ECED2200 – Digital Circuits

Finite State Machines

General Notes

- See updates to these slides: www.newae.com/teaching
- These slides licensed under 'Creative Commons Attribution-ShareAlike 3.0 Unported License'
- These slides are not the complete course – they are extended in-class
- You will find the following references useful, see www.newae.com/teaching for more information/links:
  - The book "Bebop to the Boolean Boogie" which is available to Dalhousie Students
  - Course notes (covers almost everything we will discuss in class)
  - Various websites such as e.g.: www.play-hookey.com
  - The book "Contemporary Logic Design", which was used in previous iterations of the class and you may have already

Finite State Machine (FSM)

- **Step 1:** Understand the Problem
- **Step 2:** Draw Initial State Diagram
- **Step 3:** Minimize State Diagram (if possible)
- **Step 4:** Perform State Assignment, draw state transition table
- **Step 5:** Choose Flip-Flops / Technology
- **Step 6:** Implement FSM

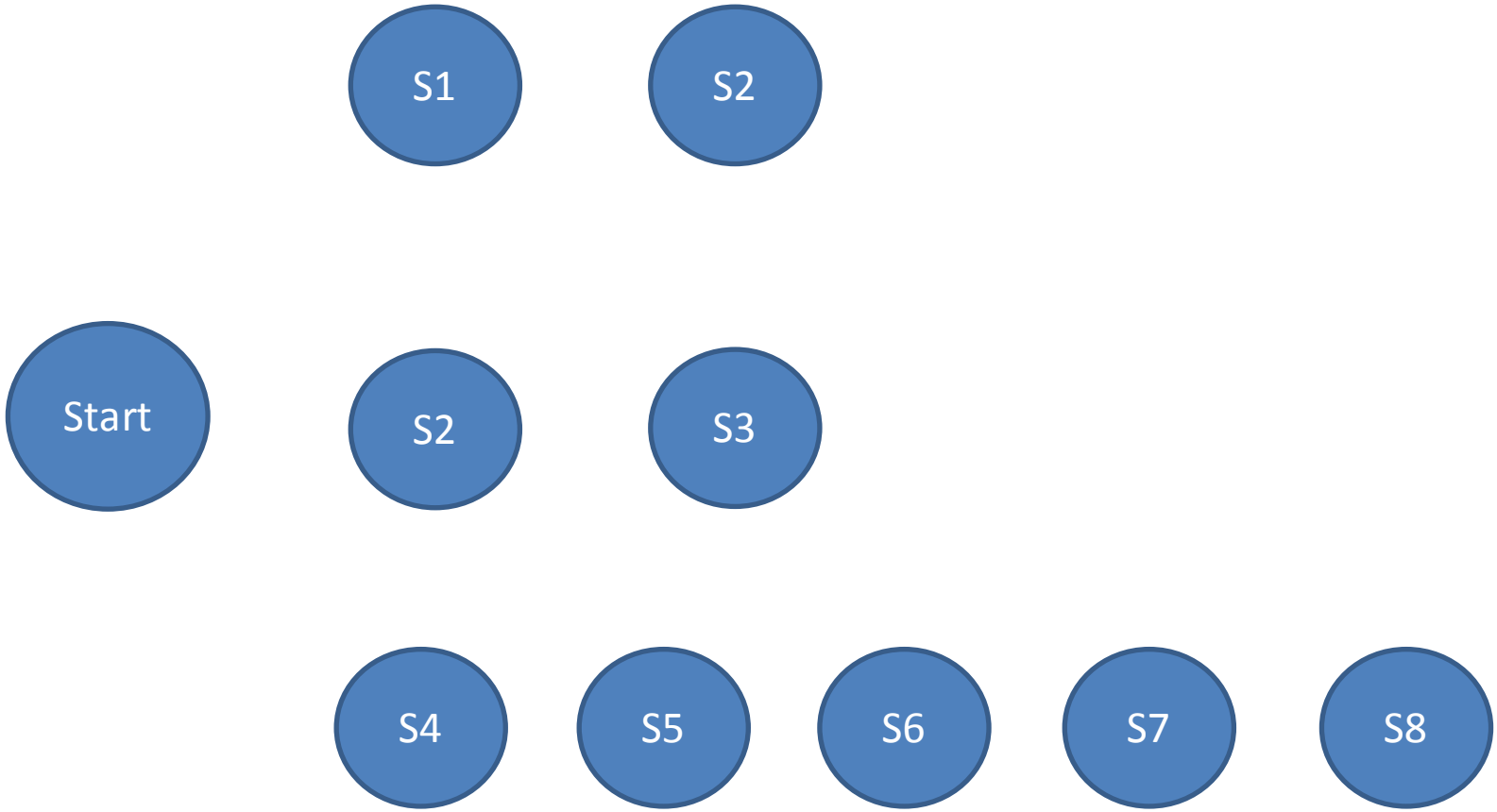
Step 1: Understand the Problem

e.g.: Design a vending machine which takes 25¢ or $1 coins, and releases candy when $1.25 has been deposited. No changes is given, assuming you have ‘Q’ and ‘L’ (Quarter & Loonie) inputs.

Step 1: Understand the Problem

Step 1: Understand the Problem

Step 2: Draw Initial Diagram



Step 3: Draw Minimized State Diagram

Step 3: Symbolic State Table

State	Quarter	Loonie	Next State	Release
0.00	0	0	0.00	0
0.00	0	1	1.00	0
0.00	1	0	0.25	0
0.00	1	1	?	0
0.25	0	0	0.25	0
0.25	0	1	1.25	0
0.25	1	0	0.50	0
0.25	1	1	?	0
0.50	0	0	0.50	0
0.50	0	1	1.25	0
0.50	1	0	0.75	0
0.50	1	1	?	0
0.75	0	0	0.75	0
0.75	0	1	1.25	0
0.75	1	0	1.00	0
0.75	0	0	?	0
1.00	0	0	1.00	0
1.00	0	1	1.25	0
1.00	1	0	1.25	0
1.00	1	1	?	0
1.25	?	?	0.00	1

Step 4: Perform State Assignment

000 = 0.00

001 = 0.25

010 = 0.50

011 = 0.75

100 = 1.00

101 = 1.25

Step 4: State Transition Table + Outputs

Present State	Quarter	Loonie	Next State	Release
000	0	0	000	0
000	0	1	100	0
000	1	0	001	0
000	1	1	?	0
001	0	0	001	0
001	0	1	101	0
001	1	0	010	0
001	1	1	?	0
010	0	0	010	0
010	0	1	101	0
010	1	0	011	0
010	1	1	?	0
011	0	0	011	0
011	0	1	101	0
011	1	0	100	0
011	0	0	?	0
100	0	0	100	0
100	0	1	101	0
100	1	0	101	0
100	1	1	?	0
101	?	?	000	1

Step 5: Choose Flip-Flops

- D flip-flops simplify design
- JK flip-flops simplify logic

D Flip-Flops (Partial Example)

State	Quarter	Loonie	Next State	D Flip-Flop Entries		
				Da	Db	Dc
000	0	0	000	0	0	0
000	0	1	100	1	0	0
000	1	0	001	0	0	1
000	1	1	?	?	?	?
001	0	0	001	0	0	1
001	0	1	101	1	0	1
001	1	0	010	0	1	0
001	1	1	?	?	?	?
010	0	0	010	0	1	0
010	0	1	101	1	0	1
010	1	0	011	0	1	1
010	1	1	?	?	?	?
011	0	0	011	0	1	1
011	0	1	101	1	0	1
011	1	0	100	1	0	0
011	0	0	?	?	?	?
100	0	0	100	1	0	0
100	0	1	101	1	0	1
100	1	0	101	1	0	1
100	1	1	?	?	?	?
101	?	?	000	0	0	0

D Flip-Flops (Partial Example)

State	Quarter	Loonie	Next State	D Flip-Flop Entries		
				Da	Db	Dc
000	0	0	000			
000	0	1	100			
000	1	0	001			
000	1	1	?			
001	0	0	001			
001	0	1	101			

Q	Q^+	D
0	0	0
0	1	1
1	0	0
1	1	1

JK Flip-Flops (Partial Example)

State	Quarter	Loonie	Next State	JK Flip-Flop Entries					
				Ja	Ka	Jb	Kb	Jc	Kc
000	0	0	000						
000	0	1	100						
000	1	0	001						
000	1	1	?						
001	0	0	001						
001	0	1	101						

Q	Q^+	J	K
0	0	0	?
0	1	1	?
1	0	?	1
1	1	?	0

Step 6: Implement FSM

- Use K-maps for inputs to FF & for outputs

Choice of States + FF

Should be apparent there are many ways to encode state, and many choice of FF to use. We are unlikely to 'happen' to choose best implementation, so will use design tools in real life…

FINITE STATE MACHINE – DESIGN EXAMPLE #1

Specifications

Design a simple stoplight. Assume one pair of lights is North-South (NS), one is East-West (EW). Pattern is:

- Green 10 seconds
- Yellow 1 second
- Red 1 second

In each direction – that is to say 1s overlap where both sets red. Assume you have 1 Hz clock available.

Step 1: Understand Problem

Step 2: Initial State Diagram

Step 3: Minimized State Diagram

Step 4: State Transition Table

A	B	C	CNT=10 (D)	A+	B+	C+	Note
0	0	0	X	0	0	1	Red - Reset
0	0	1	0	0	0	1	Green NS
0	0	1	1	0	1	0	Green NS
0	1	0	X	0	1	1	Yellow NS
0	1	1	X	1	0	1	Red - Reset
1	0	1	0	1	0	1	Green EW
1	0	1	1	1	1	0	Green EW
1	1	0	X	0	0	0	Yellow EW

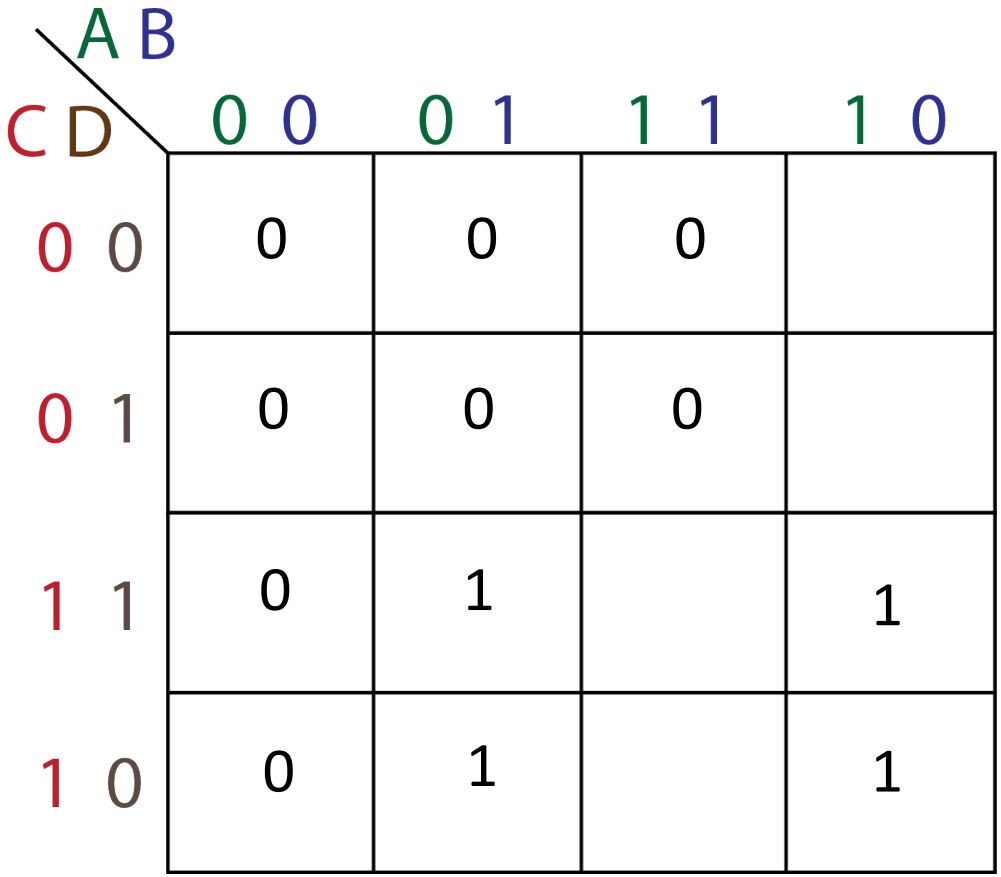
Step 5: Choose Flip-Flop

We will use D flip-flops for design convience

Step 6: Implement

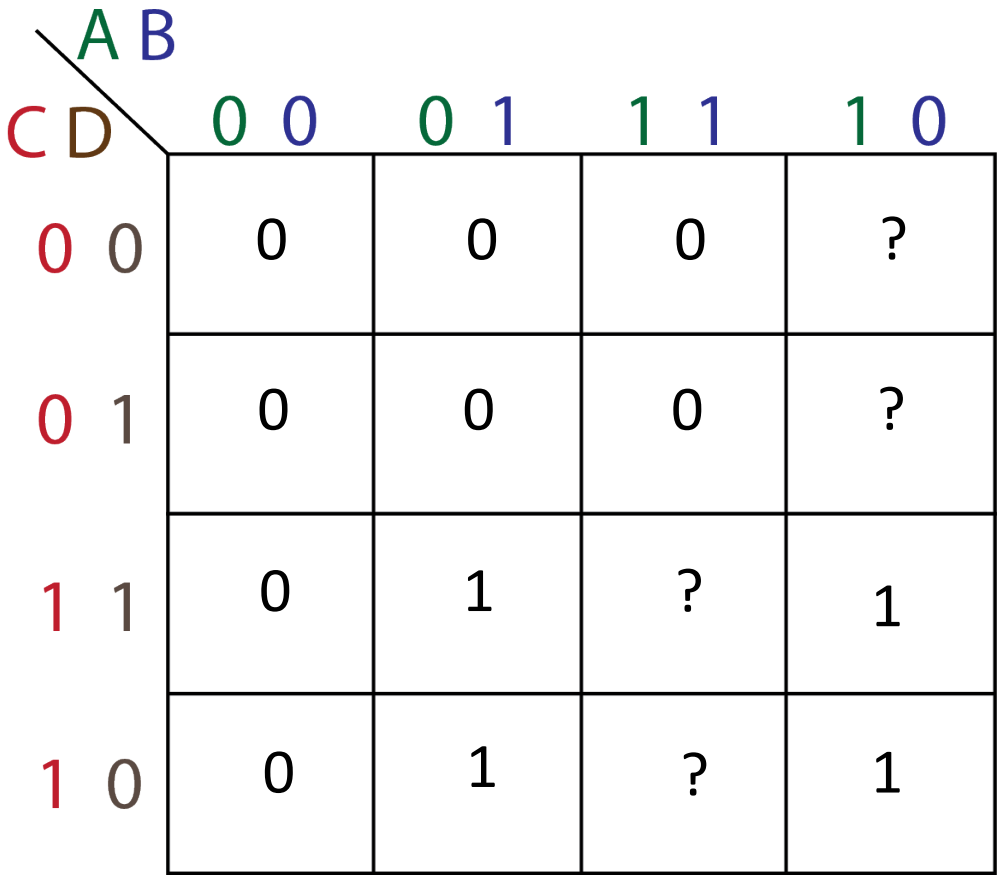
Following shows K-maps of state transition + outputs

K-Mapping: A State



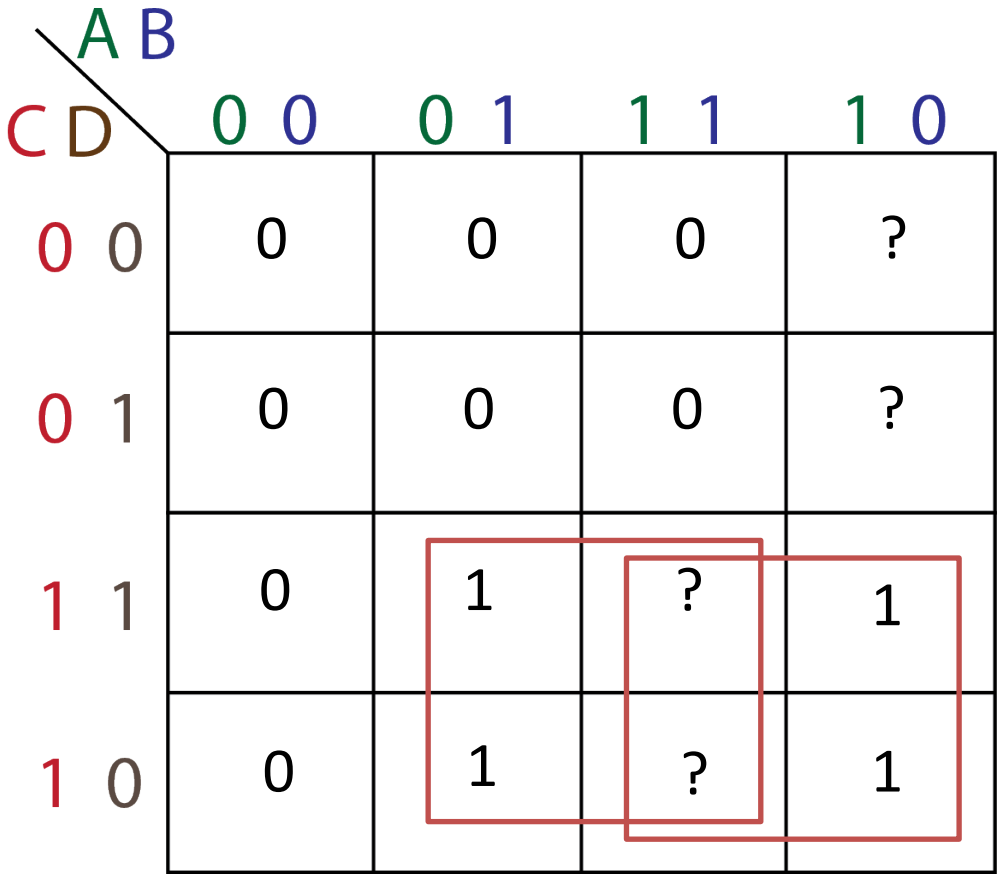
Note: Here both 1's and 0's are drawn on K-Map. This is because the given State Transition Table didn't include the don't care states. We can fill in 1's and 0's from given, and any blank are don't cares.

K-Mapping: A State



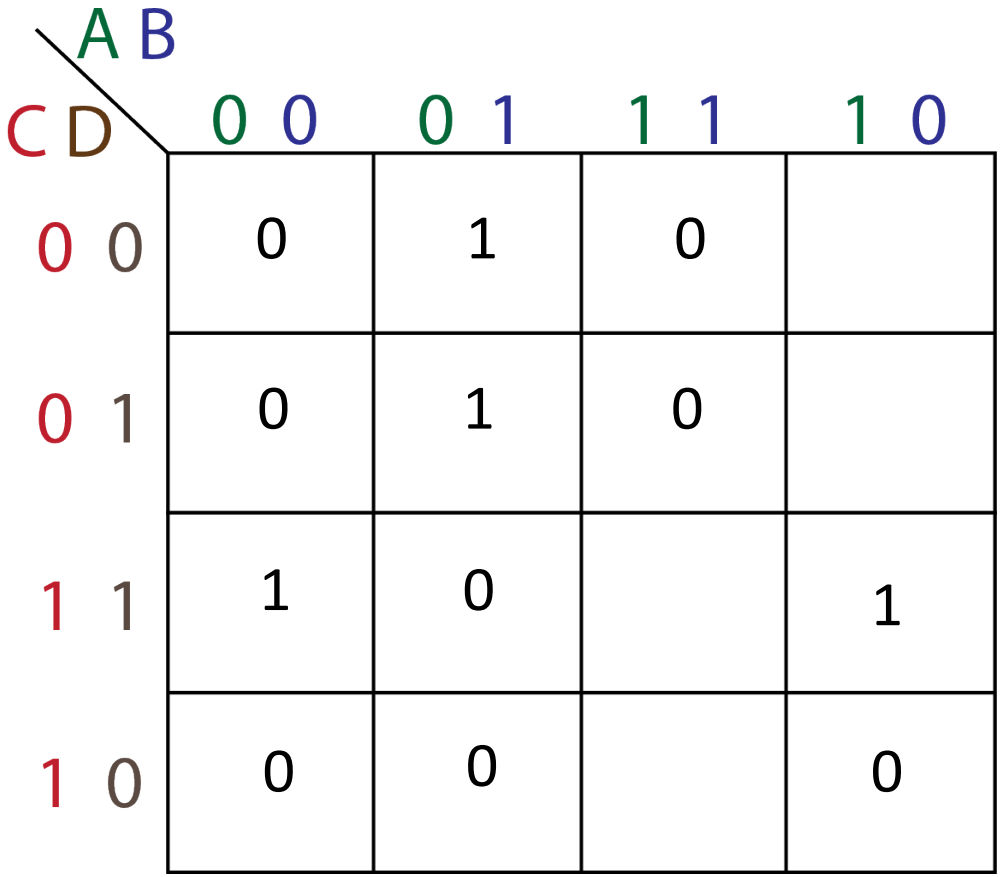
C D \ A B	0 0	0 1	1 1	1 0
0 0	0	0	0	?
0 1	0	0	0	?
1 1	0	1	?	1
1 0	0	1	?	1

K-Mapping: A State

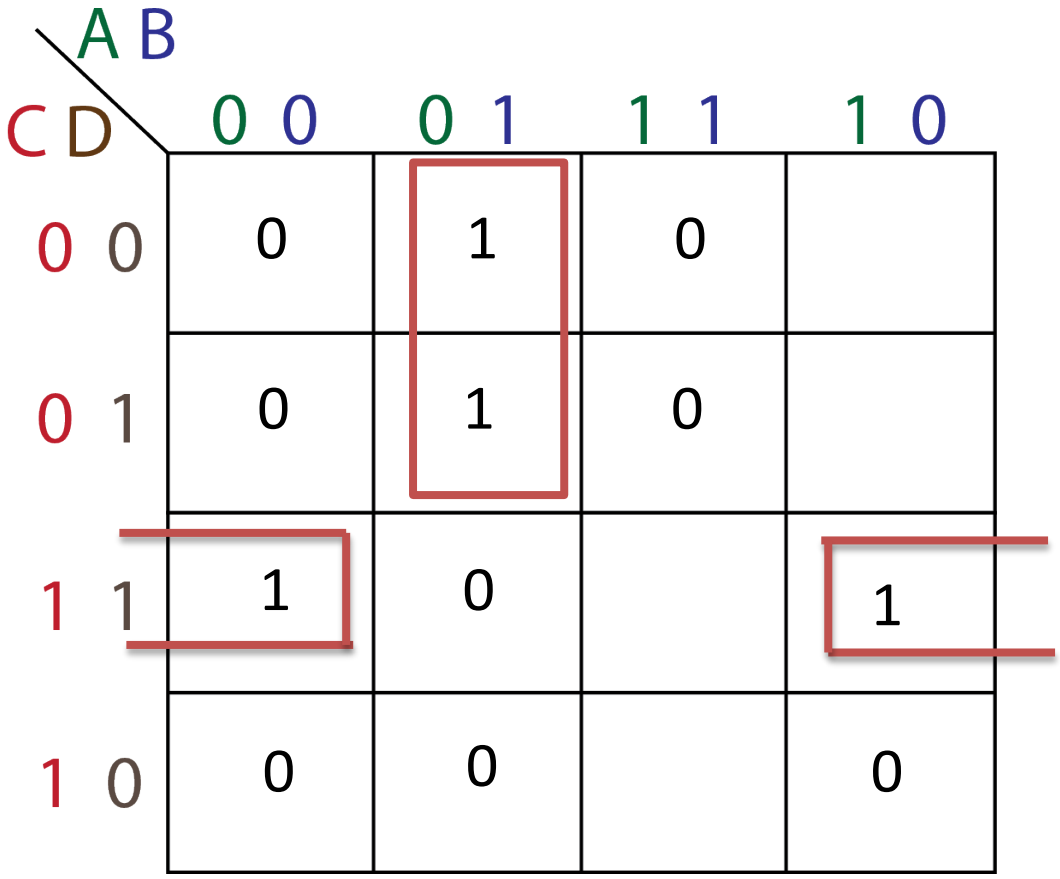


$$A^{+} = B \bullet C + A \bullet C$$

K-Mapping: B State

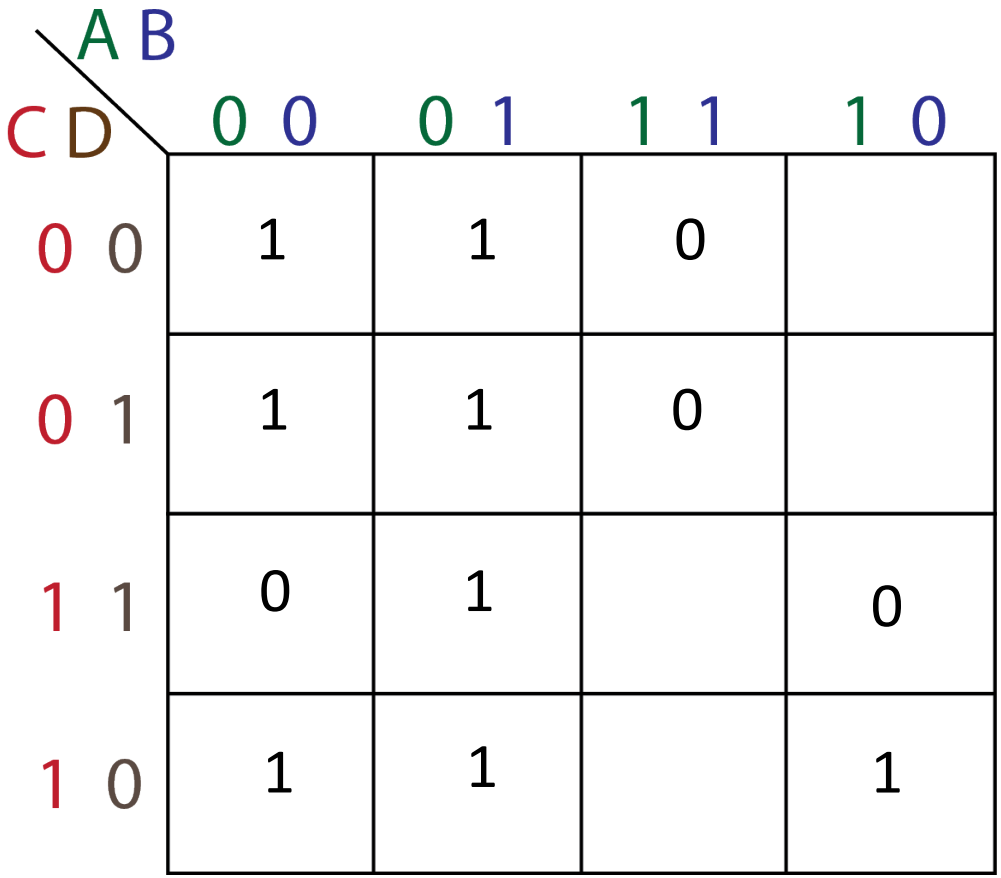


K-Mapping: B State



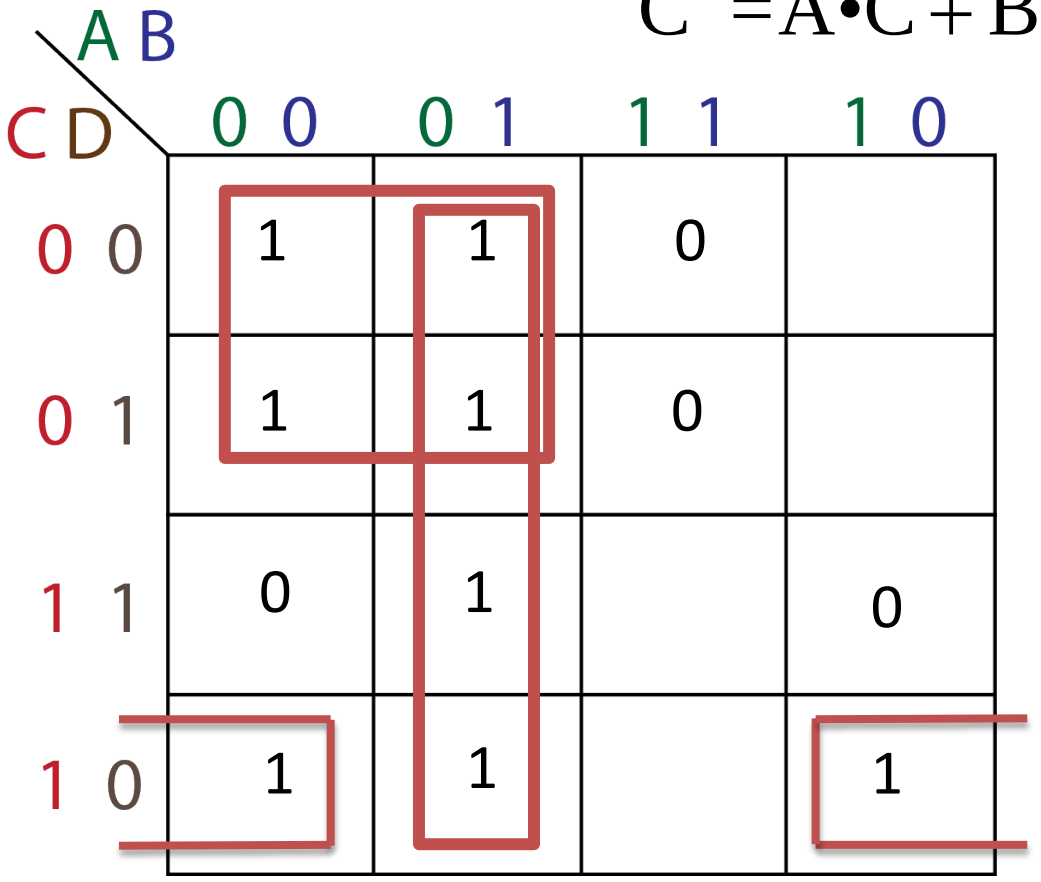
$$B^{+} = \overline{A} \bullet B \bullet \overline{C} + \overline{B} \bullet C \bullet D$$

K-Mapping: C State



K-Mapping: C State

$$C^{+} = \overline{A} \bullet \overline{C} + \overline{B} \bullet C \bullet \overline{D} + \overline{A} \bullet B$$



D Flip-Flop Implementation

$$A^{+} = \overline{B} \bullet C + A \bullet \overline{C}$$

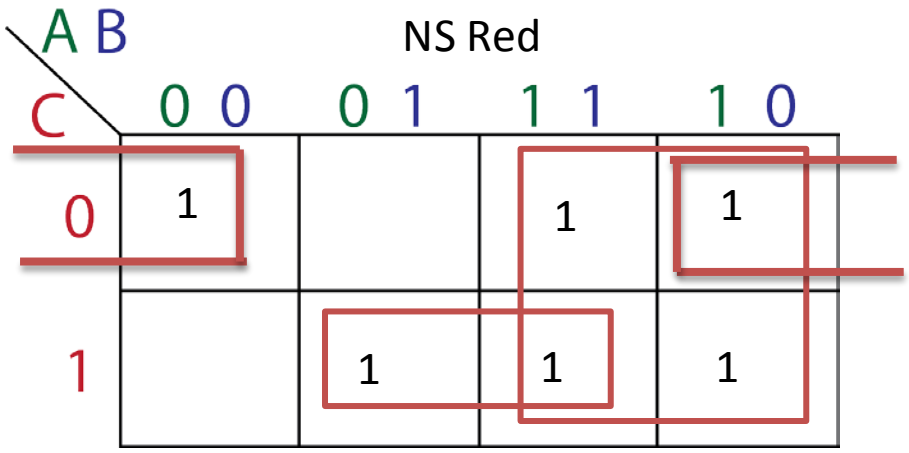
$$B^{+} = \overline{A} \bullet B \bullet \overline{C} + \overline{B} \bullet C \bullet D$$

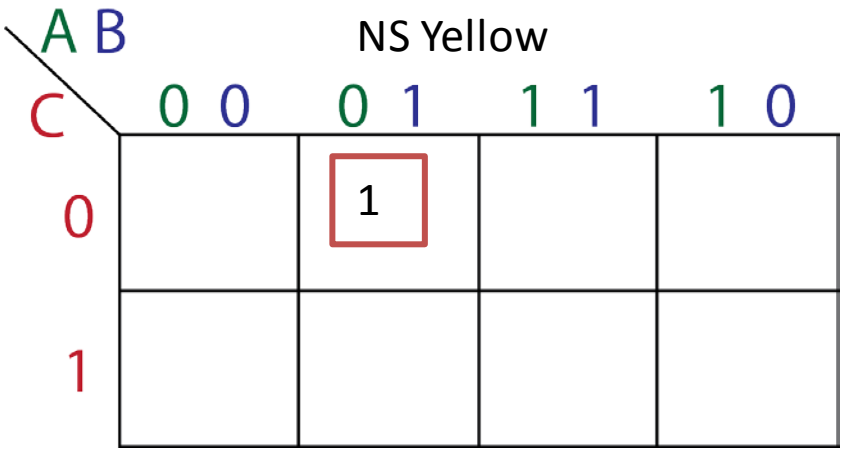
$$C^{+} = \overline{A} \bullet \overline{C} + \overline{B} \bullet C \bullet \overline{D} + \overline{A} \bullet B$$

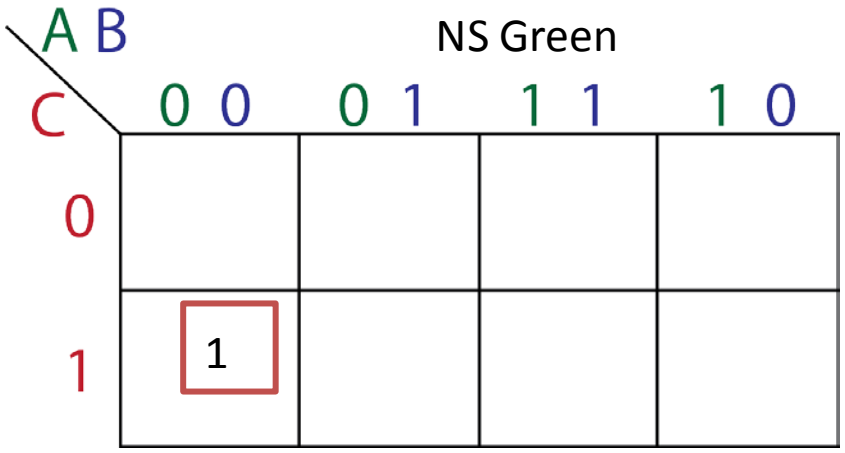
Outputs

A	B	C	Red	Yellow	Green	Red	Yellow	Green	Reset	Note
0	0	0	1	0	0	1	0	0	1	Red - Reset
0	0	1	0	0	1	1	0	0	0	Green NS
0	1	0	0	1	0	1	0	0	0	Yellow NS
0	1	1	1	0	0	1	0	0	1	Red - Reset
1	0	0	1	0	0	1	0	0	0	????
1	0	1	1	0	0	0	0	1	0	Green EW
1	1	0	1	0	0	0	1	0	0	Yellow EW
1	1	1	1	0	0	1	0	0	0	????

Output K-Maps





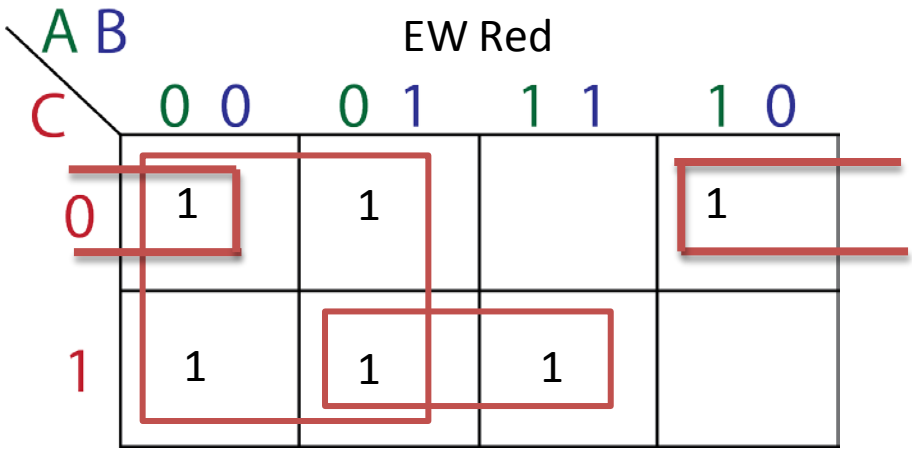


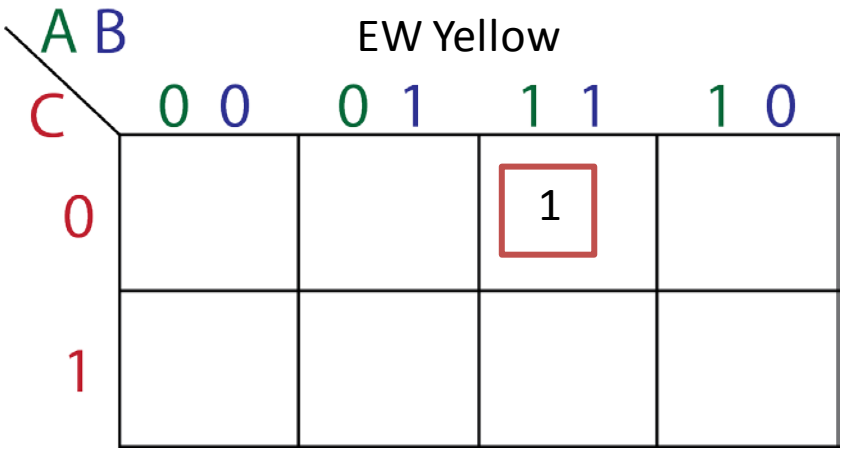
$$R_{NS} = A + B \bullet C + \overline{B} \bullet \overline{C}$$

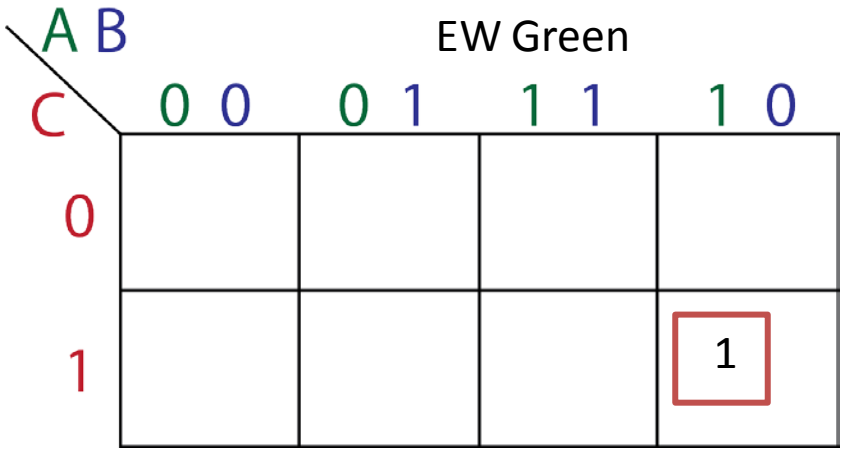
$$Y_{NS} = \overline{A} \bullet B \bullet \overline{C}$$

$$G_{NS} = \overline{A} \bullet \overline{B} \bullet C$$

Output K-Maps





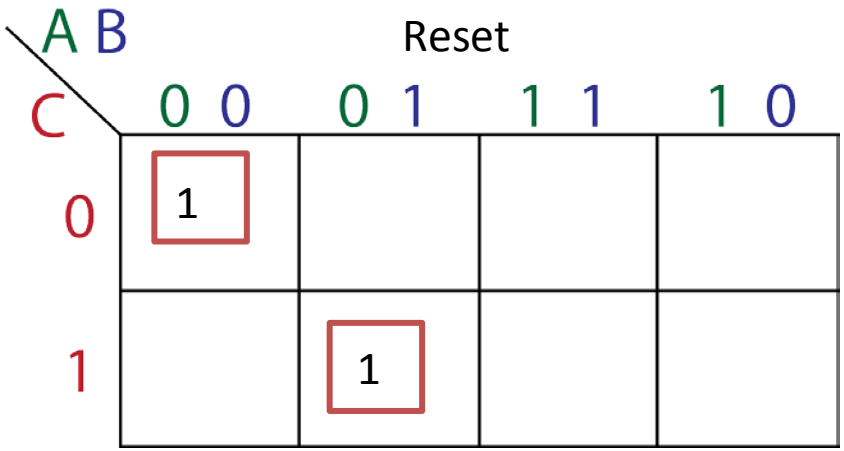


$$R_{EW} = \overline{A} + B \bullet C + \overline{B} \bullet \overline{C}$$

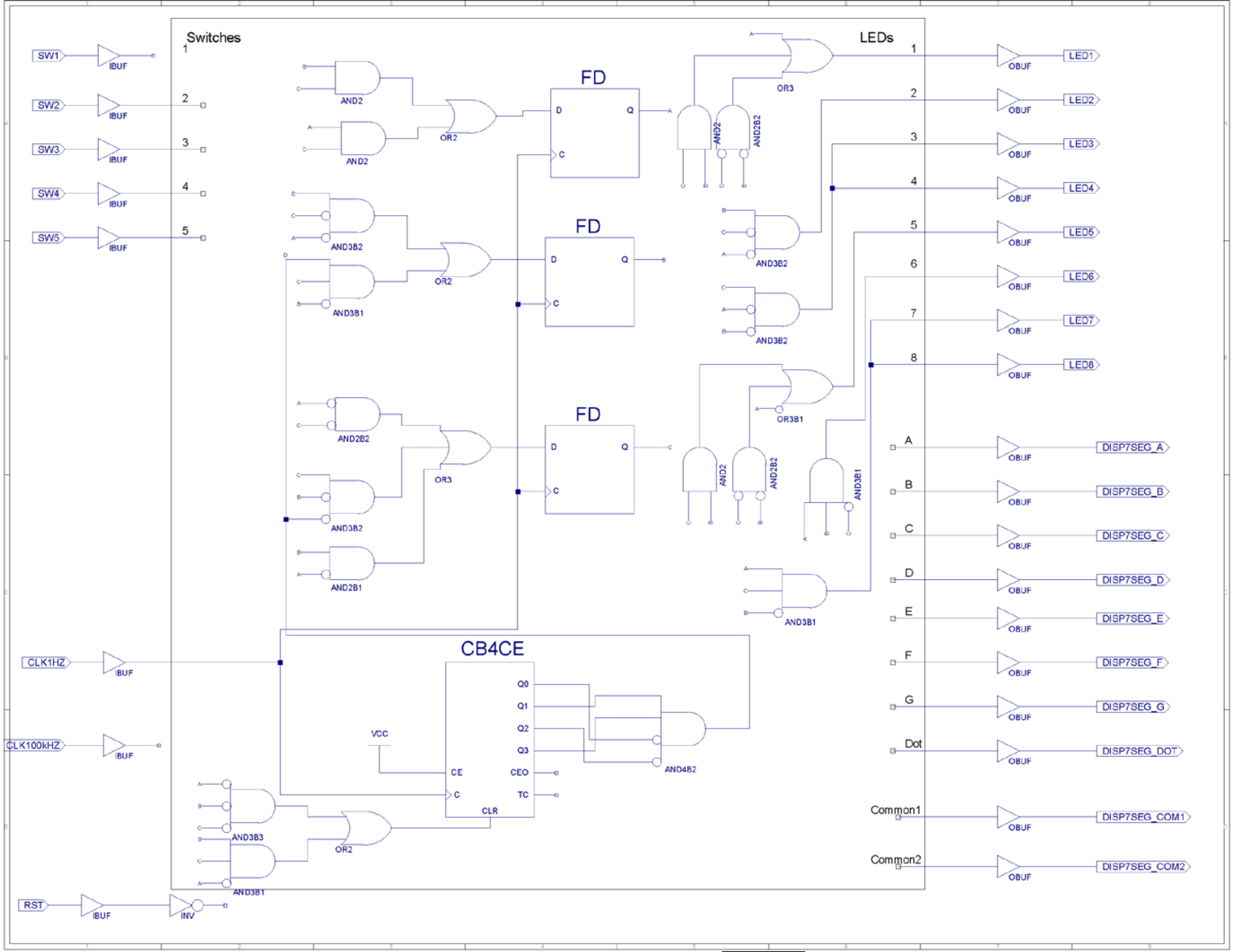
$$Y_{EW} = A \bullet B \bullet \overline{C}$$

$$G_{EW} = A \bullet \overline{B} \bullet C$$

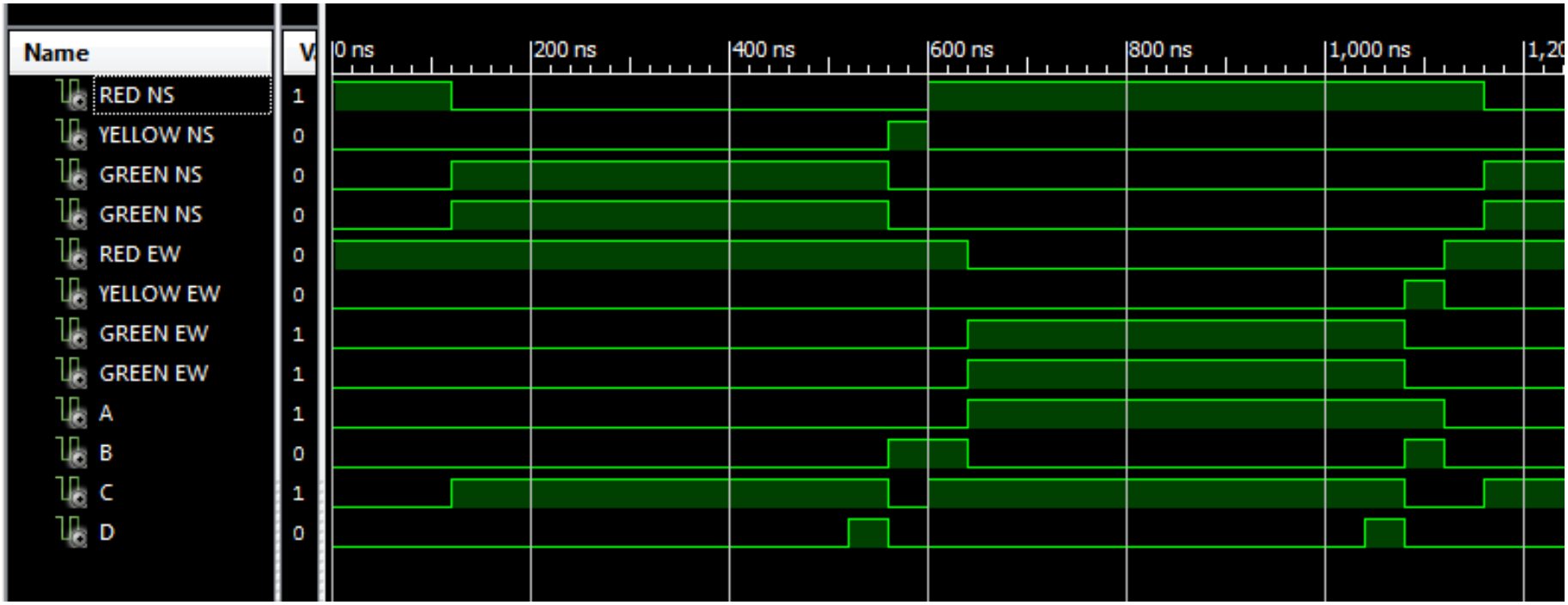
Output K-Maps



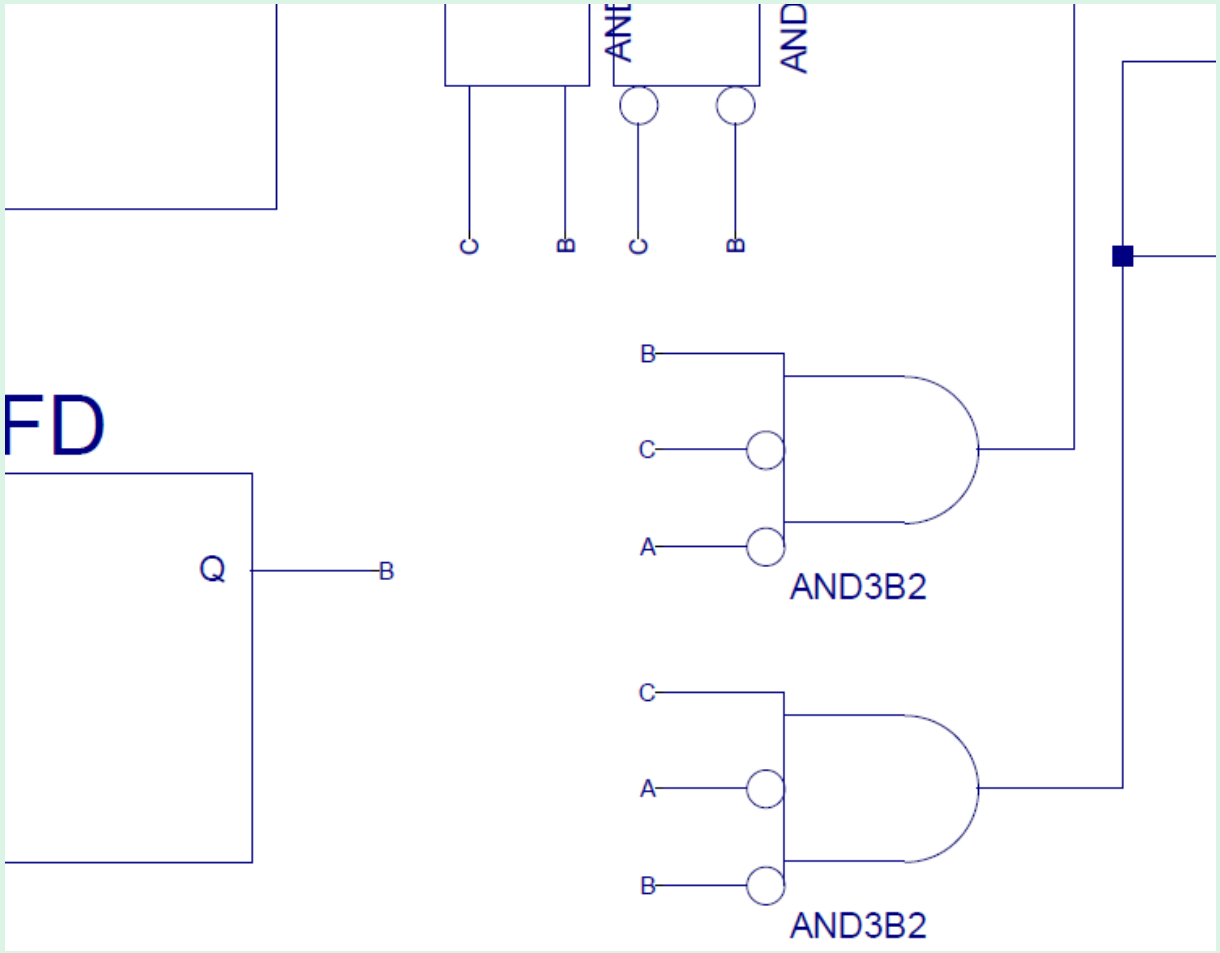
$$\text{Reset} = \overline{A} \bullet \overline{B} \bullet \overline{C} + \overline{A} \bullet B \bullet C$$


Switches
LEDs
SW1
SW2
SW3
SW4
SW5
IBUF
AND2
OR2
FD
D
Q
C
OR3
AND2B2
AND3B2
AND3B1
AND2B1
OBUF
LED1
LED2
LED3
LED4
LED5
LED6
LED7
LED8
OR3B1
AND3B1
A
B
C
D
E
F
G
Dot
DISP7SEG_A
DISP7SEG_B
DISP7SEG_C
DISP7SEG_D
DISP7SEG_E
DISP7SEG_F
DISP7SEG_G
DISP7SEG_DOT
Common1
Common2
DISP7SEG_COM1
DISP7SEG_COM2
CLK1HZ
CLK100kHZ
CB4CE
VCC
CE
CEO
TC
CLR
Q0
Q1
Q2
Q3
AND4B2
AND3B3
RST
INV

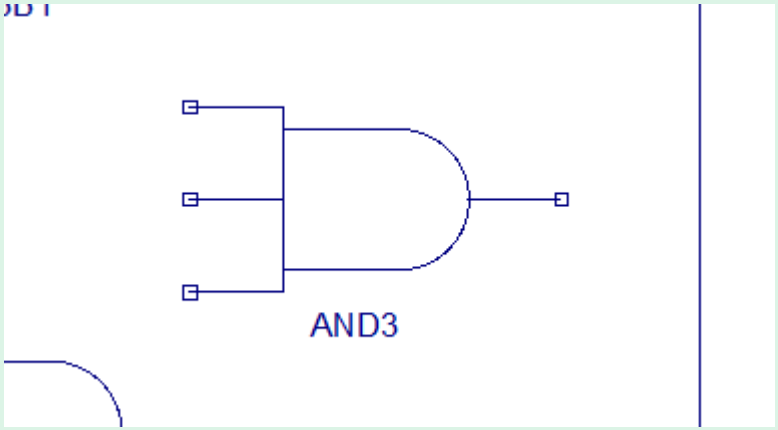
Simulation Results



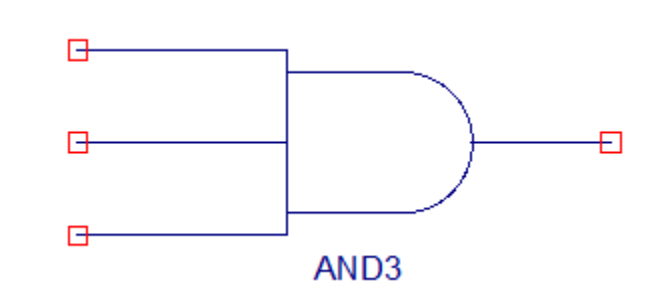
Sidenote on Schematic Notation



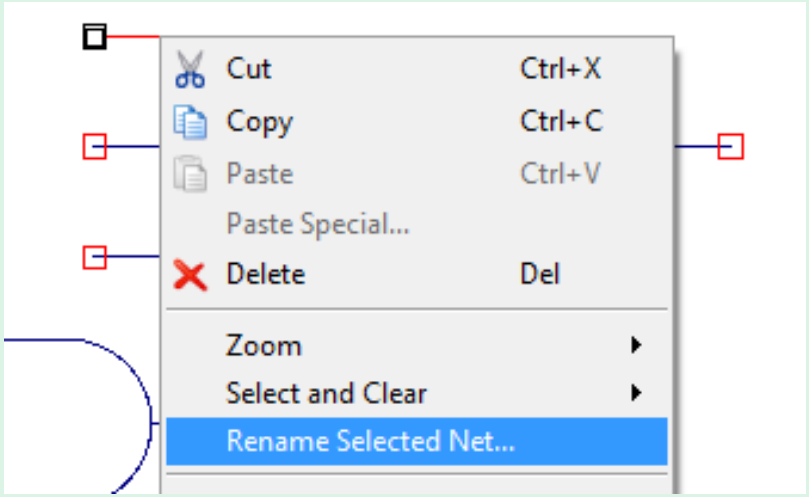
Creating with ISE: Step 1 Place Part



Creating with ISE: Step 2 Place SHORT wires

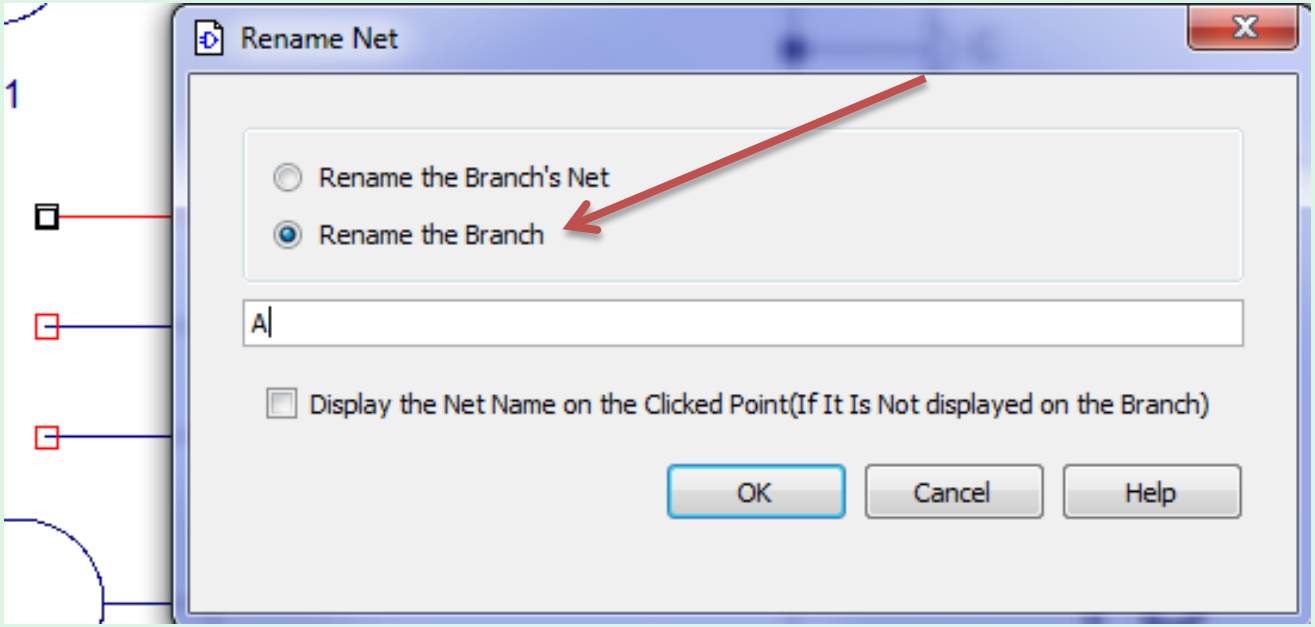


Creating with ISE: Step 3A: Rename Net



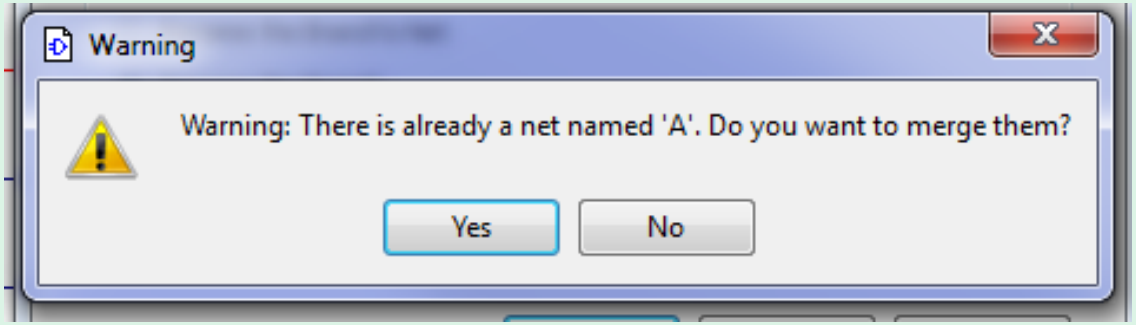
Right-click on net (must be careful not to select part or open port), you should get Rename Selected Net

Creating with ISE: Step 3B: Rename Net



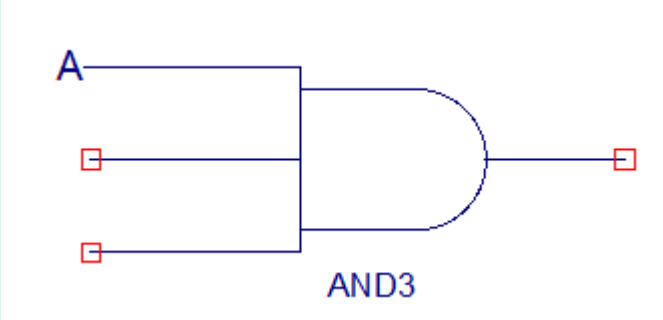
Put name in. Select 'Rename Branch' to change only this small section attached to the chip, which is what you want. Rename the Branch's net changes the name across ENTIRE schematic

Creating with ISE: Step 3c: Rename Net



This is telling you that your pin is going to be connected elsewhere. This is ok, hit Yes.

Creating with ISE: Step 4 - Done



Always use net names for larger schematics. It more easily allows you to change which input connects to a gate, since you can just rename that branch (Be SURE that 'Rename Branch' is selected or you change across entire schematic)

TYPES OF STATE MACHINE

State	Quarter	Loonie	Next State	Release
0.00	0	0	0.00	0
0.00	0	1	1.00	0
0.00	1	0	0.25	0
0.00	1	1	?	0
0.25	0	0	0.25	0
0.25	0	1	1.25	0
0.25	1	0	0.50	0
0.25	1	1	?	0
0.50	0	0	0.50	0
0.50	0	1	1.25	0
0.50	1	0	0.75	0
0.50	1	1	?	0
0.75	0	0	0.75	0
0.75	0	1	1.25	0
0.75	1	0	1.00	0
0.75	0	0	?	0
1.00	0	0	1.00	0
1.00	0	1	1.25	0
1.00	1	0	1.25	0
1.00	1	1	?	0
1.25	?	?	0.00	1

State	Quarter	Loonie	Next State	Release
0.00	0	0	0.00	0
0.00	0	1	1.00	0
0.00	1	0	0.25	0
0.00	1	1	?	0
0.25	0	0	0.25	0
0.25	0	1	0.00	1
0.25	1	0	0.50	0
0.25	1	1	?	0
0.50	0	0	0.50	0
0.50	0	1	0.00	1
0.50	1	0	0.75	0
0.50	1	1	?	0
0.75	0	0	0.75	0
0.75	0	1	0.00	1
0.75	1	0	1.00	0
0.75	0	0	?	0
1.00	0	0	1.00	0
1.00	0	1	0.00	1
1.00	1	0	0.00	1
1.00	1	1	?	0

Mealy Machine

Outputs dependant on state AND inputs

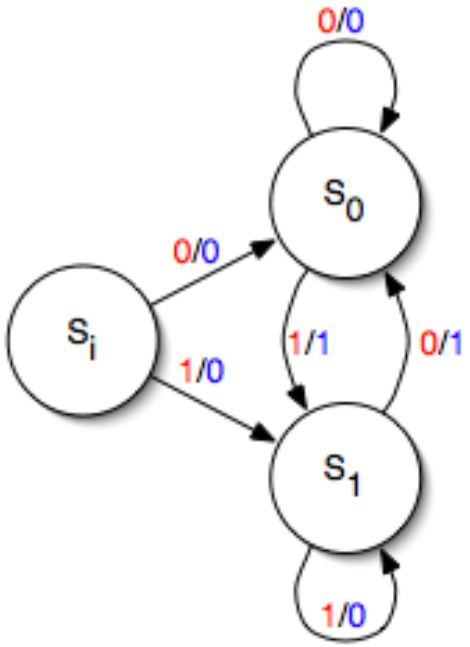


Image source: http://commons.wikimedia.org/wiki/File:Mealy.png

Moore Machine

Outputs dependant only on state

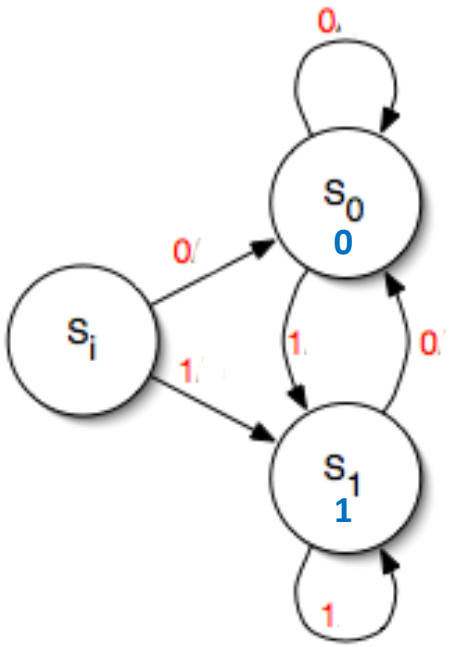


Image derived from: http://commons.wikimedia.org/wiki/File:Mealy.png

DESIGN EXAMPLE: MEALY MACHINE VS. MOORE MACHINE

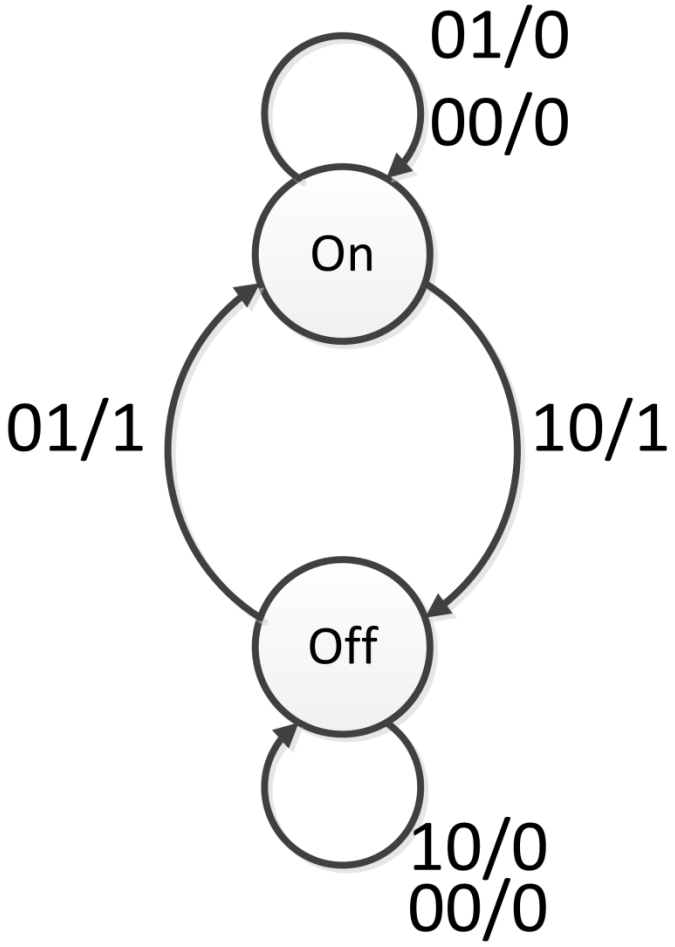
Description of Problem

I need a controller for a lamp with ON & OFF pushbuttons. The buttons are momentary only, so the controller needs state. A buzzer should beep whenever the state changes.

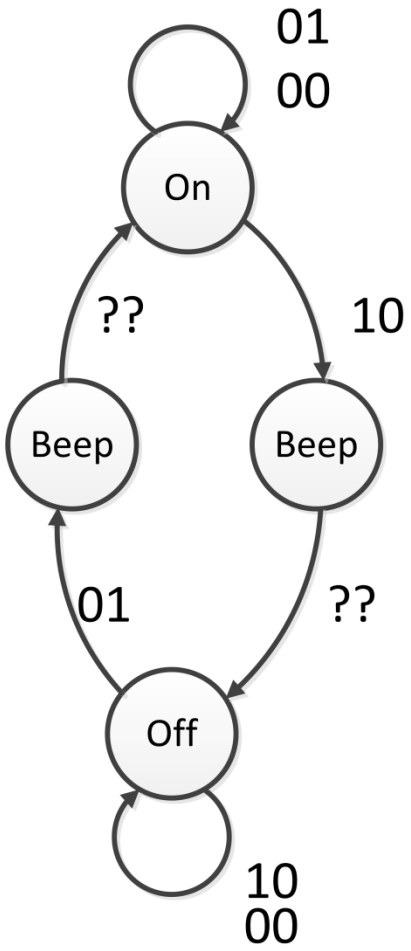
Step 1: Understand the Problem

Step 2: State Transition

Step 3: State Transition Diagram (Mealy)



Step 3: State Transition Diagram (Moore)



Step 4: State Transition Table (Mealy)

Initial State	On	Off	Next State	Lamp	Buzzer
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Step 4: State Transition Table (Mealy)

Initial State	On	Off	Next State	Lamp	Buzzer
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	1	0	1
0	1	1	0	0	0
1	0	0	1	1	0
1	0	1	0	1	1
1	1	0	1	1	0
1	1	1	1	1	0

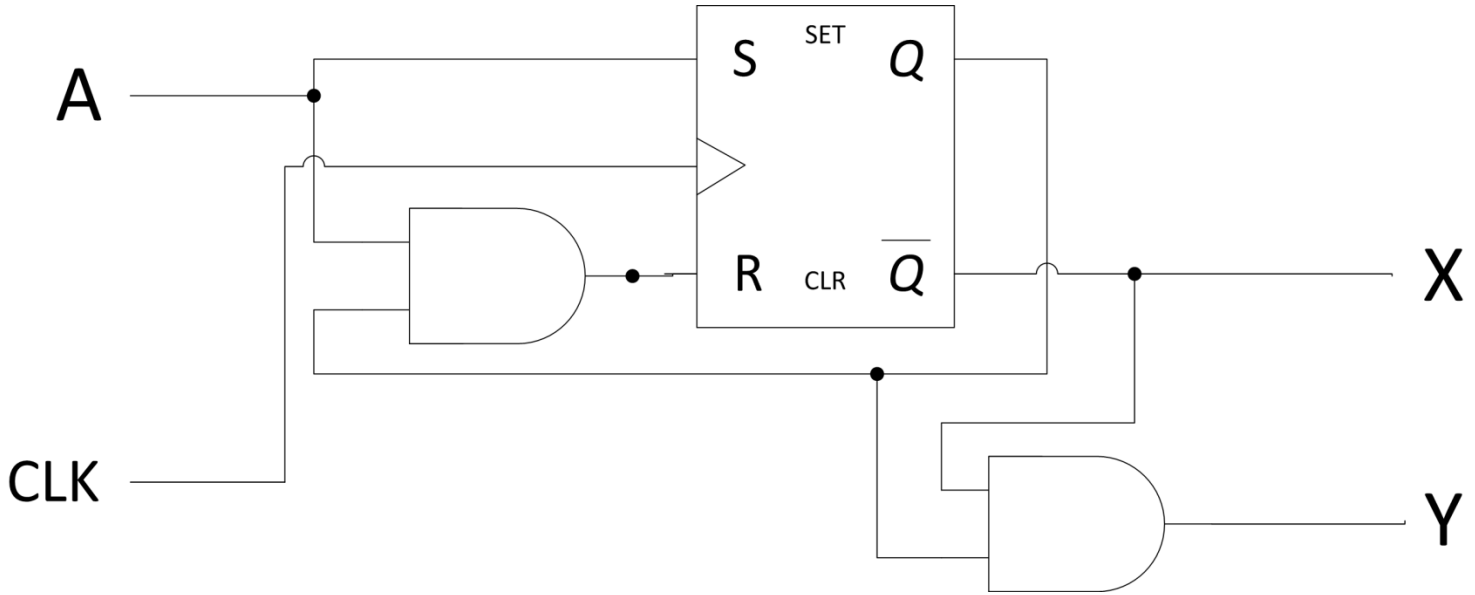
Step 4: State Transition Table (Moore)

Initial State	On	Off	Next State	Lamp	Buzzer
00	0	0			
00	0	1			
00	1	0			
00	1	1			
01	?	?			
10	0	0			
10	0	1			
10	1	0			
10	1	1			
11	?	?			

Step 4: State Transition Table (Moore)

Initial State	On	Off	Next State	Lamp	Buzzer
00	0	0	00	0	0
00	0	1	00	0	0
00	1	0	01	0	0
00	1	1	00	0	0
01	?	?	10	0	1
10	0	0	10	1	0
10	0	1	11	1	0
10	1	0	10	1	0
10	1	1	10	1	0
11	?	?	00	1	1

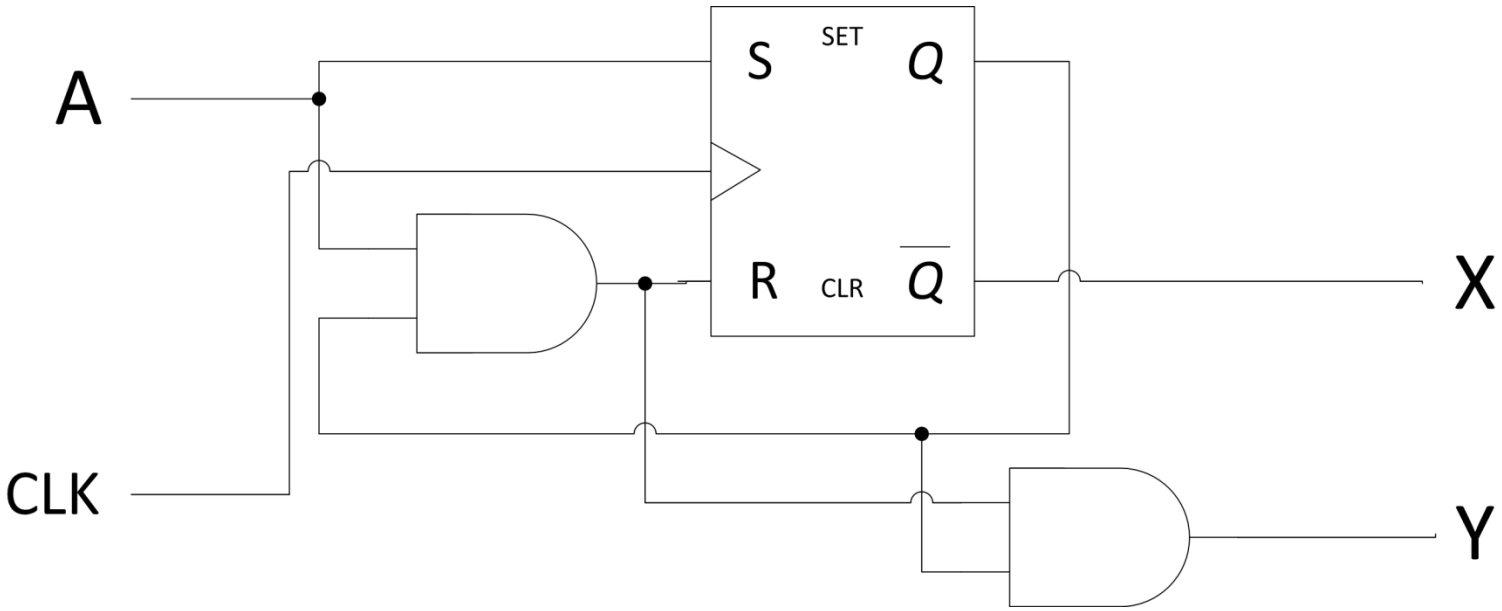
Note on Moore Machine



(Ignore the function of this, since it's nonsense!)

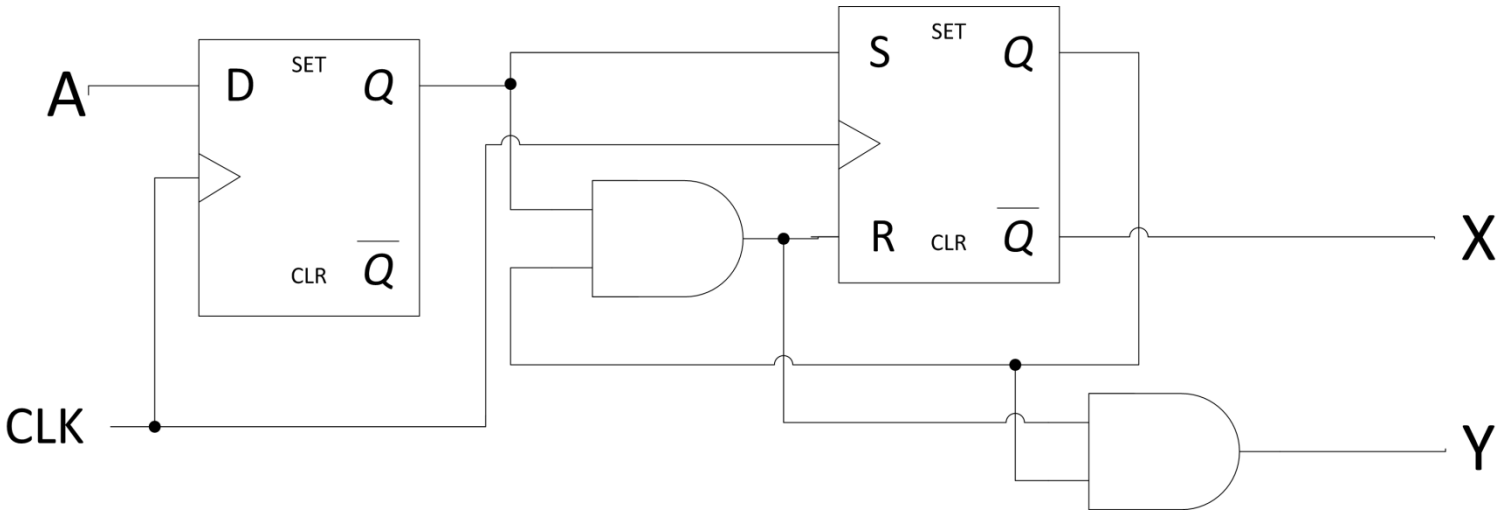
Note on Mealy Machine

Asynchronous Output



Note on Mealy Machine

Synchronous Output



CHOICE OF STATE ENCODING

Binary Encoding

Initial State	On	Off	Next State
00	0	0	00
00	0	1	00
00	1	0	01
00	1	1	00
01	?	?	10
10	0	0	10
10	0	1	11
10	1	0	10
10	1	1	10
11	?	?	00

Why Binary Encoding?

One-Hot Encoding

Initial State	On	Off	Next State
0001	0	0	0001
0001	0	1	0001
0001	1	0	0010
0001	1	1	0001
0010	?	?	0100
0100	0	0	0100
0100	0	1	1000
0100	1	0	0100
0100	1	1	0100
1000	?	?	0001

Why One-Hot Encoding?

Gray Coding

Initial State	On	Off	Next State
00	0	0	00
00	0	1	00
00	1	0	01
00	1	1	00
01	?	?	11
11	0	0	11
11	0	1	10
11	1	0	11
11	1	1	11
10	?	?	00

Why Gray Coding?

Section Summary

- See ECED2200 Notes
- Bebop to the Boolean Boogie Chapter 12