

1.

First, we can get the truth table as follows,

x	y	z	A	B	C
0	0	0	0	1	1
0	0	1	1	0	0
0	1	0	1	0	1
0	1	1	1	1	0
1	0	0	0	1	1
1	0	1	1	0	0
1	1	0	1	0	1
1	1	1	1	1	0

Then we can draw the K-maps one by one, first is for A.

		xy	00	01	11	10
		z	0	1	1	0
0	0	0	1	1	0	
	1	1	1	1	1	1

So we can get the minimized logic expression.

$$A = y + zy'$$

Second is for B.

		xy	00	01	11	10
		z	0	1	0	1
0	0	1	0	0	1	
	1	0	1	1	1	0

So we can get the minimized logic expression.

$$B = yz + y'z'$$

Third is for C.

		xy	00	01	11	10
		z	0	1	1	1
0	0	1	1	1	1	1
	1	0	0	0	0	0

So we can get the minimized logic expression.

$$C = z'$$

Then we can design our circuit, the result is shown below,

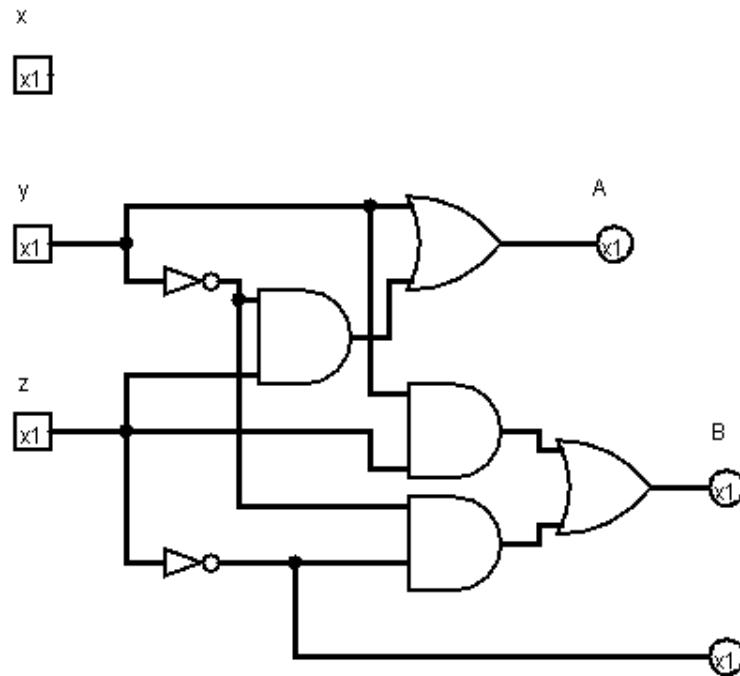


Figure 1: Combinational Circuit

2.

The solution is as follows,

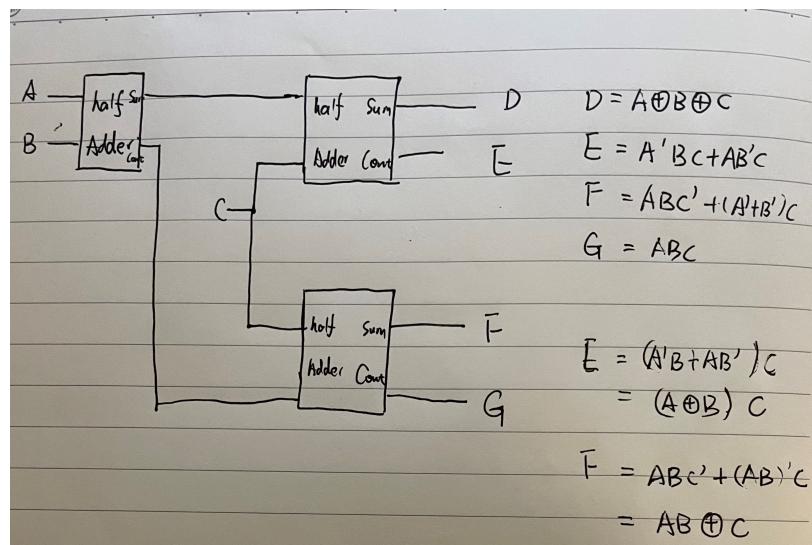


Figure 2: Result for question2

3.

We need truth table for this part, it just as follows,

x_2	x_1	x_0	y_2	y_1	y_0	$a)$	$b)$	$c)$
0	0	0	0	0	0	0	0	1
0	0	0	0	0	1	1	1	0
0	0	0	0	1	0	1	1	0
0	0	0	0	1	1	1	1	0
0	0	0	1	0	0	1	0	0
0	0	0	1	0	1	1	0	0
0	0	0	1	1	0	1	0	0
0	0	0	1	1	1	1	0	0
0	0	1	0	0	0	0	0	0
0	0	1	0	0	1	0	0	1
0	0	1	0	1	0	1	1	0
0	0	1	1	0	0	1	0	0
0	0	1	1	0	1	1	0	0
0	0	1	1	1	0	1	0	0
0	0	1	1	1	1	1	0	0
0	1	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0
0	1	0	0	1	0	0	0	1
0	1	0	0	1	1	1	1	0
0	1	0	1	0	0	1	0	0
0	1	0	1	1	1	1	0	0
0	1	1	0	0	0	0	0	0
0	1	1	0	1	1	0	0	1
0	1	1	1	0	0	1	0	0
0	1	1	1	1	1	1	0	0
1	0	0	0	0	0	0	1	0
1	0	0	0	0	1	0	1	0
1	0	0	0	1	0	0	1	0

x_2	x_1	x_0	y_2	y_1	y_0	a)	b)	c)
1	0	0	0	1	1	0	1	0
1	0	0	1	0	0	0	0	1
1	0	0	1	0	1	1	1	0
1	0	0	1	1	0	1	1	0
1	0	0	1	1	1	1	1	0
1	0	1	0	0	0	0	1	0
1	0	1	0	0	1	0	1	0
1	0	1	0	1	0	0	1	0
1	0	1	0	1	1	0	1	0
1	0	1	1	0	0	0	0	0
1	0	1	1	0	1	0	0	1
1	0	1	1	1	0	1	1	0
1	1	0	0	0	0	0	1	0
1	1	0	0	0	1	0	1	0
1	1	0	0	1	0	0	1	0
1	1	0	1	0	0	0	0	0
1	1	0	1	0	1	0	0	0
1	1	0	1	1	0	0	0	1
1	1	0	1	1	1	1	1	0
1	1	1	0	0	0	0	1	0
1	1	1	0	0	1	0	1	0
1	1	1	0	1	0	0	1	0
1	1	1	1	0	0	0	0	0
1	1	1	1	1	0	0	0	0
1	1	1	1	1	1	0	0	1

Then we can draw the K-map for a), b) and c).

a)

$x_2 \ x_1 \ x_0$	000	001	011	010	100	101	111	110
$y_2 \ y_1 \ y_0$	000	0	0	0	0	0	0	0
001	1	0	0	0	0	0	0	0
011	1	1	0	1	0	0	0	0
010	1	1	0	0	0	0	0	0
100	1	1	1	1	0	0	0	0
101	1	1	1	1	1	0	0	0
111	1	1	1	1	1	1	0	1
110	1	1	1	1	1	1	0	0

So we can get the logic function of A.

$$A = x_2' y_2 + x_1' y_2 y_1 + x_2' x_1' y_1 + x_2' x_0' y_1 y_0 + x_2' x_1' x_0' y_0 + x_1' x_0' y_2 y_0 + x_0' y_2 y_1 y_0$$

b)

$x_2 \ x_1 \ x_0$	000	001	011	010	100	101	111	110
$y_2 \ y_1 \ y_0$	000	0	0	0	1	1	1	1
001	1	0	0	0	1	1	1	1
011	1	1	0	1	1	1	1	1
010	1	1	0	0	1	1	1	1
100	0	0	0	0	0	0	0	0
101	0	0	0	0	1	0	0	0
111	0	0	0	0	1	1	0	1
110	0	0	0	0	1	1	0	0

So we can get the logic function of B.

$$B = x_2 y_2' + x_1' y_2' y_1 + x_1' x_0' y_2 y_0 + x_2 x_1' y_1 + x_2 x_0' y_1 y_0 + x_2 x_1' x_0' y_0 + x_0' y_2' y_1 y_0$$

c)

x_2	x_1	x_0	000	001	011	010	100	101	111	110
y_2	y_1	y_0								
000	1	0	0	0	0	0	0	0	0	0
001	0	1	0	0	0	0	0	0	0	0
011	0	0	1	0	0	0	0	0	0	0
010	0	0	0	1	0	0	0	0	0	0
100	0	0	0	0	1	0	0	0	0	0
101	0	0	0	0	0	1	0	0	0	0
111	0	0	0	0	0	0	1	0	0	0
110	0	0	0	0	0	0	0	0	1	0

So we can get the logic function of C.

$$C = x_2'x_1'x_0'y_2'y_1'y_0' + x_2'x_1'x_0y_2'y_1y_0 + x_2'x_1x_0y_2'y_1y_0 + x_2'x_1x_0'y_2'y_1'y_0' + \\ x_2x_1'x_0'y_2'y_1'y_0 + x_2x_1'x_0y_2y_1'y_0 + x_2x_1x_0y_2y_1y_0 + x_2x_1x_0'y_2y_1'y_0'$$

For these parts , if we want to extend it to larger numbers. C can be easily written as,

$$C = \prod_{i=0}^{i=n} (x_i + y_i)'$$

The other two parts are hard to write in functions.

But we can still extend it to large bits. For example, if we want to compare two 6-bits numbers, we can first compare higher 3 bits by our work. It should have 3 output a, b and c. Based on this, if a and b are true, then no matter what the lower 3 bits show, the output can be determined. If a and b are false and c is true, we need to compare the lower 3 bits again. If a, b, c are all false, the the result must be false.

4.

The result is as follows, I consider some gate delay.

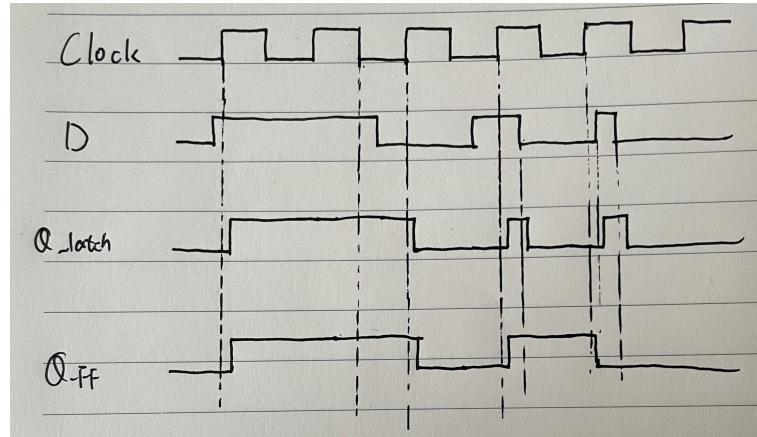


Figure 3: Q-latch and Q-Flipflop

5.

State	Remainder	Answer
Start	000000	000000
5	100000	000000
4	110000	000000
3	001000	001000
2	001100	001000
1	000100	001010
0	000000	001011

6.

Here is the Moore-type FSM for this part.

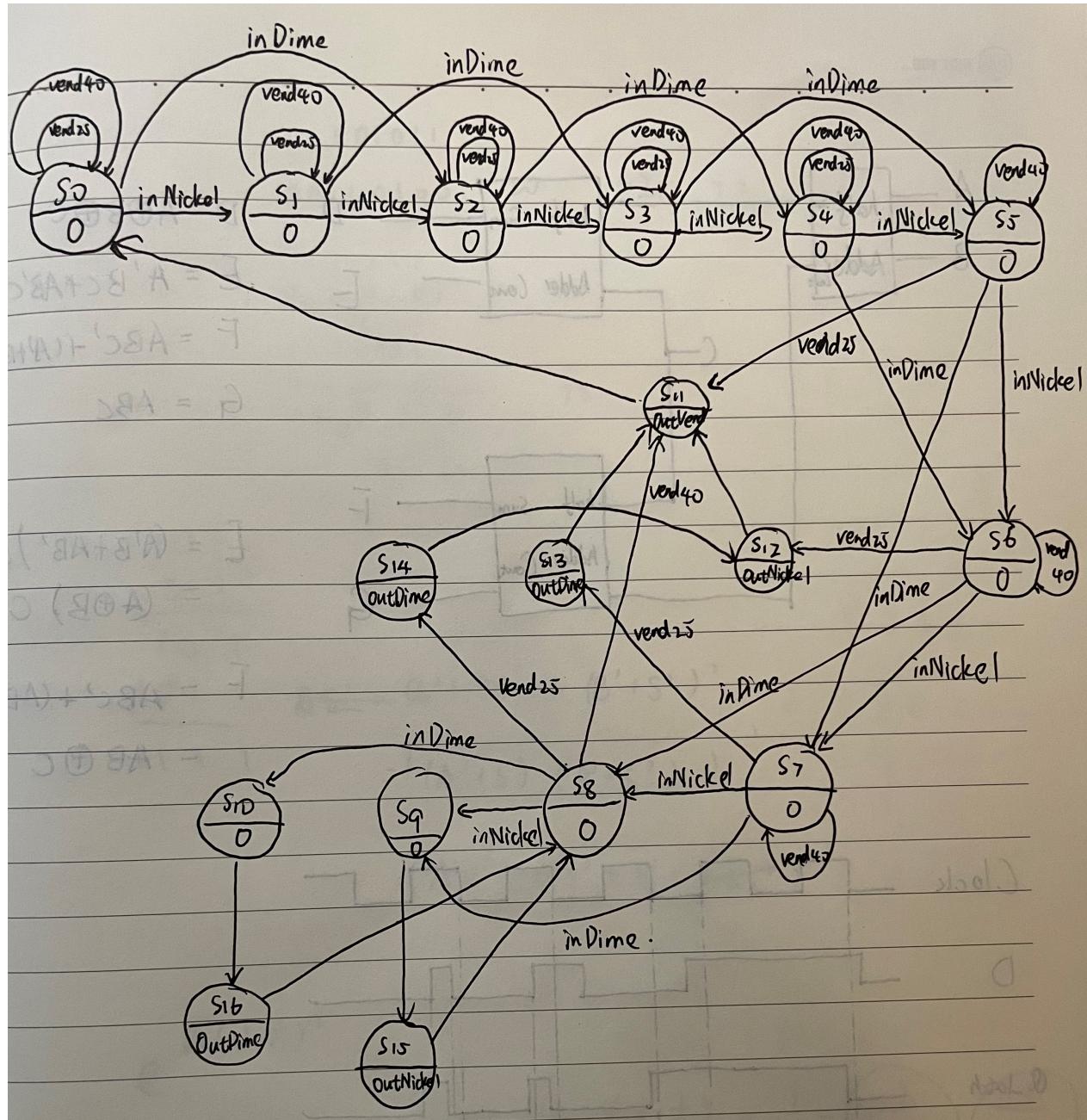


Figure 4: Moore-type FSM

Then I will briefly describe the state.

S0: The initial state, we have 0 cents here. No output.

S1: We have 5 cents here. No output.

S2: We have 10 cents here. No output.

S3: We have 15 cents here. No output.

S4: We have 20 cents here. No output.

S5: We have 25 cents here. No output.

S6: We have 30 cents here. No output.

S7: We have 35 cents here. No output.

S8: We have 40 cents here. No output.

S9: We have 45 cents here. No output.

S10: We have 50 cents here. No output.

S11: We have 0 cents left. Output is outVend. We need to dispense the requested snack. And then go to S1.

S12: We have 5 cents left. Output is outNickel. We need to give back a nickel to the user. And then go to S11.

S13: We have 10 cents left. Output is outDime. We need to give back a dime to the user. And then go to S11.

S14: We have 15 cents left. Output is outDime. We need to give back a dime to the user. And then go to S12.

S15: We have 45 cents here. Output is outNickel. We need to give back a nickel to the user. And then go to S8.

S16: We have 50 cents here. Output is outDime. We need to give back a dime to the user. And then go to S8.