# CS M51A and EE M16 Spring 2015 Section 1

**Logic Design of Digital Systems**

**Dr. Yutao He**

# Verilog Lab #3 - Design of Sequential Systems

**Due: June 2nd, 2015**


**(1) Name: Wu, Ying Bin**
**Student ID: 104485521**

**Signature: Ying Bin Wu**

**(2) Name: Yuan, Qing**
**Student ID: 104009336**

**Signature: Qing Yuan**


**Date: 06/03/15**

**Verilog Lab #3 Project Requirement**

# Dr. Yutao He

**Due: 06/03/2015**

## *Abstract*

The purpose of this project is to design a sequential system representing a vending machine controller which takes two inputs and has two outputs. The inputs of the sequential system are coin {Empty, Nickel, Dime} and Reset {True, False}. The outputs are Release Gum {True, False} and Release Nickel {True, False}. The controller takes in coins until a threshold(the price of the gum) is hit. If the threshold is hit, the controller releases a piece of gum. If the threshold is exceeded(the amount of money inputted exceeds the price of the gum), the controller will release a nickel. The reset button is there so that in any point in time, we can go back to the initial state(amount of money inputted is 0). A diagram of the sequential system is shown below.

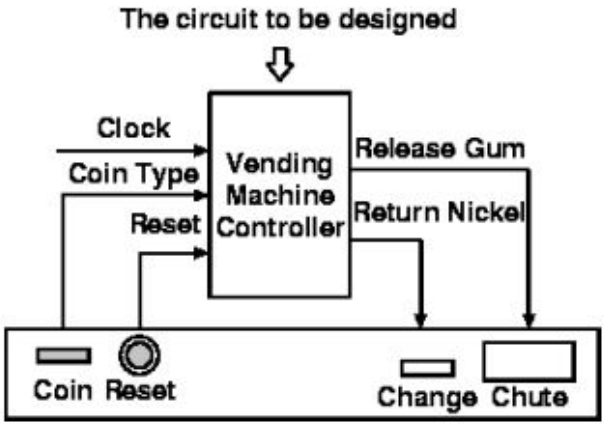
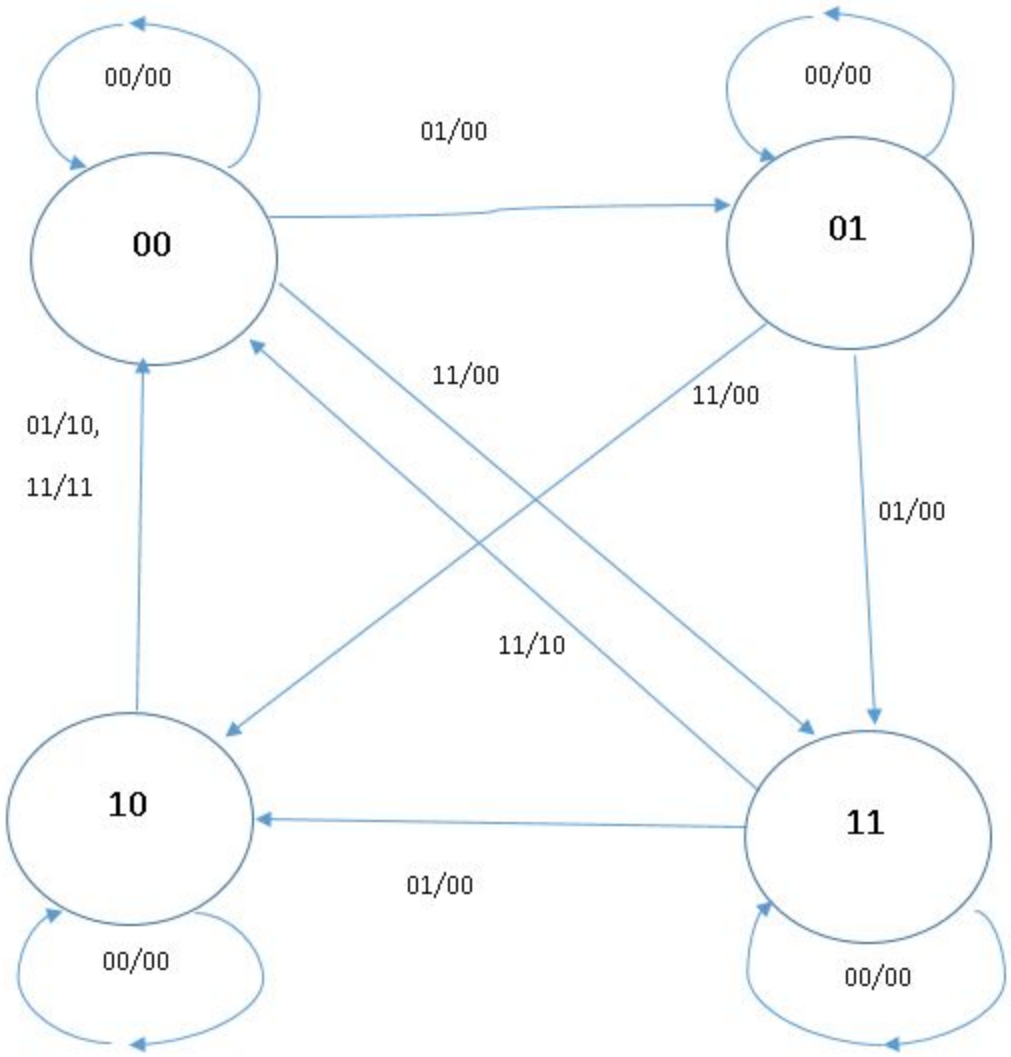


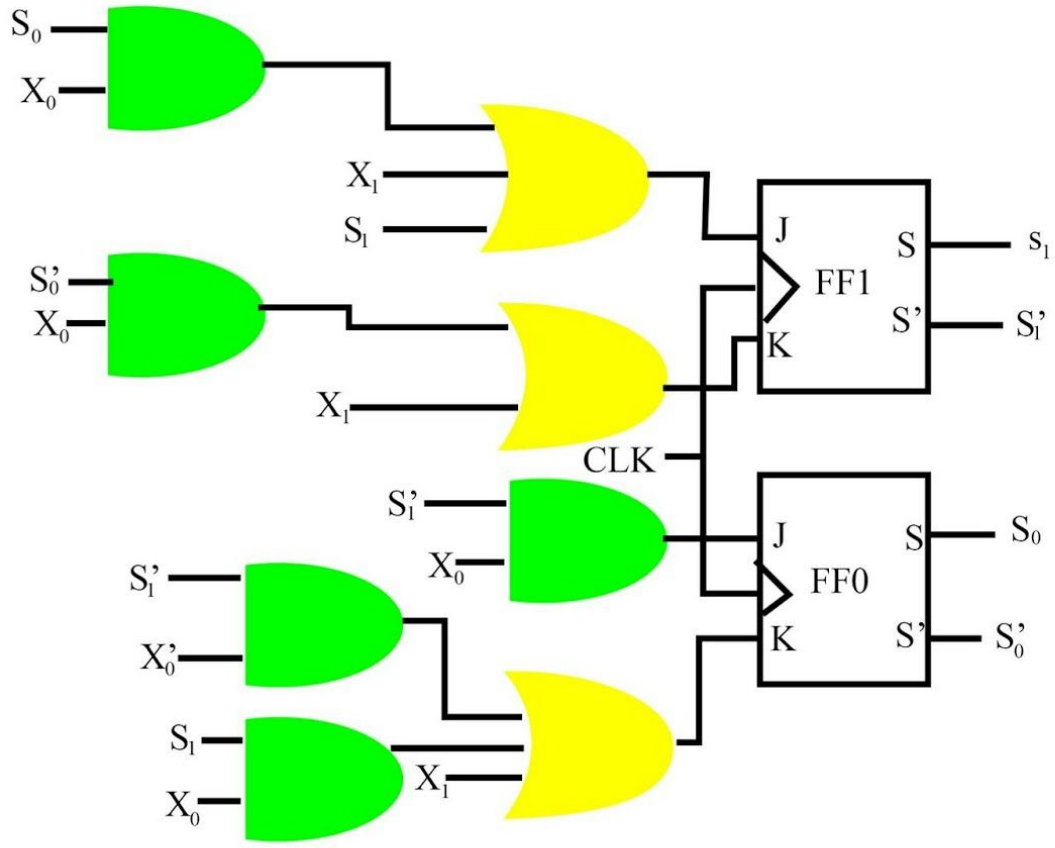

Figure 1: The Block Diagram of a Simple Vending Machine

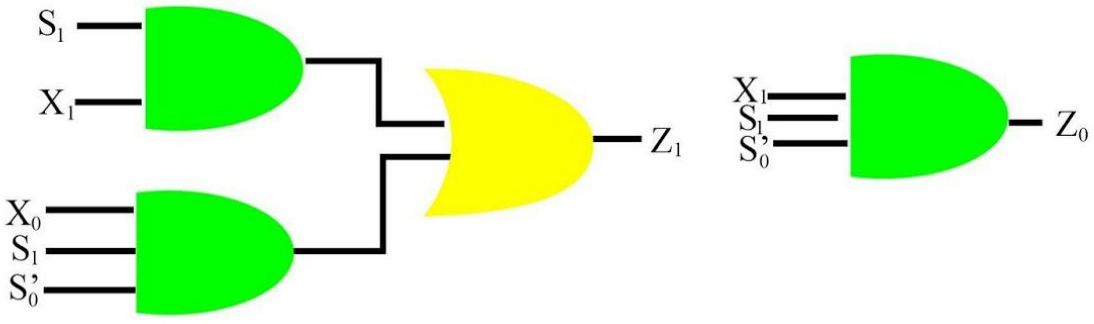
| Inputs | | Outputs | |
|---|---|---|---|
| **Variables** | **Values** | **Variables** | **Values** |
| Reset | {True (T),False (F)} | Release Gum (RG) | {T,F} |
| Coin | {Empty (E), Nickel (N), Dime (D) } | Return Nickel (RN) | {T,F} |

Table 1: The Inputs and Outputs of the $iKon$ Controller

## *The Functions of the Circuit*


00/00
01/00
00/00
00
01
11/00
11/00
01/10,
11/11
01/00
11/10
10
11
01/00
00/00
00/00


S0
X0
X1
S1
S0'
X0
X1
CLK
S1'
X0
S1'
X0'
S1
X0
X1
J
S
FF1
K
S'
s1
S1'
J
S
FF0
K
S'
S0
S0'


S1
X1
X0
S1
S0'
Z1
X1
S1
S0'
Z0

$J_1 = S_1 + x_1 + S_0x_0$
$K_1 = x_1 + S_0'x_0$
$J_0 = S_1'x_0$
$K_0 = x_1 + S_1'x_0' + S_1x_0$
$z_1 = S_1x_1 + S_1S_0'x_0$
$z_0 = S_1S_0'x_1$

***The Verilog Code***

*written by*
*Qing Yuan 104009336*
*Ying Bin Wu 104485521*

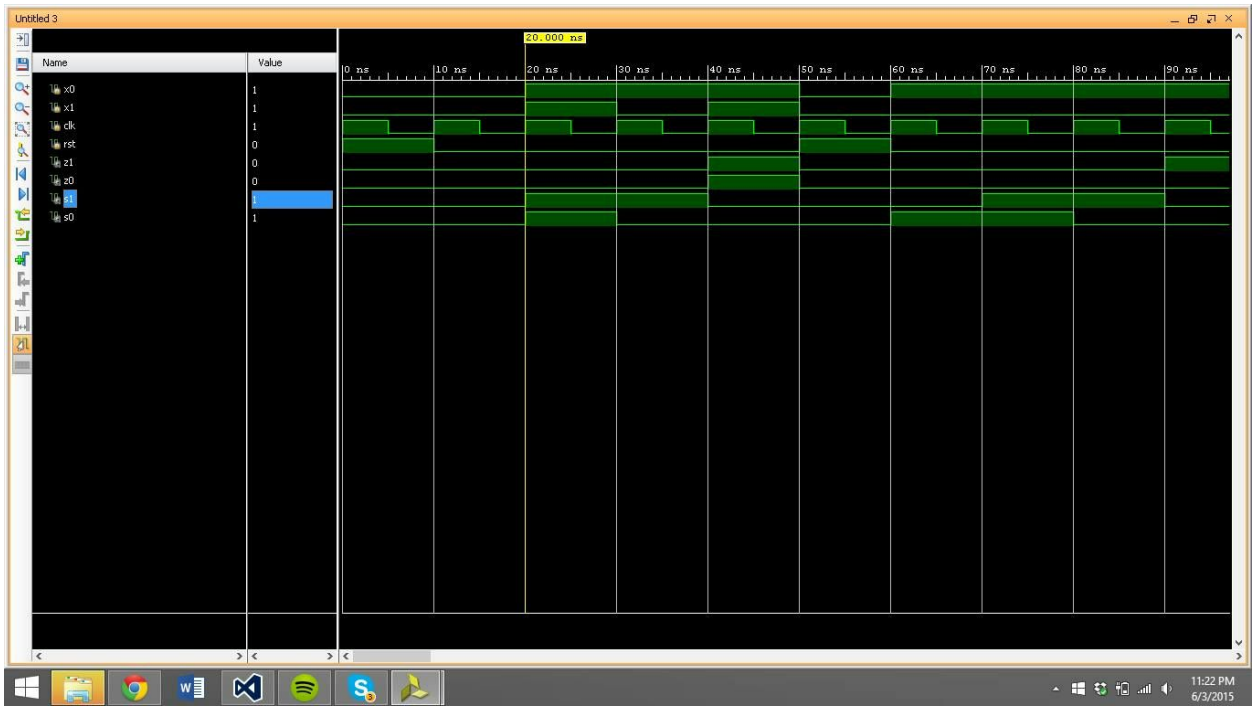
*//////////////////////////////////////////////////////////////////CODE FOR MAIN GUM DISPENSER MODULE*

```
module csm51a_proj3(
        input x0,
        input x1,
        input rst,
        input clk,
        output s1,
        output s0,
        output reg z1,
        output reg z0
);

wire J1;
wire K1;
wire J0;
wire K0;

jkff JKFF0(J0, K0, clk, rst, s0);
jkff JKFF1(J1, K1, clk, rst, s1);

always @(posedge clk, posedge rst) begin
                if (!rst) begin
                        z1 <= (s1&x1)|(s1&(~s0)&x0);
                        z0 <= (s1&(~s0)&x1);
                end
                else begin
```

```
                        z1 <= 0;
                        z0 <= 0;
                end
        end

//J1
wire temp1;
and (temp1, s0, x0);
or (J1, x1, s1, temp1);

//K1
wire temp2;
and (temp2, ~s0, x0);
or (K1, temp2, x1);

//J0
and (J0, ~s1, x0);

//K0
wire temp3;
wire temp4;
and (temp3, s1, x0);
and (temp4, ~s1, ~x0);
or (K0, x1, temp3, temp4);

endmodule


////////////////////////////////////////////////////////////////////////////////////////////////////CODE FOR J-K FLIP FLOP

module jkff(

   input J,
        input K,
        input clk,
        input rst,
        output reg Q
   );

        always @(posedge clk or posedge rst)
        begin
                        if(rst == 1)
                                begin
```

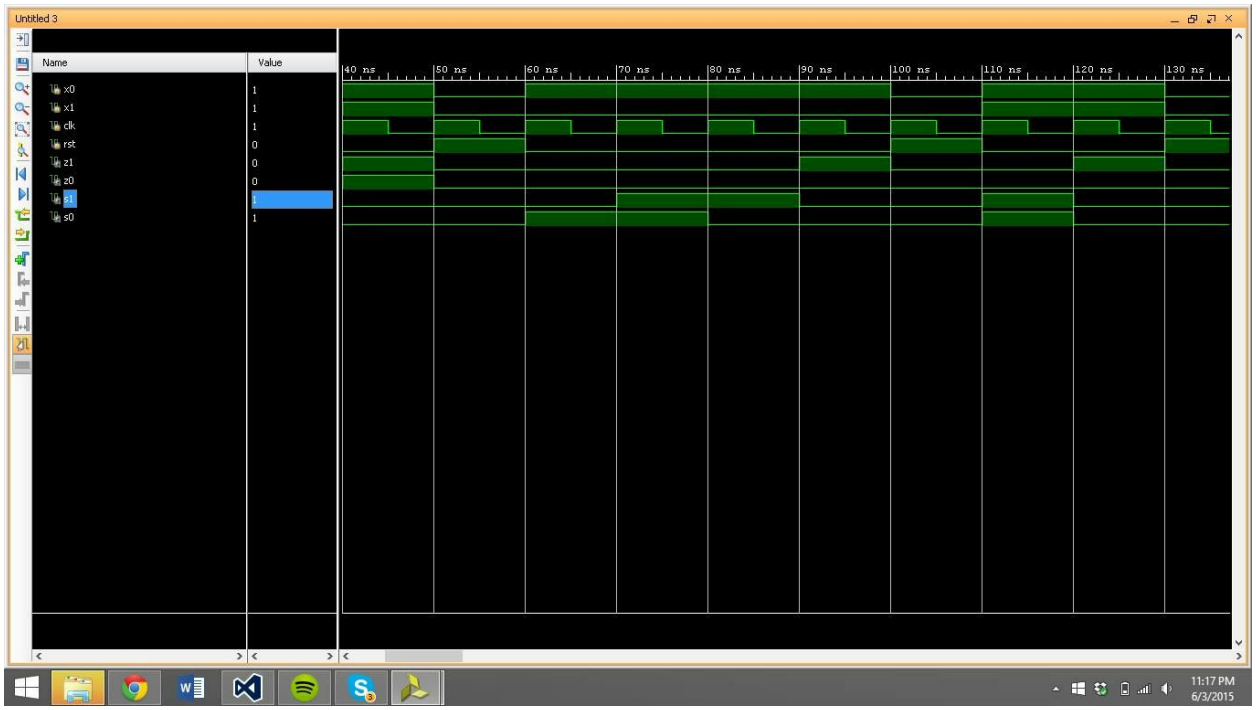
```
                    Q <= 0;
                    end
                else begin
                    case({J, K})
                    2'b00: Q <= Q;
                    2'b01: Q <= 1'b0;
                    2'b10: Q <= 1'b1;
                    2'b11: Q <= ~Q;
                    endcase
                end
        end

endmodule
```

***The Simulation Result***

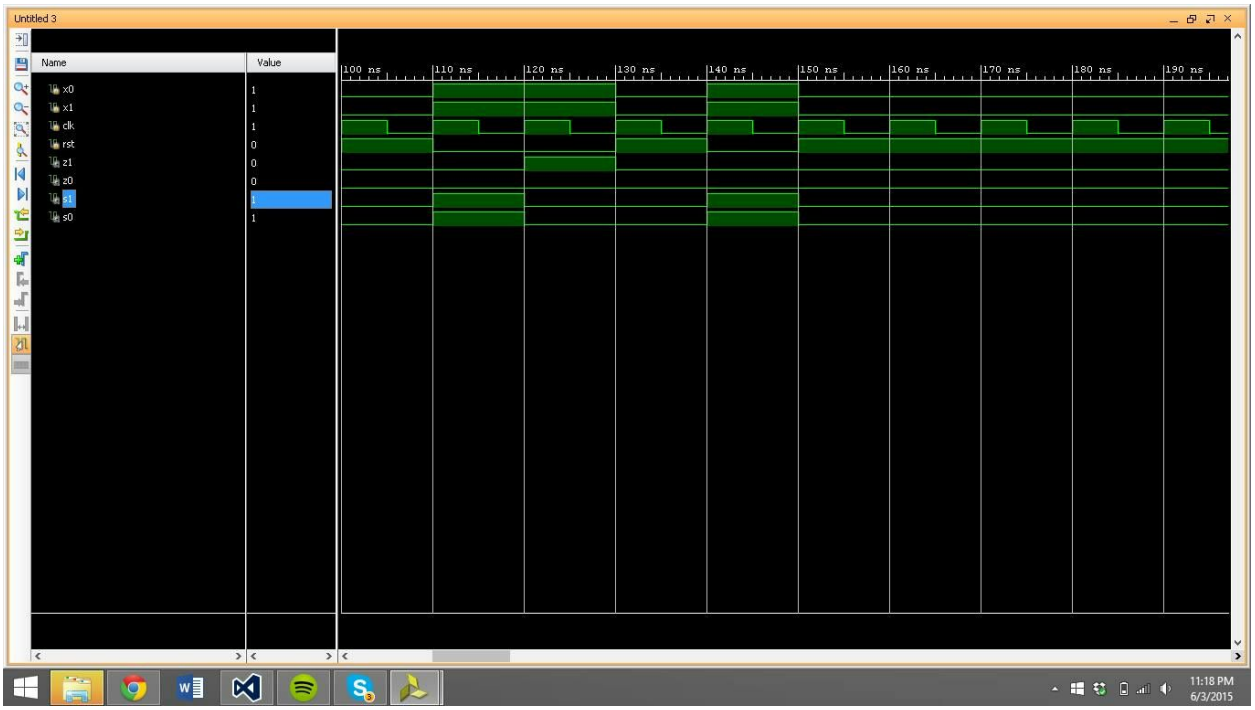



**The simulation for dime, nickel, dime from 20ns to 50ns**
**Input (x1,x0): (1,1) corresponds to dime, (0,1) corresponds to nickel**
**After first dime, state (s1, s0): (1,1)**
**output (z1, z0): (0,0)**
**After first nickel, state: (0,1)**
**output: (0,0)**
**After last dime, state: (0,0)**

**output: (1,1) returns gum and nickel**




**The simulation for nickel, nickel, nickel, nickel from 60ns to 100ns**
**Input: (0,1) corresponds to nickel**
**After first nickel, state: (0,1)**
**output (z1, z0): (0,0)**
**After second nickel, state: (1,1)**
**output: (0,0)**
**After third nickel, state: (1,0)**
**output: (0,0)**
**After fourth nickel, state: (0,0)**
**output: (1,0) returns gum**




**The simulation for dime, dime from 110ns to 130ns. Reset simulation from 140ns to 160ns**
**After first dime, state: (1,1)**
**output (z1, z0): (0,0)**
**After second dime, state: (0,0)**
**output: (1,0) returns gum**

**Reset is demonstrated as when the reset is turned on, the states go back to (0,0).**

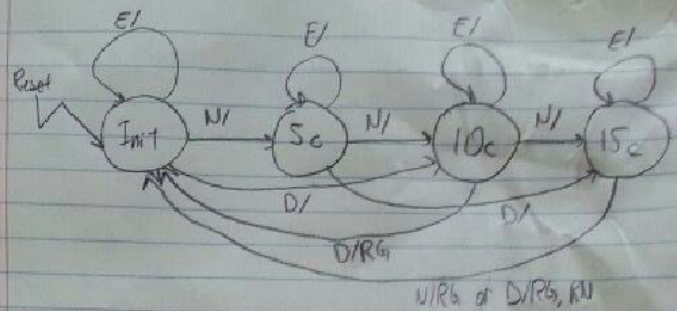
***The Design Review***

We started this project by designing the network on paper. We used k-maps to minimize the resulting network and after doing so, we implemented the gum dispenser system with verilog. Overall, the project was fairly straightforward and simple. The toughest part was trying to find a working vivado program because our licenses expired and we couldn't run simulations. Other than that, the project went very smoothly. We spent most of our time in the coding portion of this project. The network for this project is significantly bigger than the networks we had to design for the first two projects so the code was a lot more time consuming to write. Plus, figuring out how to link modules together (the jkff) was also a little bit challenging.

***Team Member Contributions***

The project as split evenly between us. We worked together on the design part and also the coding part. Ying Bin was responsible for drawing out all the state diagrams and sequential networks on paper while Qing was responsible for typing out all the code. The report is split into sections. Ying Bin was responsible for writing the abstract, the functions of the circuit, and team member contributions while Qing added the simulation results, design review, and finished the appendix.

***Appendix***




| PS | Input | | |
|---|---|---|---|
| | Coin = E | Coin = N | Coin = D |
| Init | Init, | 5c, | 10c, |
| 5c | 5c, | 10c, | 15c, |
| 10c | 10c, | 15c | Init, RG |
| 15c | 15c, | Init, RG | Init, RG, RN |

NS, output (RG, RN)

Excluding reset

| PS | Input ($x_1x_0$) | | | Input ($x_1x_0$) | | |
|---|---|---|---|---|---|---|
| $S_1S_0$ | 00 | 01 | 11 | 00 | 01 | 11 |
| 00 | 00,00 | 01,00 | 11,00 | 0-,0- | 0-,1- | 1-,1- |
| 01 | 01,00 | 11,00 | 10,00 | 0-,-1 | 1-,-0 | 1-,-1 |
| 11 | 11,00 | 10,00 | 00,10 | -0,-0 | -0,-1 | -1,-1 |
| 10 | 10,00 | 00,10 | 00,11 | -0,0- | -1,0- | -1,0- |

NS, output ($z_1z_0$) | $J_1K_1$, $J_0K_0$

$J_1$:

| $S_1S_0$ \ $x_1x_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 1 | - |
| 01 | 0 | 1 | 1 | - |
| 11 | - | - | - | - |
| 10 | - | 1 | - | - |

$J_1 = S_1 + x_1 + S_0x_0$

$K_1$:

| $S_1S_0$ \ $x_1x_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | - | - | 1 | - |
| 01 | - | - | 1 | - |
| 11 | 0 | 0 | - | - |
| 10 | 0 | 1 | - | - |

$K_1 = S_0'x_0 + x_1$

$J_0$:

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

$J_0 = S_1'X_0$

$K_0$:

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

$K_0 = X_1 + S_1'X_0' + S_1X_0$

$Z_1$:

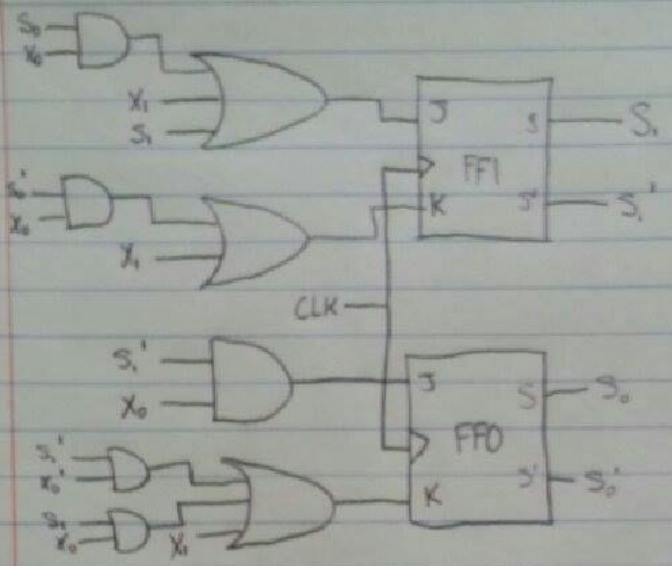
| $S_1S_0$ \ $X_1X_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | - |
| 01 | 0 | 0 | 0 | - |
| 11 | 0 | 0 | 1 | - |
| 10 | 0 | 1 | 1 | - |

$Z_1 = S_1X_1 + X_0S_1S_0'$

$Z_0$:

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

$Z_0 = X_1S_1S_0'$


J
S
FF1
K
S'
CLK
J
S
FF0
K
S'