

CS M51A and EE M16 Spring 2015 Section 1

Logic Design of Digital Systems

Dr. Yutao He

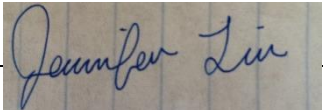
Verilog Lab #3 - Design of Sequential Systems

Due: June 2, 2015

Team ID: T09

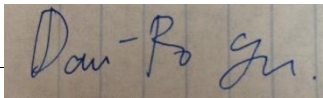
(1) Name: Lin Jennifer

Student ID: 904299916

Signature: 

(2) Name: Yu Dau-Po

Student ID: 504451468

Signature: 

Date: 6/2/2015

Result	
Correctness	
Creativity	
Report	
Total Score	

Verilog Lab #3 Project Requirement

Dr. Yutao
He

Due: 6/2/2015

1 Objectives

The last project in this quarter is intended to: (1) get you familiar with the design flow of building a sequential circuit from the high-level specification to the final working system composed of logic gates and flip-flops, (2) obtain the hands-on experiences of using flip-flops and connecting the clock signal properly in the sequential system.

2 The Project Description

2.1 The High-Level Specification

The finite state machine (FSM) to be built in the project is a simple vending machine controller called iKon. A controller is an important class of sequential systems that produces a set of control signals as its states are traversed. These control signals are used to determine actions performed in another system.

Here is how the iKon controller is supposed to work. The vending machine delivers a package of gum after it has received 20 cents in coins. The machine has a single coin slot that accepts only nickels and dimes, one coin at a time. A mechanical sensor indicates to the controller if any coin has been inserted into the coin slot and which type of coin it is. A reset button causes two actions in case a customer changes his/her mind: (1) set the controller to the initial state, and (2) drive another mechanism to return all deposited coins. As a result, the controller does not have to generate a control signal that commands return of all coins when the reset button is pressed. The outputs of the controller cause a single package of gum to be released down a chute to a customer, or return 5 cents changes if necessary.

The high-level interface between the iKon controller and the rest of the vending machine is shown in Figure 1. The inputs, outputs and the states of the controller are listed in Table 1 and Table 2, respectively.

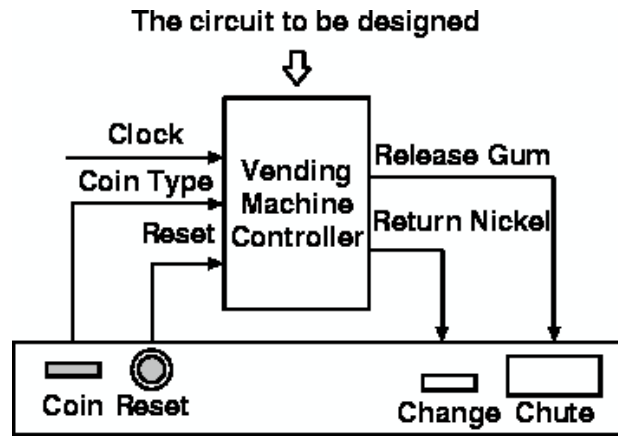


Figure 1: The Block Diagram of a Simple Vending Machine

Inputs		Outputs	
Variables	Values	Variables	Value
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 iK on Controller

The complete behavior of the iK on controller can be specified by the state diagram in Figure 2. To keep the state diagram clear and readable, those transitions caused by the reset input signal are not included. It is obvious though that if the reset input is asserted at any state, the controller will go to the initial state. In addition, only transitions that explicitly cause a state change, and the output in states where it is asserted, are included in the diagram. For example, the transition from state 15c to initial state is labeled with N/RG or D/RG, RN, which means that the transition can only happen in two cases (except when the reset is asserted): (1) if another nickel is deposited, then a package of gum is released, or (2) if a dime is deposited, then a package of gum is released and 5c (a nickel) is returned.

2.2 The Binary-Level Specification

2.2.1 Encoding Schemes of Inputs

You must use the encoding schemes shown in Table 3 for the inputs.

States	Descriptions
Init	the initial state
5c	the amount of deposited coins is 5 cents
10c	the amount of deposited coins is 10 cents
15c	the amount of deposited coins is 15 cents

Table 2: The States of the iK on Controller

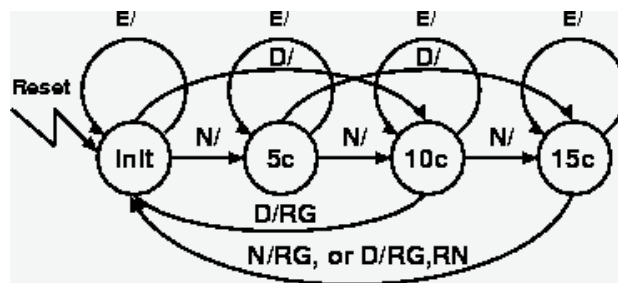


Figure 2: The State Diagram of the iK on Controller

2.2.2 Encoding Schemes of Outputs

You must use the encoding schemes shown in Table 4 for the outputs.

2.2.3 Encoding Schemes of States

You must use the encoding schemes shown in Table 5 for the states.

2.3 Additional Requirements for The Implementation

In addition to complying with the high-level and binary-level specifications, your design must also satisfy the following requirements:

Coin	x_1x_0	Reset	r
Empty	00	False	0
Nickel	01	True	1
Dime	11		

Table 3: The Encoding Scheme for Inputs

RG	z_1	RN	z_0
False	0	False	0
True	1	True	1

Table 4: The Encoding Scheme for Outputs

State	s_1s_0
Init	00
5c	01
10c	11
15c	10

Table 5: The Encoding Scheme for States

- **Combinational Logic:** The combinational logic in the system should be implemented with the minimal two-level OR-AND network. Note that NOT gates used to generate negated input variables are not counted as one additional level.
- **Flip-flops:** You should use JK flip-flops to store s_1 and s_0 .

2.4 Extra Credits

This part is not for the minimal requirement but for those highly motivated minds.

It can be easily seen that there is one basic limitation in the original iK on controller - it does not take quarters. Suppose that TripleMint Corp., the buyer of the iK on controller, is not satisfied with the design and complains to your boss in your company, NeoElekTech, who then asks you to design the next generation of iK on called iK onPlus by modifying your original design. In addition, your company NeoElekTech recently purchased a batch of multiplexers at very cheap price through a secret channel.

As a result, your boss decided to implement combinational logic of the iK onPlus controller with MUXes instead of OR, AND gates to reduce the cost. Design and implement the iK onPlus controller.

3 Report Outline

Each team is required to submit one report that provides complete documentation of your project including the detailed design steps. As in all technical writing, its purpose is to communicate your work with your colleagues in an efficient and professional way so that your design can be upgraded and maintained even if you are no longer around. As a result, the report should be clear, concise and complete and should contain the following parts:

(1) Title Page

It is provided and you just need to fill in your information in the blanks.

(2) Abstract

This is the brief high-level description of the project in plain English text.

(3) The Functions of the Circuit

It is part of the design work for you to obtain the binary-level specification for the function of the circuit in the form of switching expressions and the schematic. This section should present both minimal switching expressions in OR-AND form and the schematic of the circuit composed of logic gates and flip-flops. Detailed steps of obtaining them should go to Appendix.

(4) The Verilog Code

The Verilog code you write is the implementation of the circuit. You must include it in your report with the names and SID numbers of your team members included.

(5) The Simulation Result

You have to demonstrate that your implementation works as specified by showing the simulation result. In your simulation, you should show the following three sequences: (1) D, N, D; (2)N, N, N, N, and (3)D, Reset. Please clearly write down the necessary information on the waveform diagram so that one of your colleagues who does not know anything about your project could understand the behavior of the system you are trying to implement (Uncommented timing diagrams will be considered incomplete and points will be marked down).

(6) The Design Review

This section includes a summary of your experiences throughout the project. It should be no more than two pages and may include such topics as what you have learned, problem encountered during the implementation and the workarounds you came up with, the approach you used, the most important aspects of the project for you, where you spent most of your time and suggestions you want to make. In particular, you may include here the new design scheme you proposed for the iK onPlus controller and alternative implementation with MUXes for extra points.

(7) Team Member Contributions

Teamwork requires that each member be responsible and assume a relatively equal share of workload. In this section, a detailed description on each member's responsibility and contribution should be presented clearly, including an estimate of percentage of efforts on the project and a summary list of each member in the project.

Each member requires to review and sign the final report on the cover page.

(8) Appendix - The detailed design worksheet

This part must be included in your report and must show the complete worksheet during the pencil-and-paper design. It should contain:

- 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 OR-AND switching expressions and the final schematic of the circuit.

4 Project Submission

You should submit one zipped file named with "Txx.zip" via the on-line submission link. The zipped file should consist of four separate files:

1. The pdf file of your report. It must be named as "Txx.pdf" where xx is your Team ID assigned to you, that is, your report should be called Txx.pdf;
2. The Verilog file of your circuit implementation. It should be named as-csm51a proj3.v.
3. The Verilog file of your implementation of a JK flipflop. It should be named as jkff.v.
4. The testbench file. It should be named as csm51a proj3 tb.v.
5. If you complete the extra point problem, please also include the verilog file for the enhanced controller named- csm51a proj3 bonus.v and the testbench file named esm51a proj3 bonus tb.v.

5 Project Deadline

The report is due by midnight (11:59:59pm) on June 2 (Tuesday), 2015. The deadline must be observed strictly and late submissions will be subject to penalty.

6 Design Tips

6.1 About JK Flip-Flops

In practice, the majority of digital designs that are modeled in Verilog are clocked, synchronous systems using edge-triggered flip-flops. You need to design a JK-flip-flop module in a separate file called jkff.v and use it.

Asynchronous input means that the effect of activating the input takes place immediately, independent of the clock signal. It is useful because when

designing a sequential system it is important to define the initial state of operation. This state must be valid when the circuit is first powered on, or after an initialization phase (activating a RESET signal, for example). In general, the initial state is chosen such that all state bits are zeroes. The value zero in all FFs is easily obtained by connecting the clear input of each FF together to the same signal line. Before the circuit is used, this line is made active (high or low, depending on the specific device being used) for a short period of time. This way, the initial state is known when the circuit starts to operate.

6.2 About Input RESET

Input RESET is used to set the controller to a valid initial state at any moment. Since the initial state is encoded with all zeros (see Table 5), you can implement the input RESET signal simply by connecting it directly to the asynchronous clear input of a JK flip-flop. This way, whenever the input RESET is asserted (has value 1), all flip-flops will be set to state 0. As a result, you do not need to include the input RESET as a variable when you design the circuit.

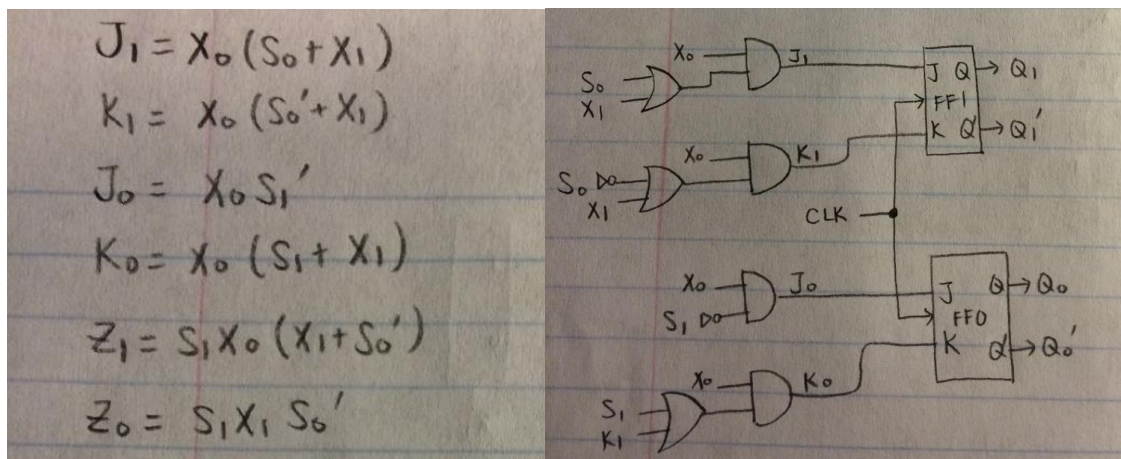
6.3 About The Clock

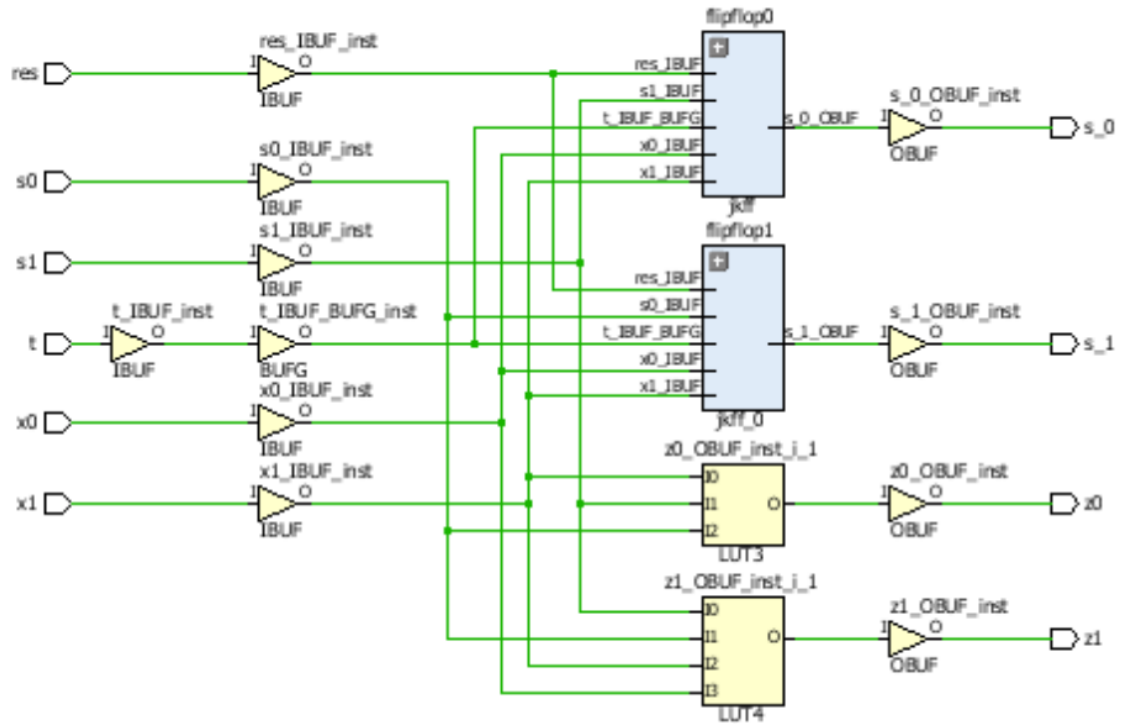
During the normal operation, everything in a synchronous sequential system is controlled by the clock. As a result, the clock input of each JK flip-flop must be connected to the clock signal. You will set the clock waveform when you generate your simulation waveforms.

Abstract

For this project, we implemented a vending machine controller that takes in nickels and dimes. When the machine receives 20 cents, a package of gum will be released and returns change if necessary. A reset button is created that allows customers to get all deposited money back, and the machine will return to its initial state. The combinational logic of this system is implemented using a two-level OR-AND network, and JK flip-flops are used to store s_1 and s_0 , which indicate the state.

Functions of the circuit





Verilog Code

```

module csm51a_proj3(x1,x0,s1,s0,t,res,z1,z0,s_1,s_0);
input x1,x0,s1,s0,t,res;
output z1, z0, s_1, s_0;
wire j1,j0,k1,k0,Q1,Q0;
assign j1= (s0 | x1) & x0;
assign k1= x0 & (x1 | ~s0);
assign j0= x0 & (~s1);
assign k0= x0 & (s1 | x1);
assign z1= s1 & x0 & ((~s0) | x1);
assign z0= s1 & x1 & (~s0);
jkff flipflop1(.J(j1), .K(k1), .clk(t), .reset(res), .Q(Q1));
jkff flipflop0(.J(j0), .K(k0), .clk(t), .reset(res), .Q(Q0));
assign s_1=Q1;
assign s_0=Q0;

endmodule

```

```

module jkff(J, K, clk, reset, Q);
input J, K, clk, reset;
output Q;
reg Q;

```

```

always @(posedge clk or posedge reset)
begin
    if(reset == 1'b1)
        begin
            Q <= 1'b0;
        end
    else
        begin
            if(J == 1'b0 && K == 1'b0)
                Q <= Q;else
            if(J == 1'b1 && K == 1'b0)
                Q <= 1'b1;else
            if(J == 1'b0 && K == 1'b1)
                Q <= 1'b0;else
            if(J == 1'b1 && K == 1'b1)
                Q <= ~Q;
        end
    end
endmodule

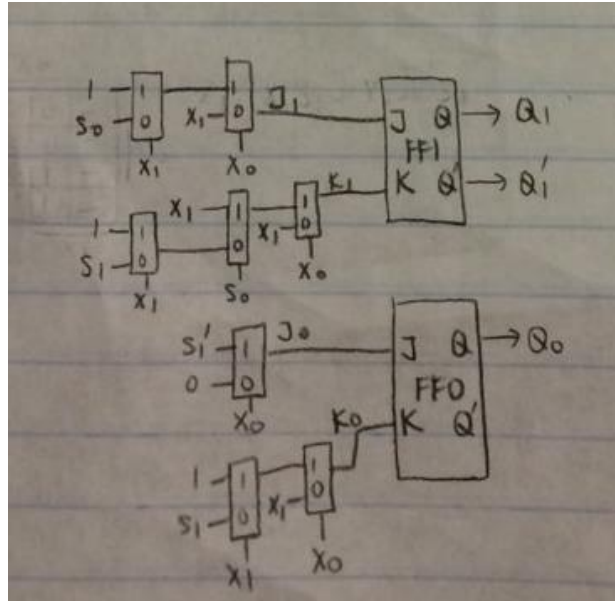
```

Design Review

This project gave us hands-on experience on building a sequential circuit using both combinational logic and flip-flops. Many real-world systems are sequential as opposed to combinational, therefore this assignment is definitely tougher than the previous two projects, but it is also one that we learned the most from.

During this project, we were really careful at every step, especially when we filled in the K-maps according to the JK flip-flop excitation table. A single error at this step would affect the maxterm minimization process and yield incorrect results. One of the toughest part of this project is figuring out how to implement these networks on Verilog. Implementing combinational circuits on Verilog is rather straightforward, but when it comes to sequential circuits, we have to take into account the clock signals and previous states. Therefore we spent a significant amount of time figuring out the correct syntax.

In addition, rather than using a two level OR-AND network, we realized that the system could be implemented with multiplexers instead like the following:



Moreover, we made an effort in building a more advanced version of this vending machine that would accept not only nickels and dimes, but also quarters. We followed the specification and implemented the combinational logic of this advanced vending machine with multiplexers rather than OR and AND gates. Details of this implementation are included under the extra credit section.

Team Member Contribution

Workload is equally shared between both of us. Since this project is much more difficult than the previous two assignments, we spent a lot of time discussing together, trying to figure out how to approach and implement the required sequential system. Therefore we did most of the work together and checked on each other to make sure we are on the right track.

Extra Credit

$$Z_1 = X_1 X_0' + X_1 S_1 + X_0 S_0' S_1$$

$$= X_1 X_0' + X_1 S_1 + (X_0 S_0' S_1)(X_1 + X_1')$$

$$1) \underline{X_1} = X_0' + S_1 + X_0 S_0' S_1$$

$$2) \underline{X_1'} = X_0 S_0' S_1$$

$$1) \underline{X_0'} + S_1 + X_0 S_0' S_1$$

$$= X_0' + S_1 (X_0 + X_0') + X_0 S_0' S_1$$

$$a) \underline{X_0} = S_1 + S_0' S_1 = S_1 (1 + S_0) = S_1$$

$$b) \underline{X_0'} = 1 + S_1 = 1$$

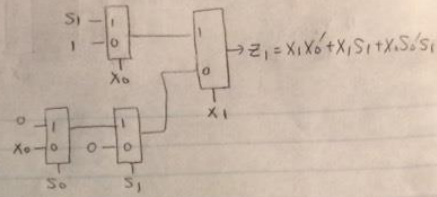
$$2) \underline{X_0 S_0' S_1}$$

$$a) \underline{S_1} = X_0 S_0'$$

$$b) \underline{S_1'} = 0$$

$$a) \underline{S_0} = 0$$

$$S_0' = X_0$$



$$Z_0 = X_1 X_0' + X_1 S_0' S_1$$

$$1) \underline{X_1} = X_0' + S_0' S_1$$

$$X_1' = 0$$

$$1) \underline{X_0'} + S_0' S_1$$

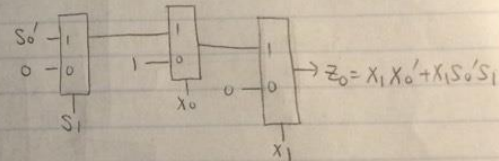
$$= X_0' + (S_0' S_1)(X_0 + X_0')$$

$$a) \underline{X_0} = S_0' S_1$$

$$b) \underline{X_0'} = 1 + S_0' S_1 = 1$$

$$a) \underline{S_1} = S_0'$$

$$S_1' = 0$$



$$J_1 = X_1 X_0 + S_1 X_0$$

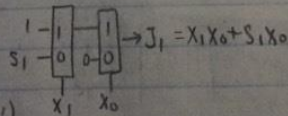
$$1) \underline{X_0} = X_1 + S_1$$

$$X_0' = 0$$

$$1) \underline{X_1 + S_1 (X_1 + X_1')}$$

$$\underline{X_1} = 1 + S_1 = 1$$

$$X_1' = S_1$$



$$K_1 = X_1 + X_0 S_0'$$

$$= X_1 (X_0 + X_0') + X_0 S_0'$$

$$1) \underline{X_0} = X_1 + S_0'$$

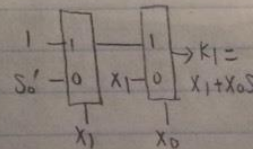
$$X_0' = X_1$$

$$1) \underline{X_1 + S_0'}$$

$$= X_1 + S_0' (X_1 + X_1')$$

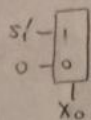
$$\underline{X_1} = 1 + S_0' = 1$$

$$X_1' = S_0'$$



$$J_0 = X_0 S_1'$$

$$1) \frac{X_0}{X_0'} = \frac{S_1'}{0}$$



$$K_0 = X_1 + X_0 S_1$$

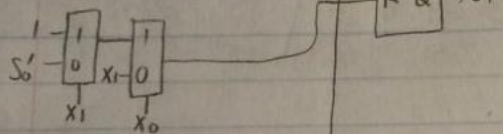
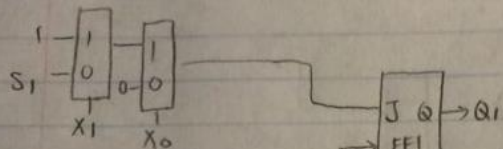
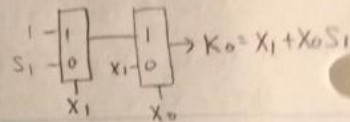
$$= X_1(X_0 + X_0') + X_0 S_1$$

$$1) \frac{X_0}{X_0'} = \frac{X_1 + S_1}{X_1}$$

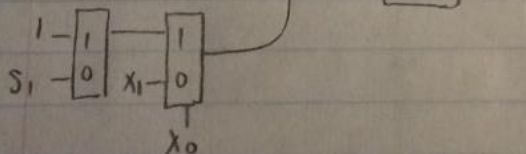
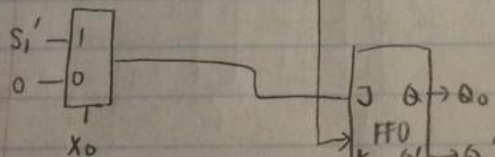
$$1) \frac{X_1 + S_1}{X_1 + S_1} = \frac{X_1 + S_1(X_1 + X_1')}{X_1 + S_1(X_1 + X_1')}$$

$$X_1' = 1 + S_1 = 1$$

$$X_1' = S_1$$



CLK



Appendix

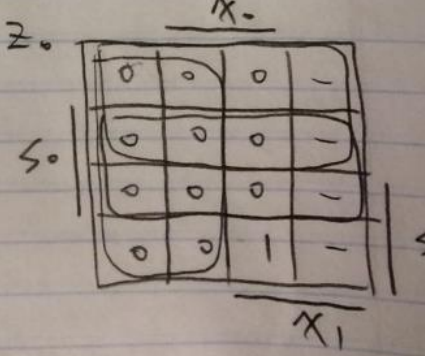
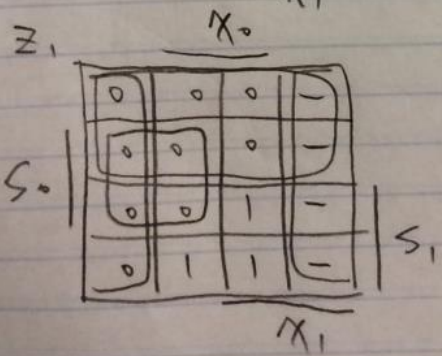
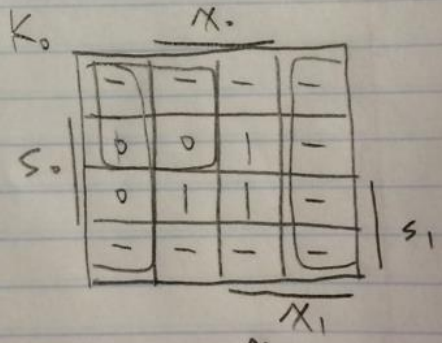
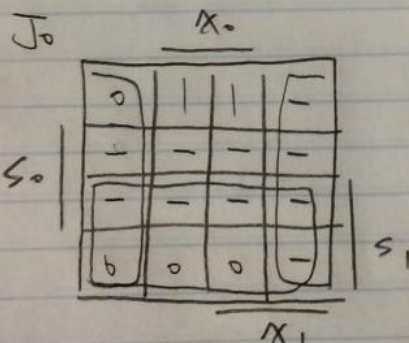
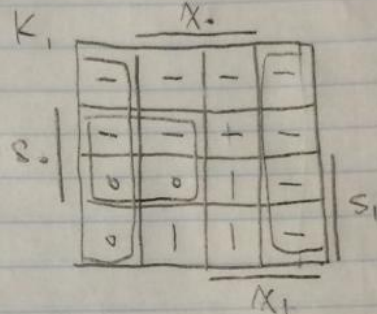
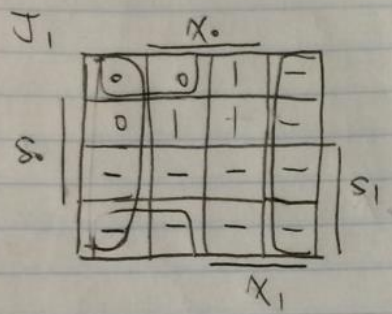
input $x(t) = (x_1, x_0)$, $x_i \in \{0, 1\}$, $r(t) \in \{0, 1\}$
 output $z(t) = (z_1, z_0)$, $z_i \in \{0, 1\}$
 state $s(t) = (\text{init}, 5c, 10c, 15c)$
 initial state $s(0) = \text{init}$
 input x_1, x_0

state	$s_1 s_0$
init	00
5c	01
10c	11
15c	10

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

MS, $s_1 s_0$ output $z_1 z_0$

if $r(t) = 1$
 $\Rightarrow s(t+1) = 00$



$$\begin{aligned} \bar{J}_1 &= X_0 (S_0 + X_1) & K_1 &= X_0 (X_1 + S_0') \\ J_0 &= X_0 S_1' & K_0 &= X_0 (S_1 + X_1) \\ Z_1 &= S_1 X_0 (S_0' + X_1) & Z_0 &= S_1 X_1 S_0' \end{aligned}$$

