end
endmodule

//jkff.v
//Michael Wu ID: 404751542
//Minghong Zhou ID: 004424670
module jkff(
      input wire j,
      input wire k,
      input wire clk,
      input wire clr,
      output reg q);
always @(posedge clk) begin
      if (!clr)
      q<=(j&~q)|(~k&q);
end
always @(posedge clr) begin
      q<=0;
end
endmodule
```

The Simulation Result



In this simulation the top row is the clock. Our system is synchronized with the positive clock edge. The next row is the input sent by the test bench. Below that is the synchronized input signal, x1 and x0, which only changes corresponding to the clock's positive edge. Below that is the asynchronous reset signal. Next, the output z is shown. The first digit of the output corresponds to the Return Gum signal, and the second digit corresponds to the Return Nickel signal. Finally below that is the state signals, s1 and s0. The synchronized input is used to determine the output z and the next state.

In our test bench, we first tested the sequence Dime, Nickel, Dime, by sending the signals 11, 01, then 11. The states changed and the system correctly output 11 at the end of the sequence, which means the system returned gum and a nickel. Next, we tested the sequence Nickel, Nickel, Nickel, Nickel. We did this by sending the 01 signal for four clock ticks, and our system correctly output 10, which means the system returned gum and did not return a nickel. Finally, we sent Dime, reset. This resulted in the output staying at 0, and the synchronized input and the states became 0. It did not change states, so the reset signal worked correctly. Thus our system is behaving as intended.

The Design Review

In this project, we learned a lot. Designing a vending machine controller reinforced our knowledge of JK flip-flops and gave us a better understanding of their role in sequential systems. Furthermore, this project presented us with a problem that required knowledge from different chapters to solve. While implementing the vending machine in verilog, we went through a series of steps. We first found the inputs, outputs and states of the system. Then we drew the state diagram and the state table for the system. Next we minimized the switching expressions for the state and output variables using k-maps. After that we converted the switching expressions to a schematic. Finally, we created verilog code that matched our schematic and tested it. We ran into problems with the outputs at first, but we were able to solve it by synchronizing our inputs to the vending machine controller. This process let us practice what we learned in class and design something useful with our knowledge. Most of our time working on the project was spent debugging the verilog code, so that was the most important part for us.

Team Member Contributions

Both team members did roughly 50% of the work. Below is a detailed list of the work we have done for project. In the parentheses next to each line are the names of the team members responsible for that part of the project.

Inputs, Outputs, and States of the System (Minghong, Michael)
Encoding Schemes of Inputs, Output, and States (Minghong, Michael)
State Diagram and State Table (Minghong, Michael)
Minimization Procedure for State and Output Variables (Minghong, Michael)
Final Minimal Expressions of the Switching Expressions (Minghong, Michael)
Final Schematic of the Circuit (Minghong, Michael)
The Design Review (Minghong)
Abstract (Minghong)
Verilog (Michael)
Simulation Result (Michael)

inputs, outputs, and states and encoding scheme

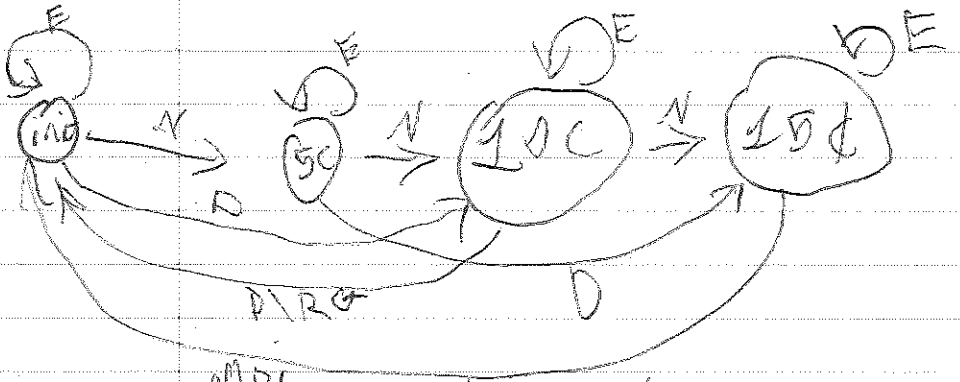
inputs		outputs	
coin value	reset	RG	RN
Empty 00	false 0	false 0	false 0
Nickel 01	true 1	true 1	true 1
Dime 11			

States

init	00
5¢	01
10¢	11
15¢	10

$S_2 S_0$

State diagram and table



NRG or DRG, RN

PS	input E	N	D
init	init	5¢	10¢
5¢	5¢	10¢	15¢
10¢	10¢	15¢	init
15¢	15¢	init	init

NS

PS	input E	N	D
init	0	0	0
5¢	0	0	0
10¢	0	0	1
15¢	0	1	1

RG

PS	input E	N	D
init	0	0	0
5¢	0	0	0
10¢	0	0	0
15¢	0	0	1

RN

State table in binary

PS	input 00	01	11	10
00	00	01	11	
01	01	11	10	
11	11	10	00	
10	10	00	00	

NS

PS	input 00	01	11	10
00	0	0	0	
01	0	0	0	
11	0	0	1	
10	0	0	1	

RG

PS	input 00	01	11	10
00	0	0	0	
01	0	0	0	
11	1	0	0	
10	1	0	0	

RN

Minimization for flip flop inputs

PS	input	J_1	K_1	J_0	K_0
00	00	0	-	0	-
00	01	0	-	1	-
00	10	-	-	-	-
00	11	1	-	1	-
01	00	0	-	-	0
01	01	1	-	-	0
01	10	-	-	-	-
01	11	1	-	-	1
10	00	-	0	0	-
10	01	-	1	0	-
10	10	-	-	-	-
10	11	-	1	0	-
11	00	-	0	-	0
11	01	-	0	-	1
11	10	-	-	-	-
11	10	-	1	-	1

J_1

S_1S_0 \ X_1X_0	00	01	11	10
00	0	0	1	-
01	0	1	1	-
11	-	-	-	-
10	-	-	-	-

$J_1 = X_1 + S_0X_0$

K_1

S_1S_0 \ X_1X_0	00	01	11	10
00	-	-	-	-
01	-	-	-	-
11	0	0	1	-
10	0	0	1	-

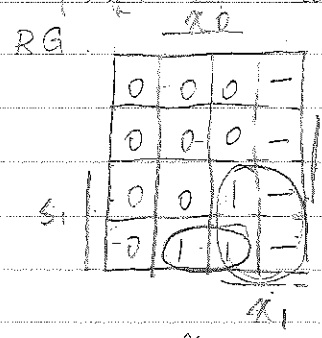
$K_1 = X_1 + S_0'X_0$

Q	J	K	00	01	11	10
0	0	0	0	1	1	
1	1	1	1	0	1	

$Q = QJ + K'Q$

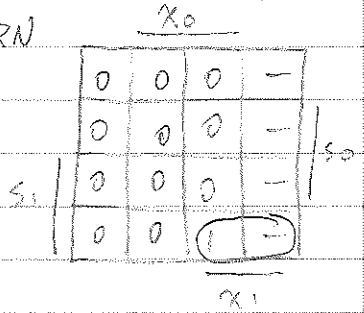
kmap's for the outputs:

RG



$RG = s_1 s_0' x_0 + s_1 x_1$

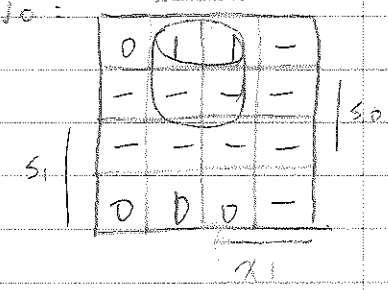
RN



$RN = s_1 s_0' x_1$

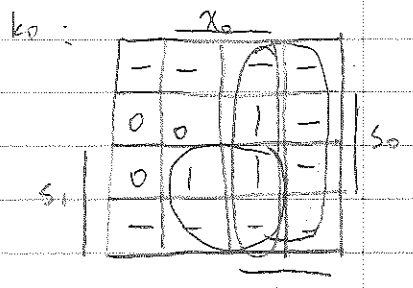
kmap's for the flip-flop inputs:

J_0:



$J_0 = s_1' x_0$

k_0:



$k_0 = s_1 x_0 + x_1$

final Minimal expressions

$Z_1 = S_1 S_0' X_0 + S_1 X_1$

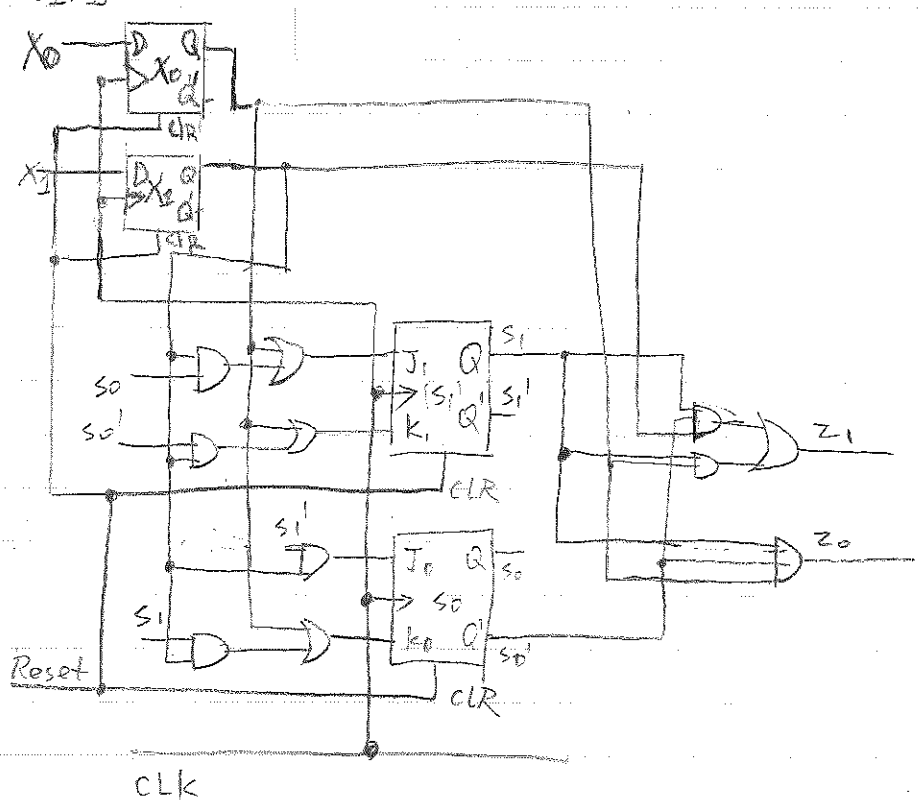
$Z_0 = S_1 S_0' X_1$

$J_1 = X_1 + S_0 X_0$

$K_1 = X_1 + S_0' X_0$

$J_0 = S_1' X_0$

$K_0 = X_1 + S_1 X_0$



We use d flip flops to synchronize the inputs