Digital Logic Assignment 4

12111448 侯芳旻

Question 1

- (10 points) Implement a full adder with two 4 × 1 multiplexers. Draw the logic diagram.

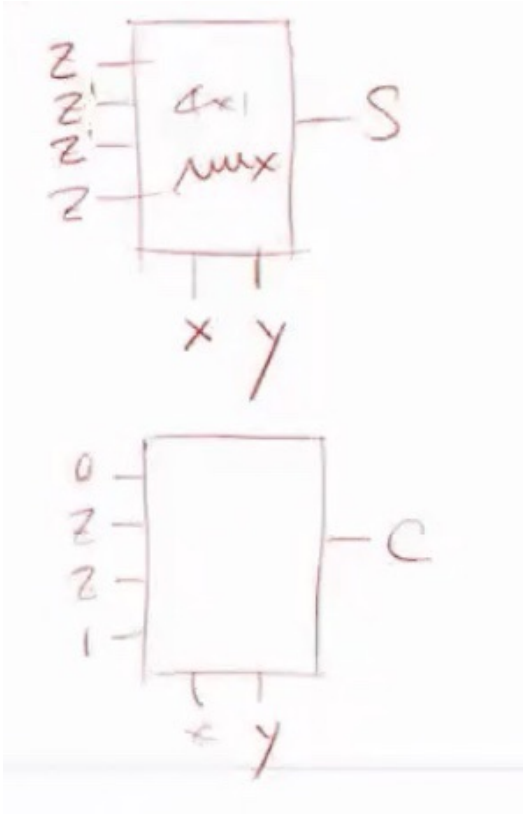
A full adder has 3 input x, y, z and 2 output carry c, sum s.

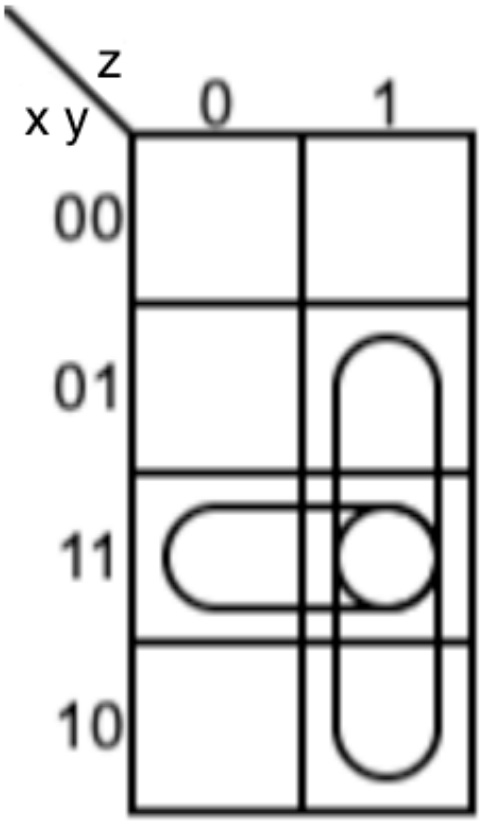
just use 2 mux is possible.
As the input of MUX can be NOT just 0/1. it can be a variable

Then we can create the truth table of full adder:

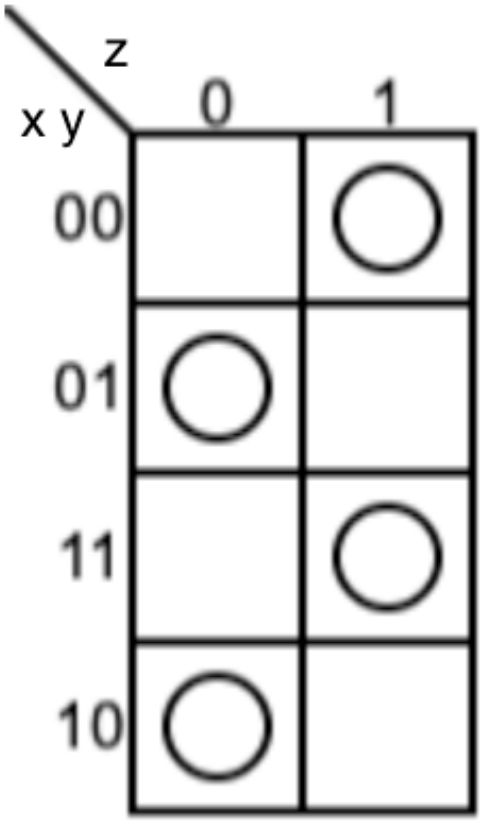
x	y	z	c	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

then draw the k-map of c and s:



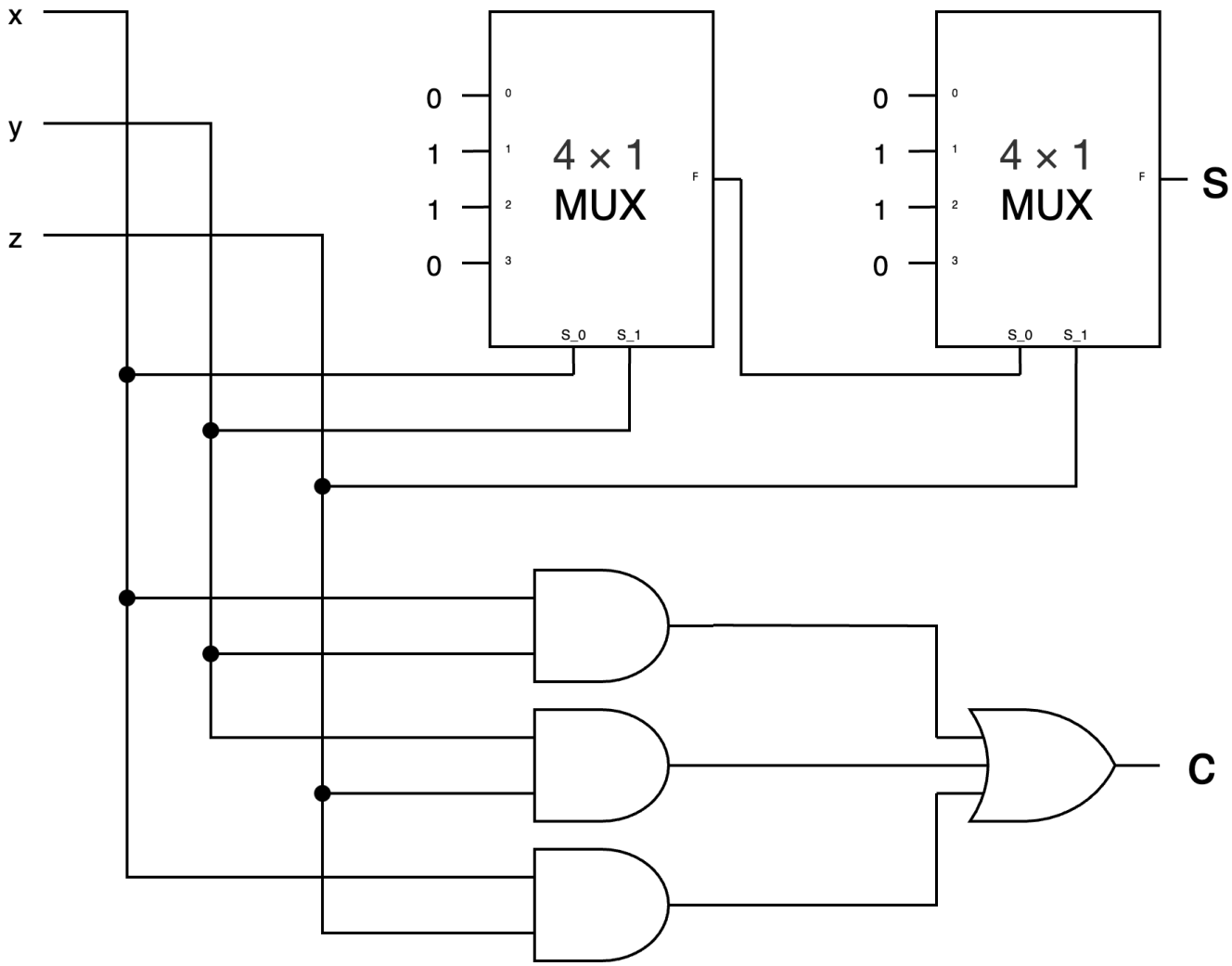


$$c = xy + yz + xz \tag{28}$$



$$s = xy'z' + x'yz' + x'y'z + xyz \tag{29}$$

then draw the diagram:



Question 2

- (10 points) Design a sequence generator to generate the sequence 110101. Draw the logic diagram.

From the formula:

$$N \leq 2^n - 1 \tag{30}$$

the length of the sequence

For the sequence 110101, $N = 6$, then

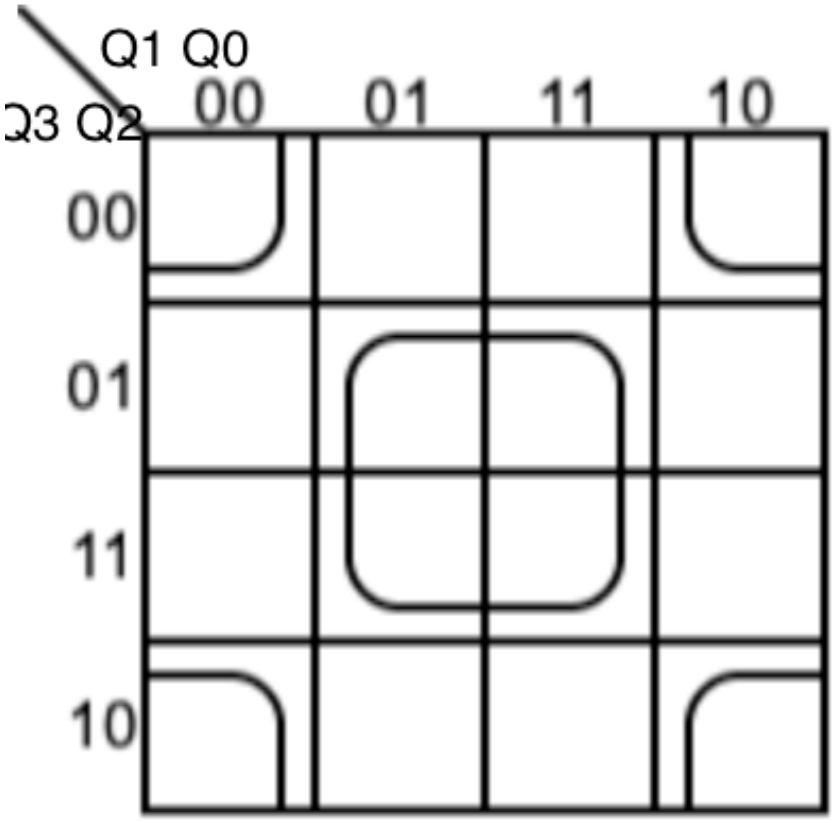
$$n_{min} = 3 \tag{31}$$

clk	Q_2	Q_1	Q_0
↑	1	1	0
↑	1	1	1
↑	0	1	1
↑	1	0	1
↑	0	1	0
↑	1	0	1

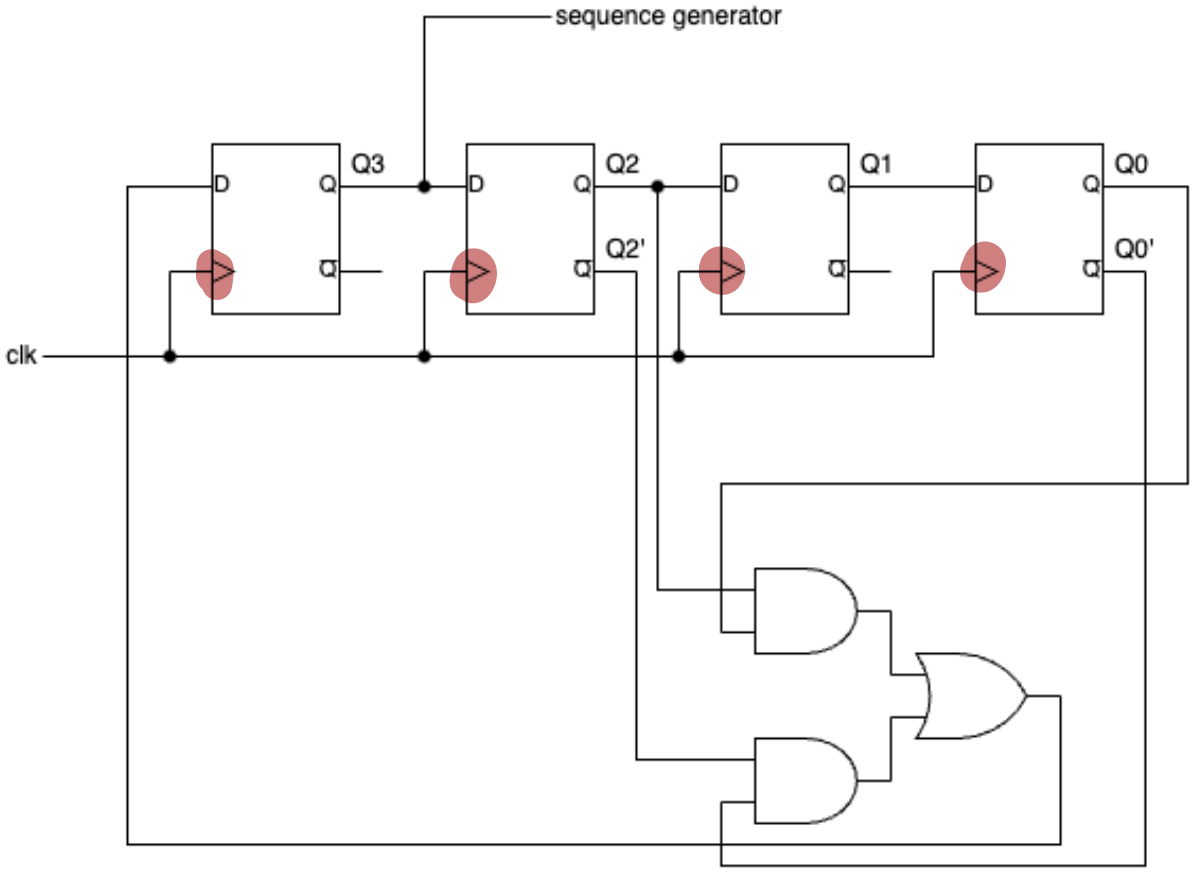
As 101 occurs twice, $n = 3$ is not sufficient. So let $n = 4$:

clk	Q_3	Q_2	Q_1	Q_0	Z
↑	1	1	0	1	1
↑	1	1	1	0	0
↑	0	1	1	1	1
↑	1	0	1	1	0
↑	0	1	0	1	1
↑	1	0	1	0	1

Then draw the Kmap



$$Z = Q_2' Q_0' + Q_2 Q_0 \tag{32}$$



Question 3

- (10 points) Design a combinational circuit that generates the 9's complement of a BCD digit. Invalid codes are don't-care conditions. Draw the logic diagram.

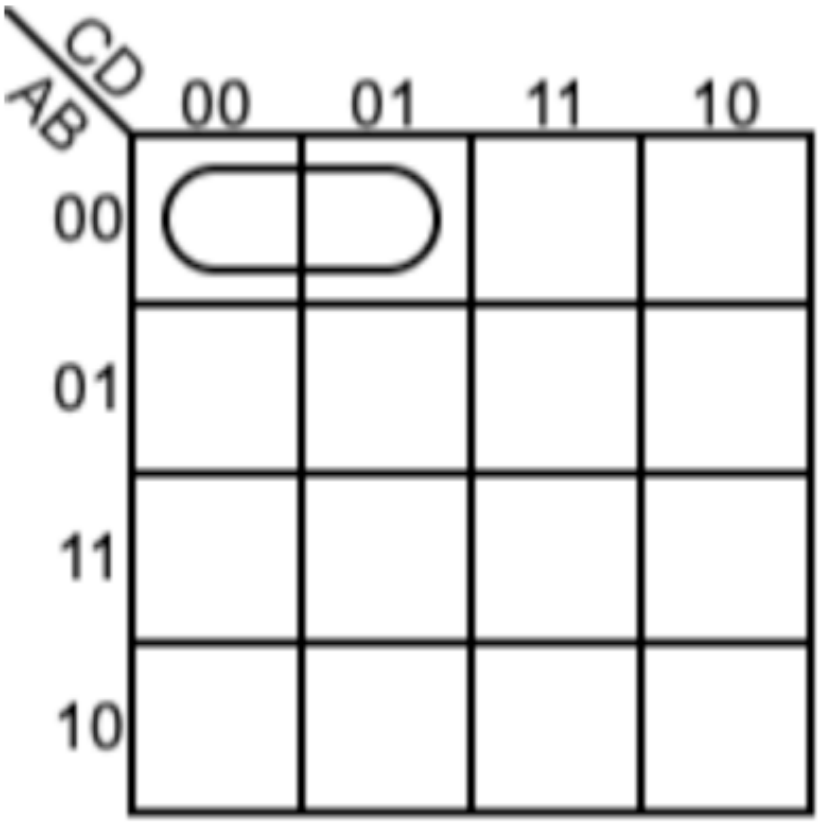
For BCD digit, let ABCD represent the 4 digits of inputs from MSB to LSB and EFGH be the outputs digits from MSB to LSB.

recommended to use W.X.Y.Z to represent the output and A.B.C.D to represent the input.

As the 9's complement $= 1001 - ABCD$, the truth table is :

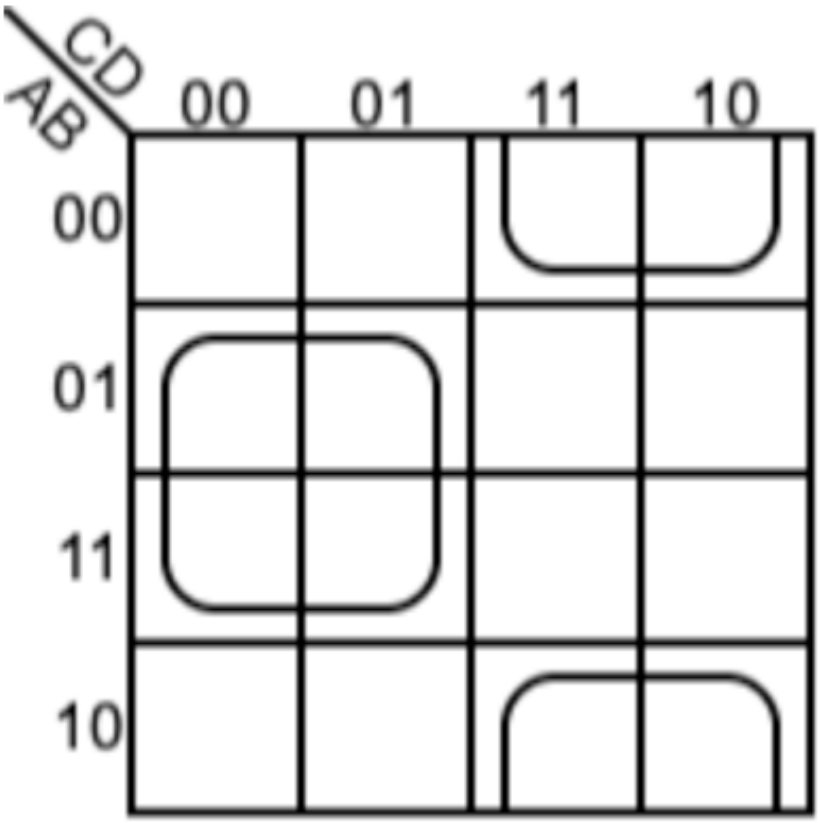
A	B	C	D	E	F	G	H
0	0	0	0	1	0	0	1
0	0	0	1	1	0	0	0
0	0	1	0	0	1	1	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	0	0
0	1	1	0	0	0	1	1
0	1	1	1	0	0	1	0
1	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0

then the Kmap of E:



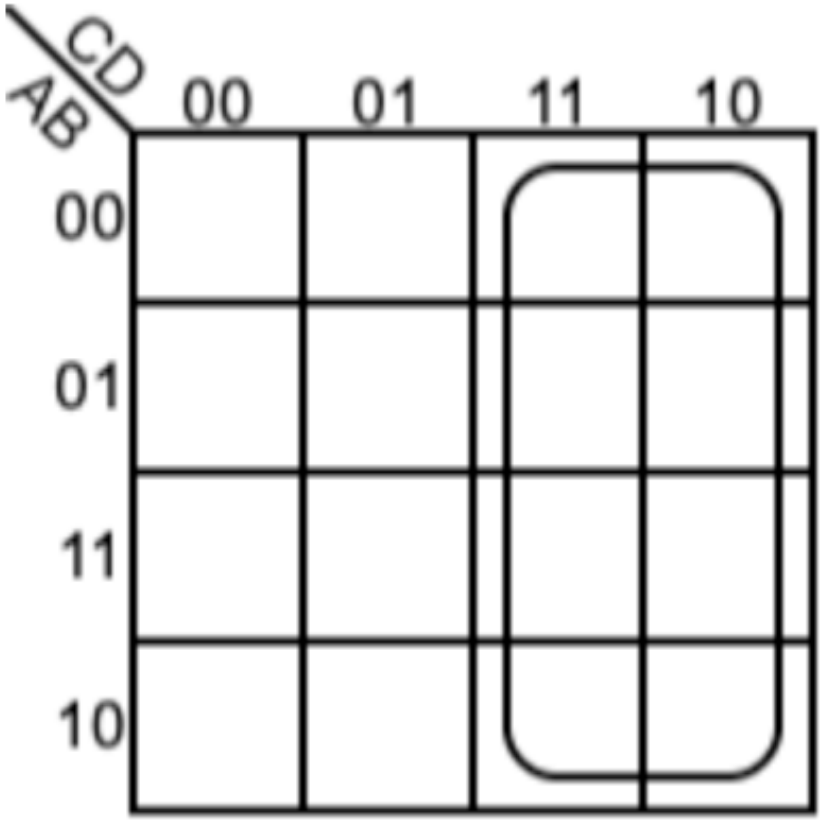
$$E = A'B'C' \tag{33}$$

the Kmap of F:



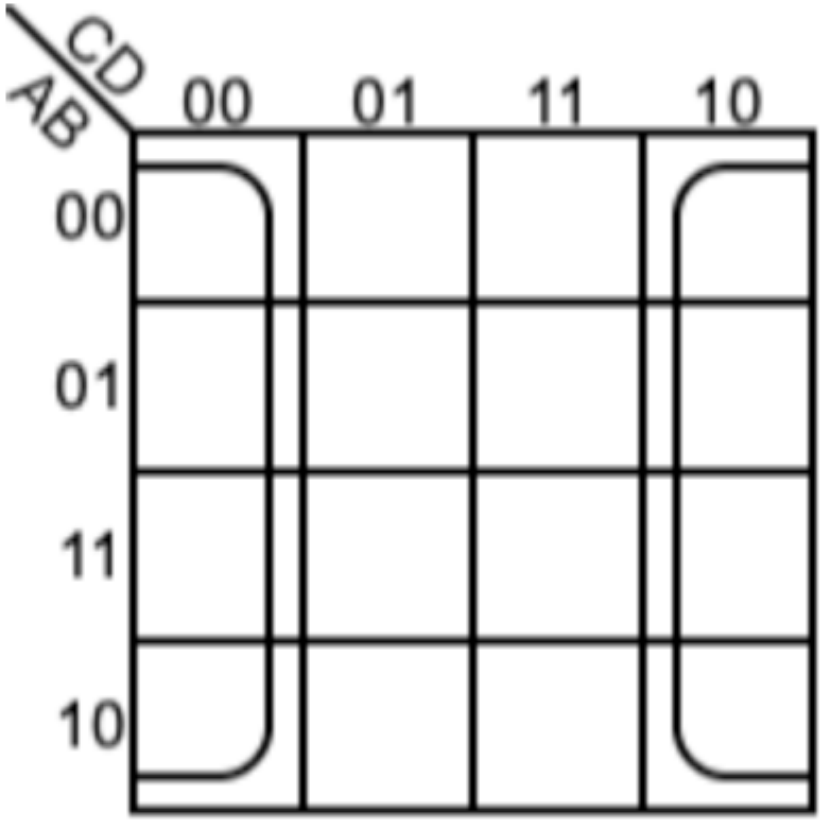
$$F = BC' + B'C \tag{34}$$

the Kmap of G:



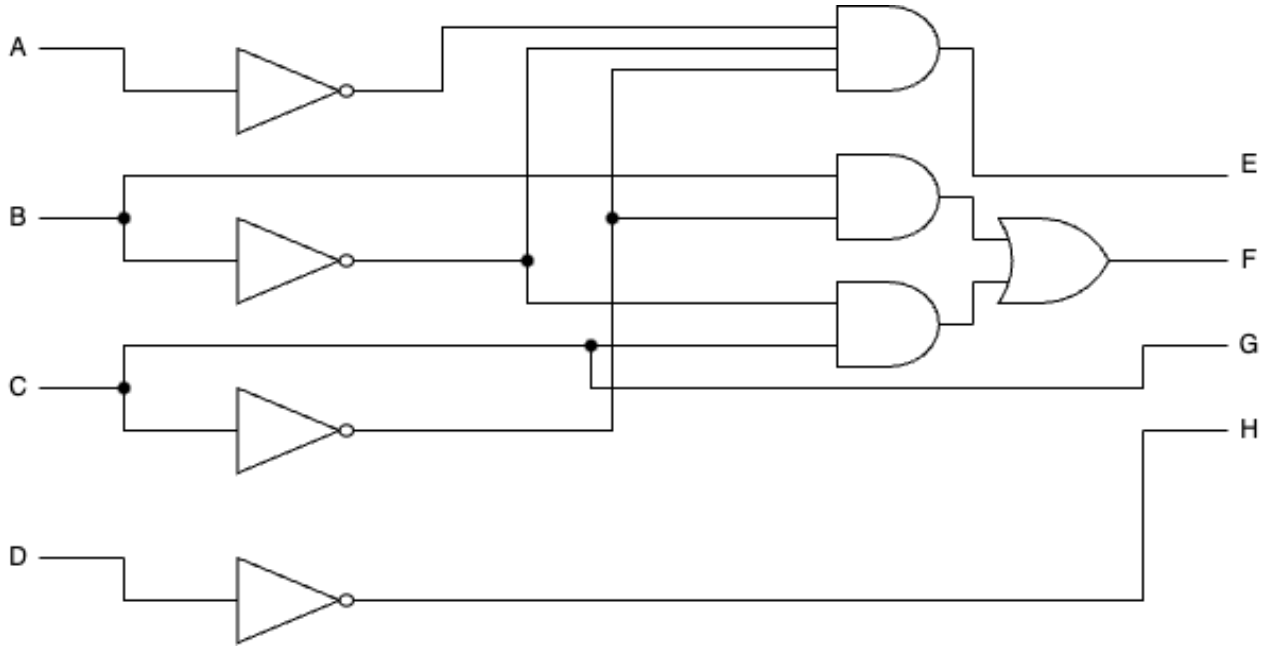
$$G = C \tag{35}$$

the Kmap of H:



$$H = D' \tag{36}$$

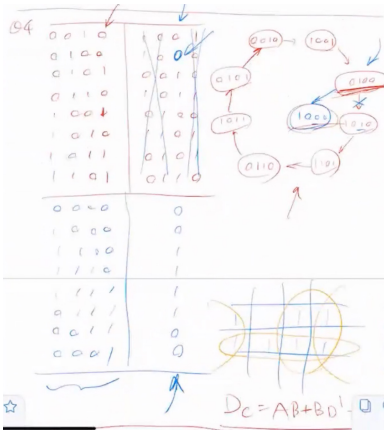
then draw the diagram:



Question 4

- (20 points) List the eight unused states in a 4-bit Johnson's counter. Determine the next state for each unused state and show that, if the circuit finds itself in an invalid state, it does not return to a valid state. Then modify the circuit so that the circuit reaches a valid state from any one of the unused states. Draw the logic diagram.

1. unused state:
    1. $0010 \rightarrow 1001$
    2. $0100 \rightarrow 1010$
    3. $0101 \rightarrow 0010$
    4. $0110 \rightarrow 1011$
    5. $1001 \rightarrow 0100$
    6. $1010 \rightarrow 1101$
    7. $1011 \rightarrow 0101$
    8. $1101 \rightarrow 0110$

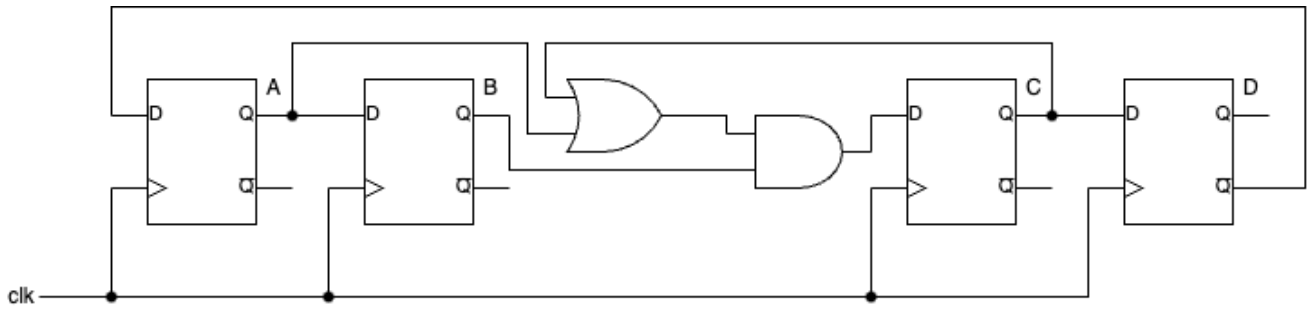
    From aboove, we can find that the next state of each invalid state is another different invalid state, and it is clear to see that the no matter how many times it goes through, it will always return an in valid type. So if the circuit get into an invalid state, it can NOT return to valid state.

2. To modify it, disconnect the output B of the B flip-flop with the D input D_c of flip-flop C. Instead, let

$$D_c = (A + C)B \tag{37}$$

then draw the diagram:

to modify the circuit, we just need to choose one invalid state, and let the next state of the invalid state to be valid. and only revise one bit is easier to revise. you just need to add a combinational circuit before the input of a ff. and the input of combinational circuit is the states of all ff.



Question 5

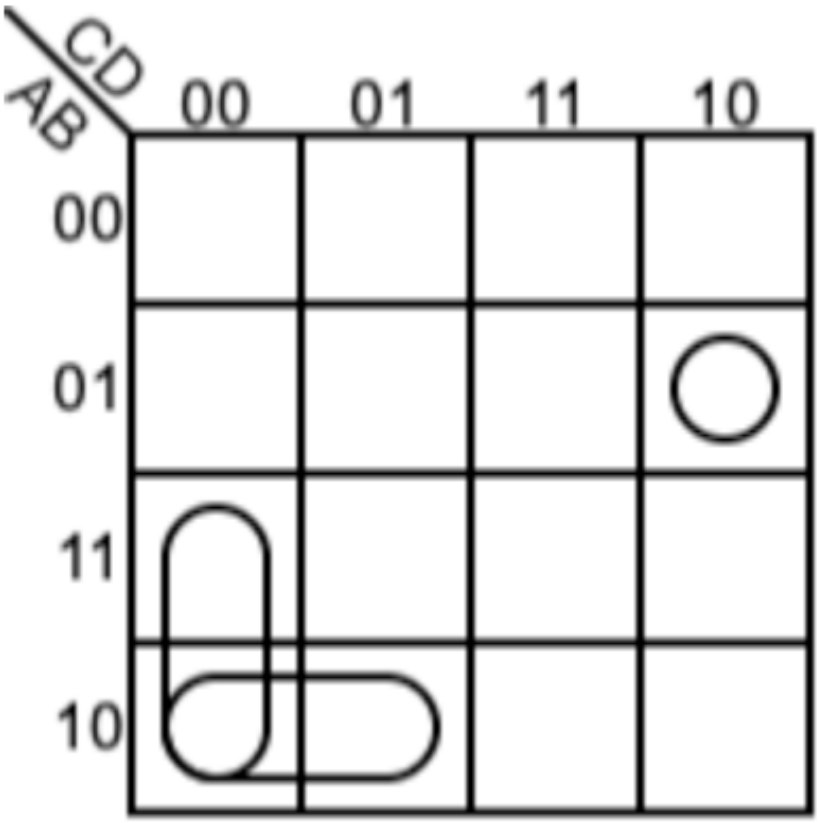
- (20 points) Design a synchronous counter that has the following sequence: 0010, 0110, 1001, 1000, 1100, 1101, and repeat. From the undesired states the counter must always go to 0010 on the next clock pulse. Draw the logic diagram.

Let A, B, C, D represent the output of the flip-flops. Then draw the state diagram:

$D_C = D_A'$

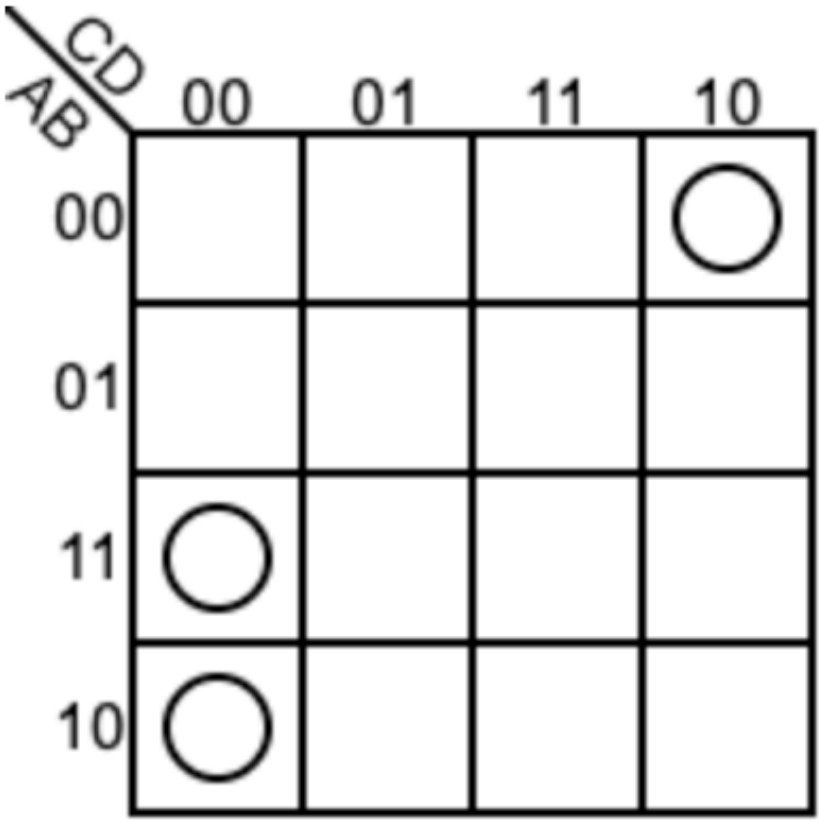
A(t)	B(t)	C(t)	D(t)	A(t+1)	B(t+1)	C(t+1)	D(t+1)	D_A	D_B	D_C	D_D
0	0	0	0	0	0	1	0	0	0	1	0
0	0	0	1	0	0	1	0	0	0	1	0
0	0	1	0	0	1	1	0	0	1	1	0
0	0	1	1	0	0	1	0	0	0	1	0
0	1	0	0	0	0	1	0	0	0	1	0
0	1	0	1	0	0	1	0	0	0	1	0
0	1	1	0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	0	0	0	1	0
1	0	0	0	1	1	0	0	1	1	0	0
1	0	0	1	1	0	0	0	1	0	0	0
1	0	1	0	0	0	1	0	0	0	1	0
1	0	1	1	0	0	1	0	0	0	1	0
1	1	0	0	1	1	0	1	1	1	0	1
1	1	0	1	0	0	1	0	0	0	1	0
1	1	1	0	0	0	1	0	0	0	1	0
1	1	1	1	0	0	1	0	0	0	1	0

Kmap of D_A:



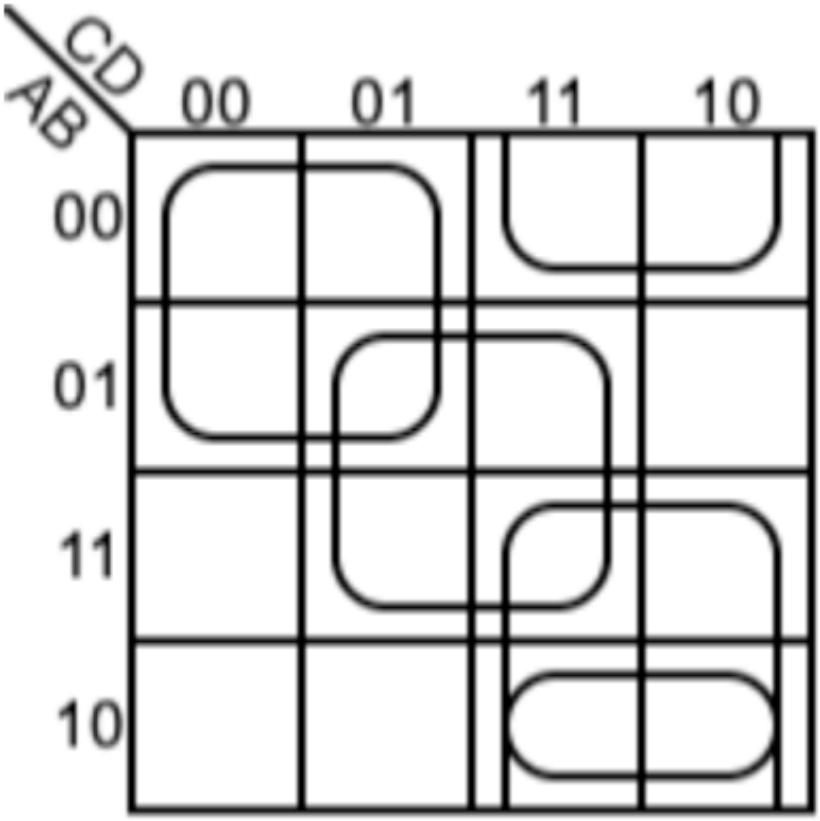
$$D_A = A'BCD' + AC'D' + AB'C' \tag{38}$$

Kmap of D_B:



$$D_B = AC'D' + A'B'CD' \tag{39}$$

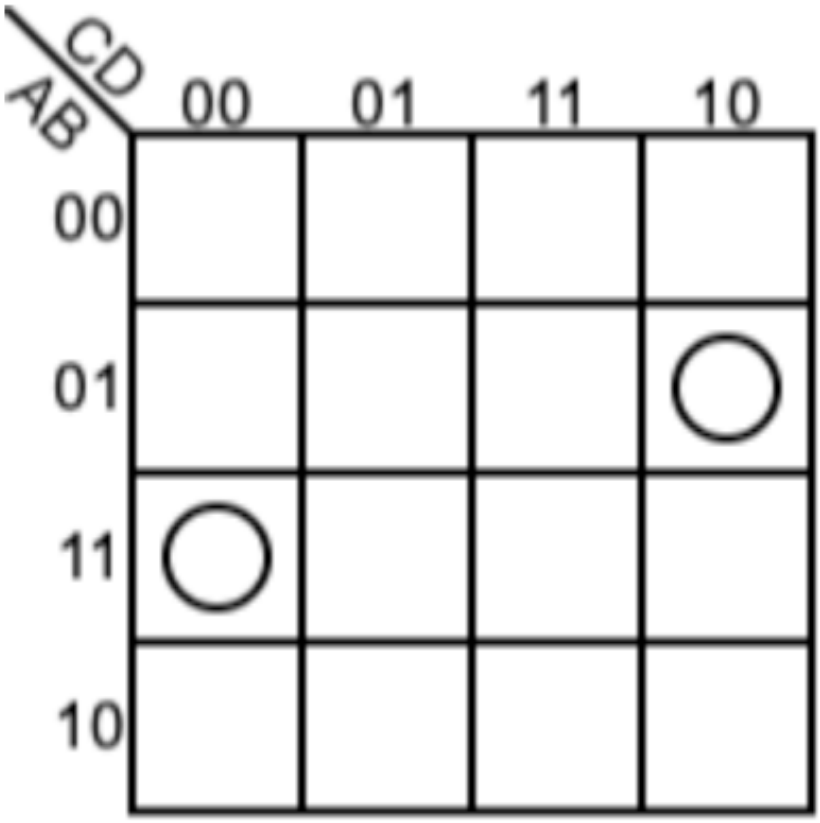
Kmap of D_C:



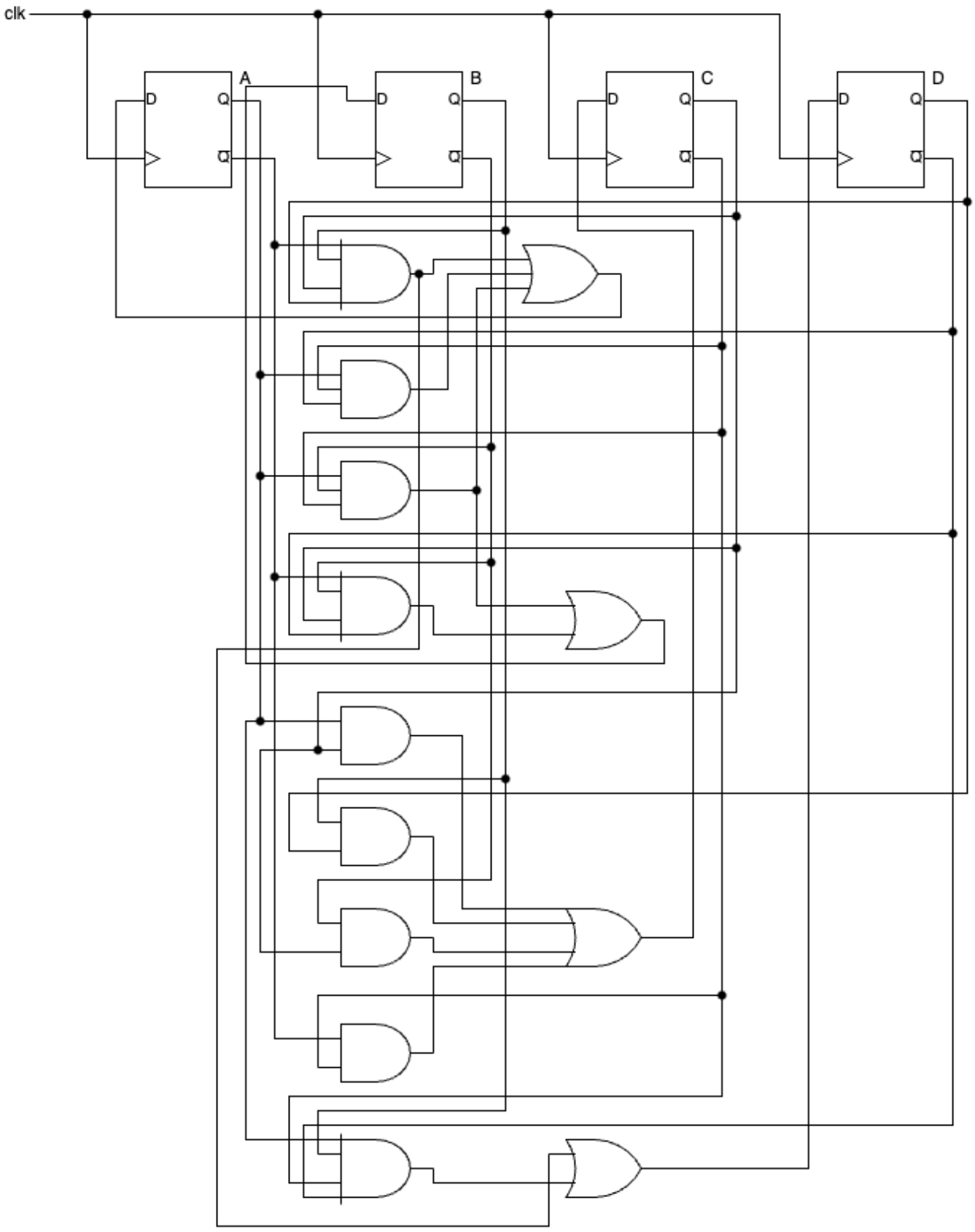
$$D_C = AC + BD + B'C + A'C' \tag{40}$$

Kmap of D_D:

$= D'_A$

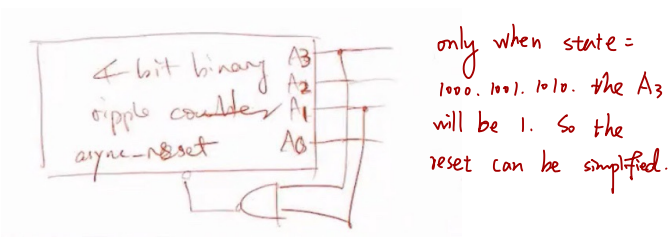


$$D_D = ABC'D' + A'BCD' \tag{41}$$



Question 6

- (15points)Implement a BCD ripple counter using a four bit binary ripple counter with asynchronous clear and external NAND gates. Draw the logic diagram.

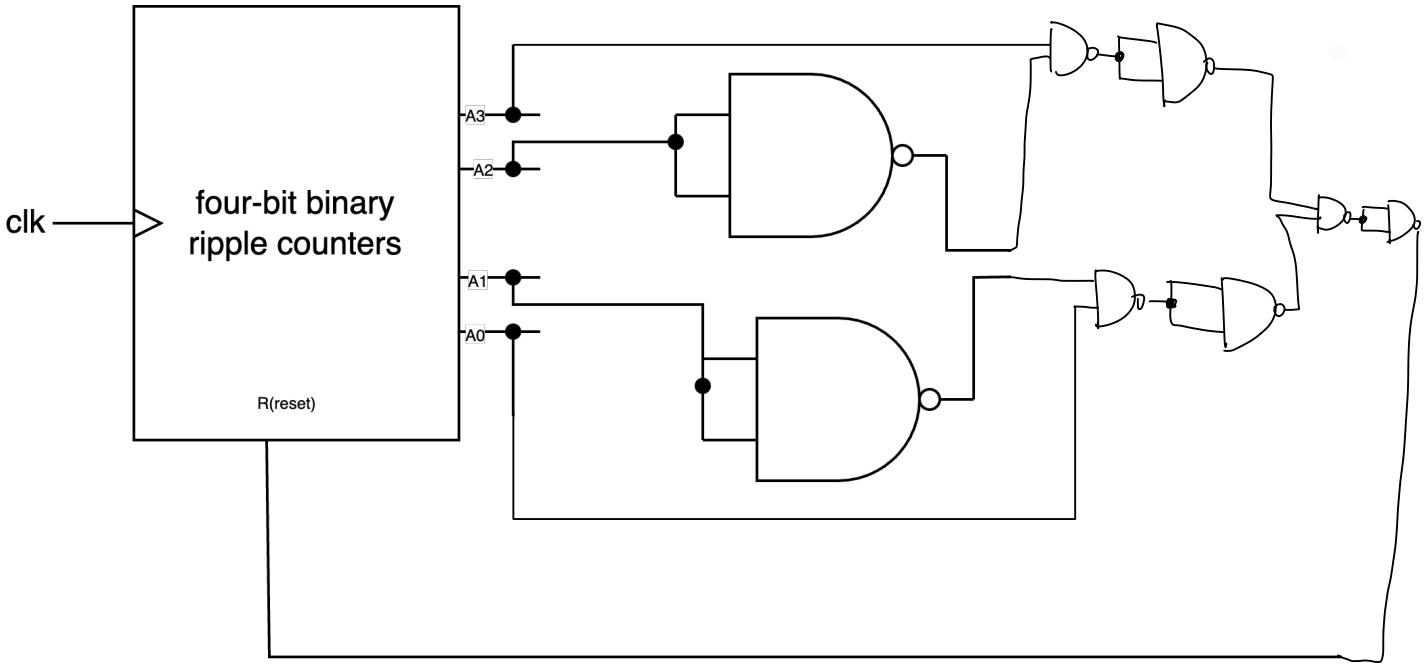


As the BCD ripple counter will go to state 0000 after reach state 1001, which is the difference compared to a four bit binary ripple counter.

As for NAND, $AB = [(AB)'(AB)']'$

So we can add NAND gates to let the circuit reset to 0000 after it reached 1001.

$$A_3A_2'A_1'A_0 = A_3(A_2A_2)'(A_1A_1)'A_0 = (A_3A_2'A_1'A_0)'[(A_3A_2'A_1'A_0)'(A_3A_2'A_1'A_0)']' \quad (42)$$



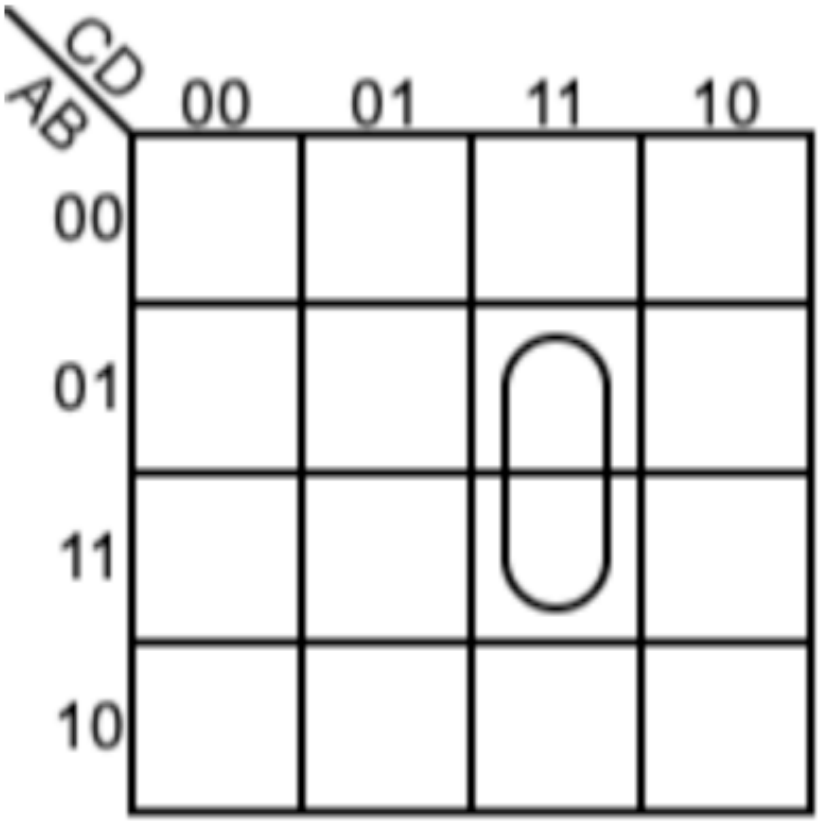
Question 7

- (15 points) Determine the input equations for a BCD counter that uses
  1. only JK flip-flops, and
  2. only D flip-flops.

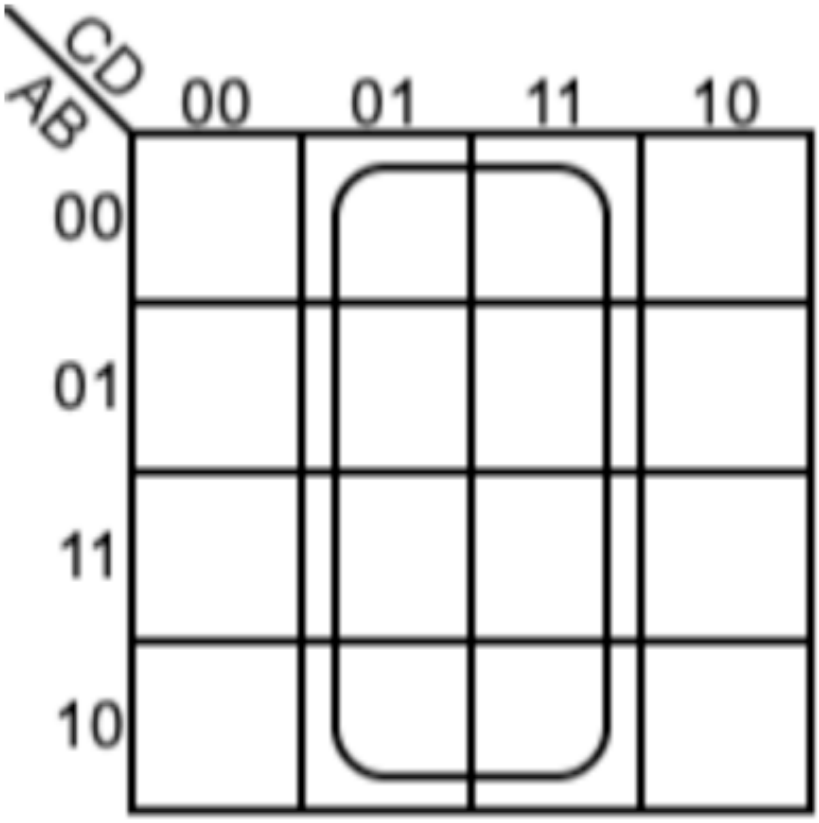
  Compare these two designs and the one given in the lecture that uses only T flip-flops. Determine which one is the most efficient and why.

  1. Use ABCD to represent the states.

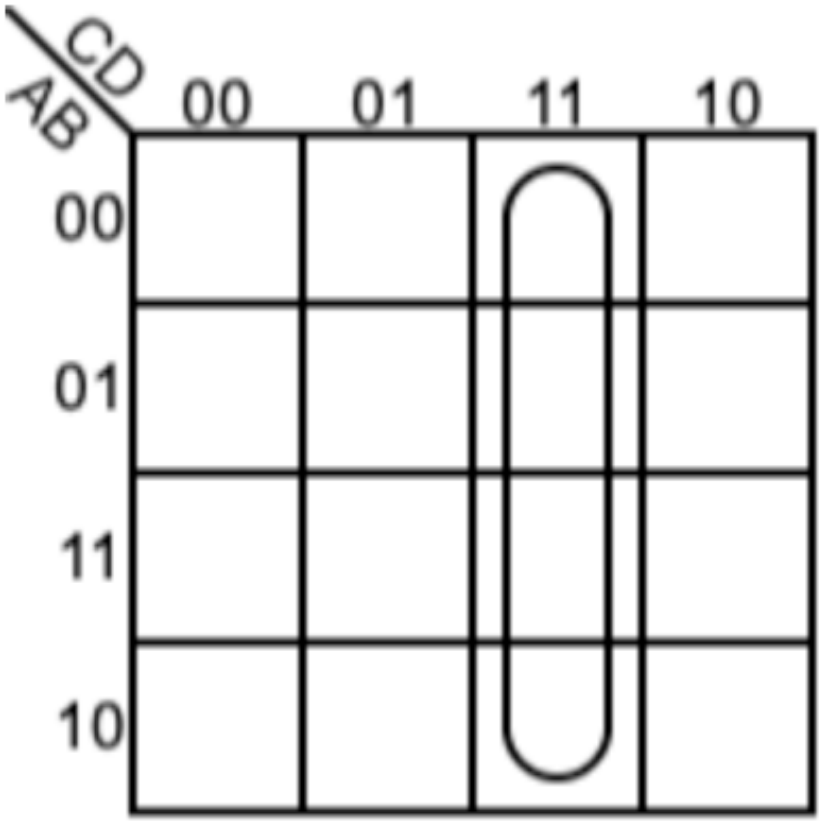
A(t)	B(t)	C(t)	D(t)	A(t+1)	B(t+1)	C(t+1)	D(t+1)	J_A	K_A	J_B	K_B	J_C	K_C	J_D	K_D
0	0	0	0	0	0	0	1	0	x	0	x	0	x	1	x
0	0	0	1	0	0	1	0	0	x	0	x	1	x	x	1
0	0	1	0	0	0	1	1	0	x	0	x	x	0	1	x
0	0	1	1	0	1	0	0	0	x	1	x	x	1	x	1
0	1	0	0	0	1	0	1	0	x	x	0	0	x	1	x
0	1	0	1	0	1	1	0	0	x	x	0	1	x	x	1
0	1	1	0	0	1	1	1	0	x	x	0	x	0	1	x
0	1	1	1	1	0	0	0	1	x	x	1	x	1	x	1
1	0	0	0	1	0	0	1	x	0	0	x	0	x	1	x
1	0	0	1	0	0	0	0	x	1	0	x	0	x	x	1



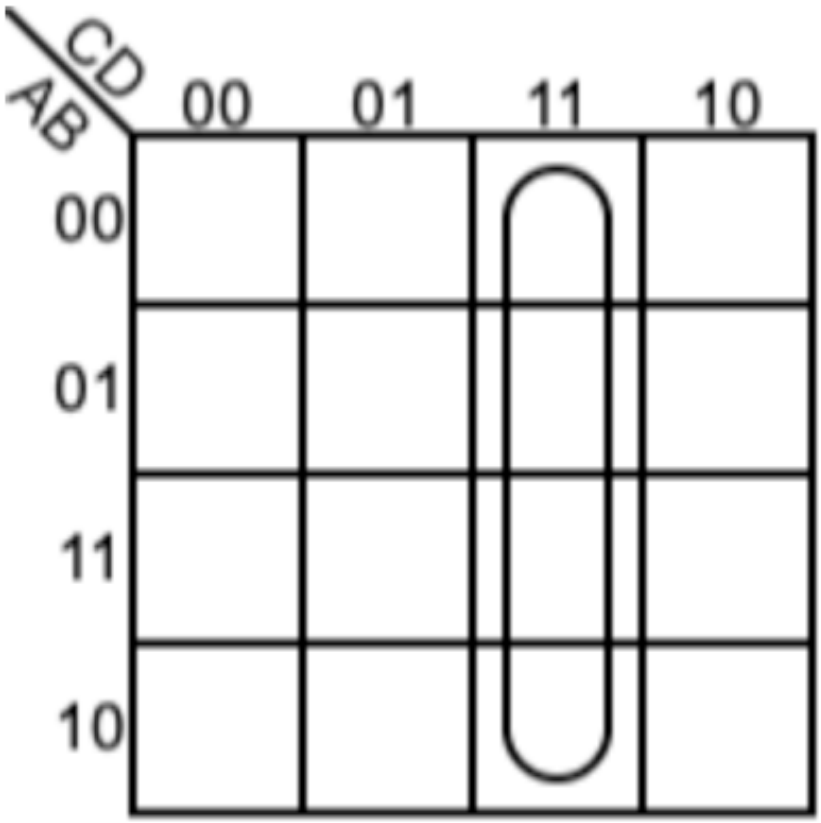
$$J_A = BCD \tag{43}$$



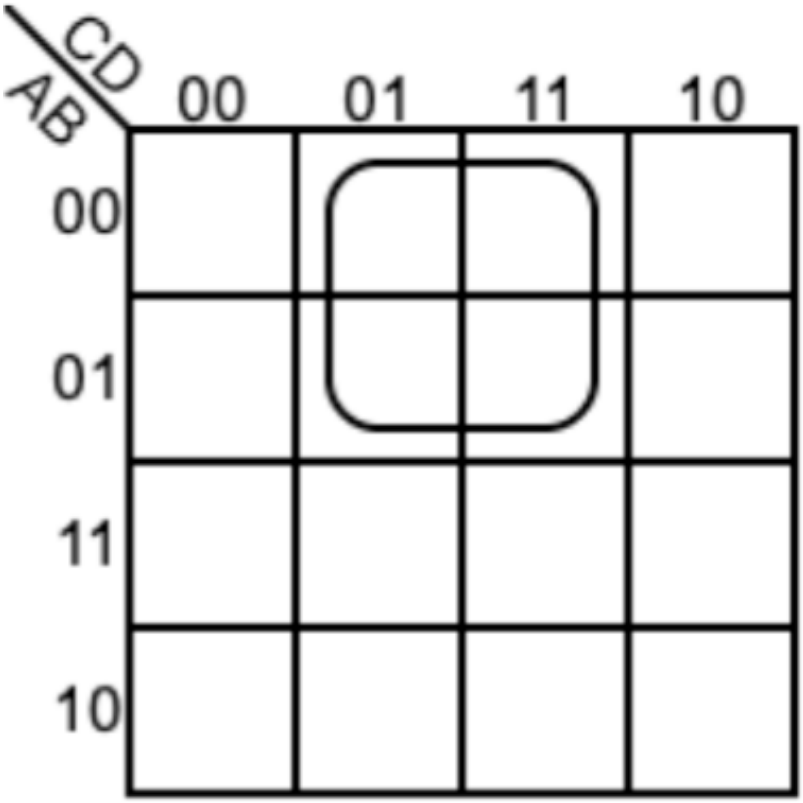
$$K_A = D \tag{44}$$



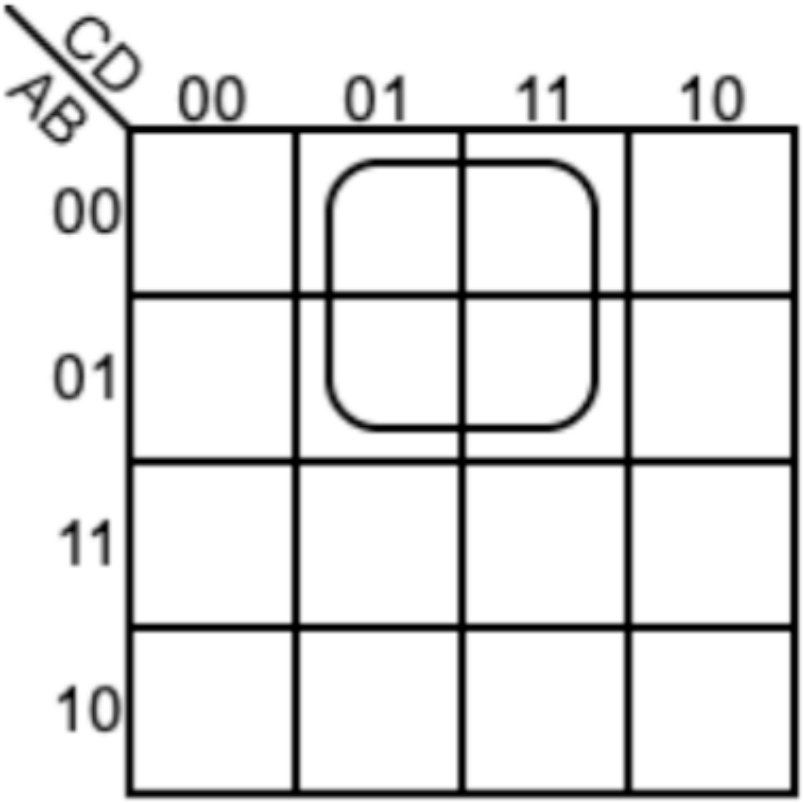
$$J_B = CD \tag{45}$$



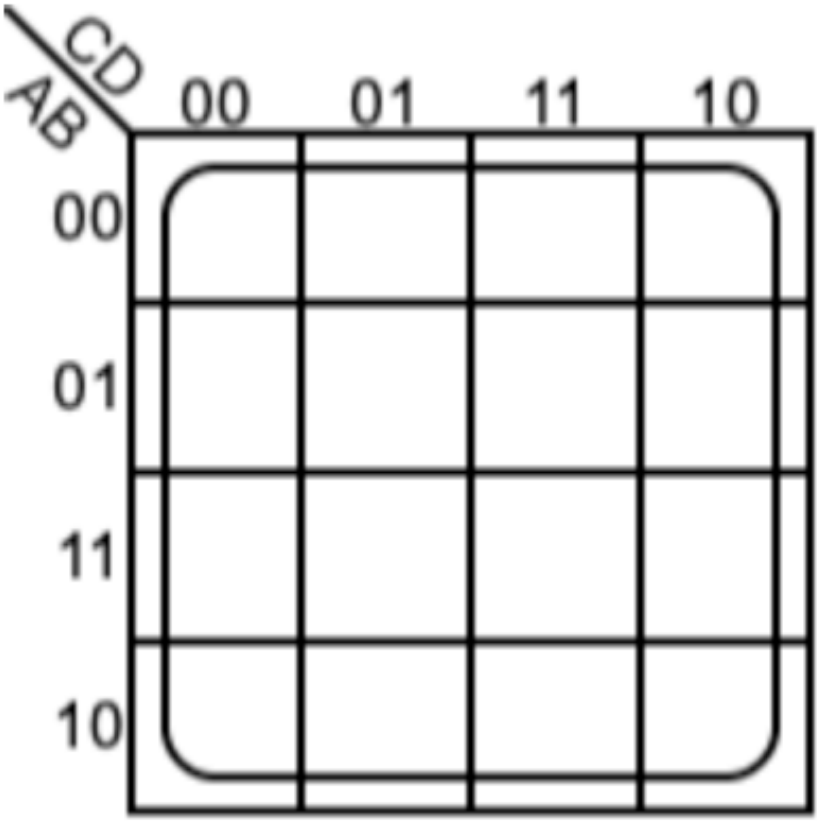
$$K_B = CD \tag{46}$$



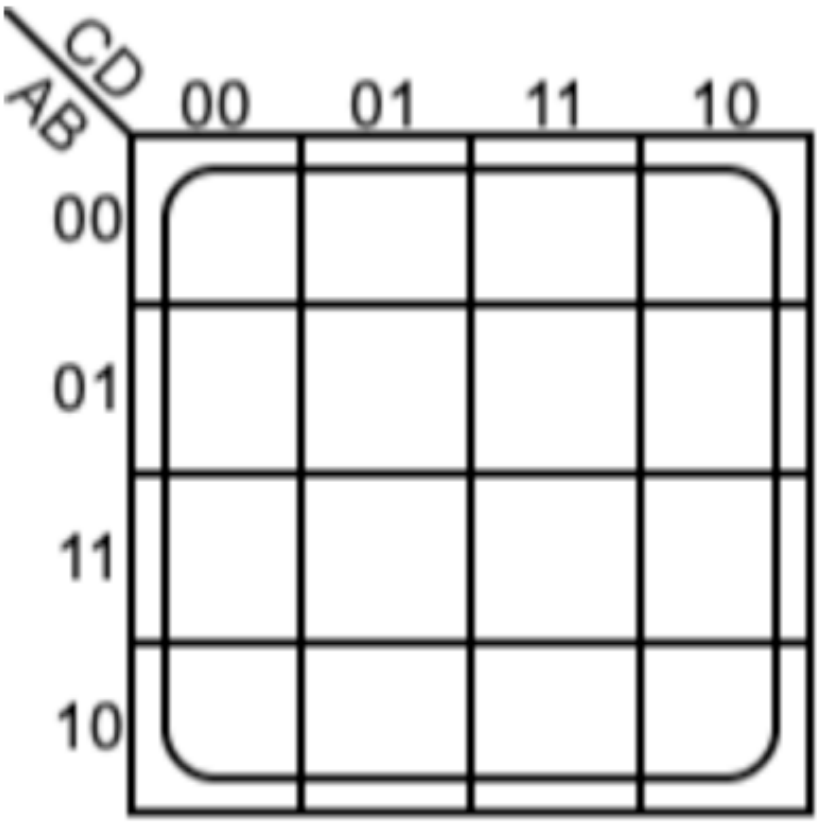
$$J_C = A'D \tag{47}$$



$$K_C = A'D \tag{48}$$

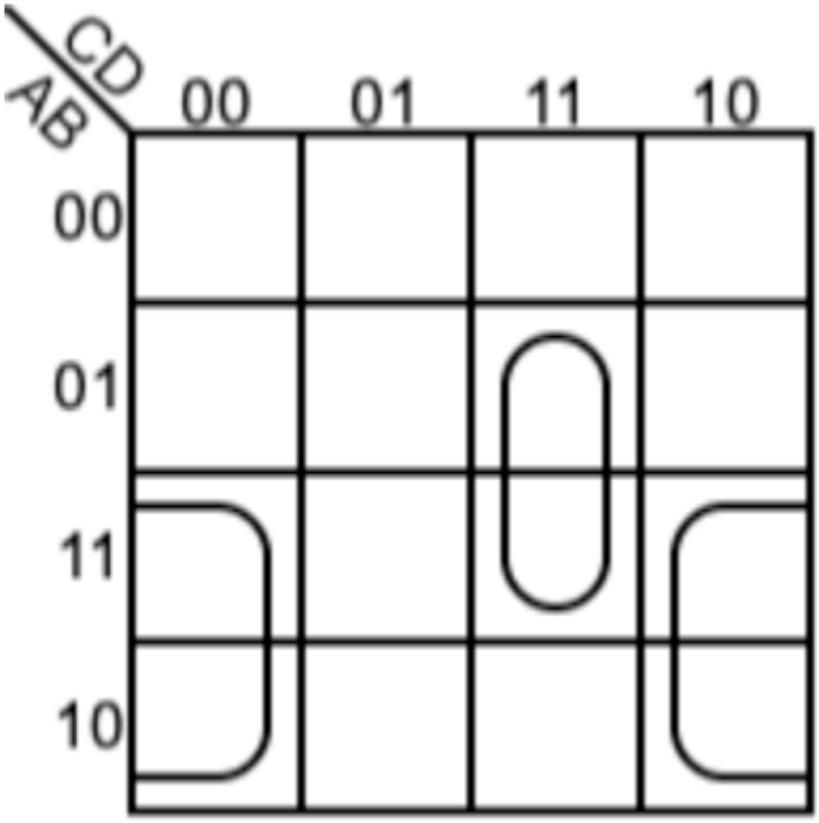


$$J_D = 1 \tag{49}$$

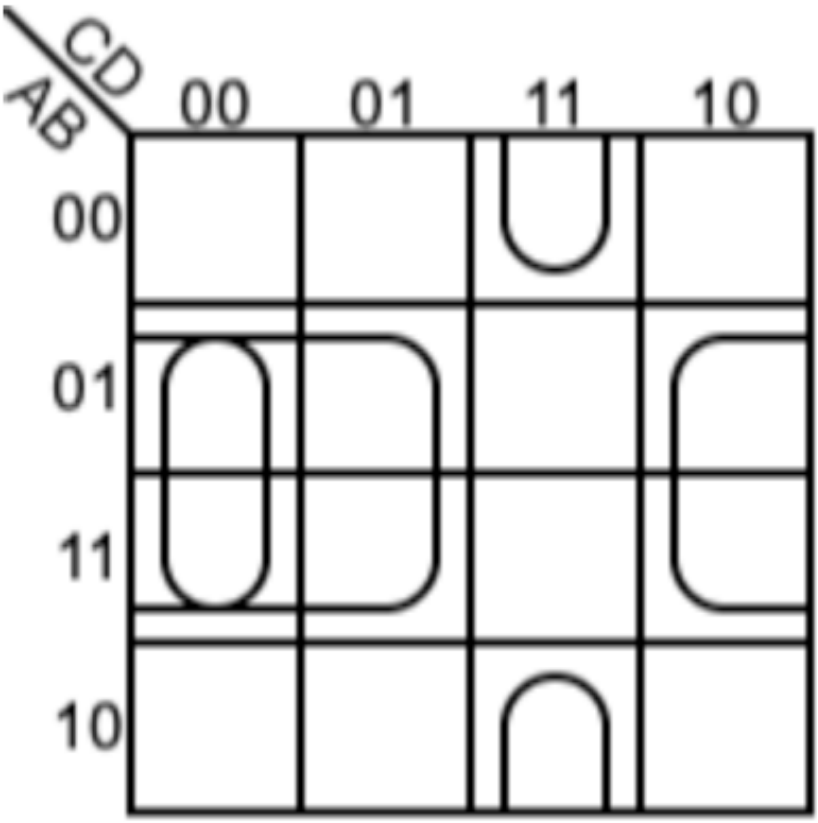


$$K_D = 1 \tag{50}$$

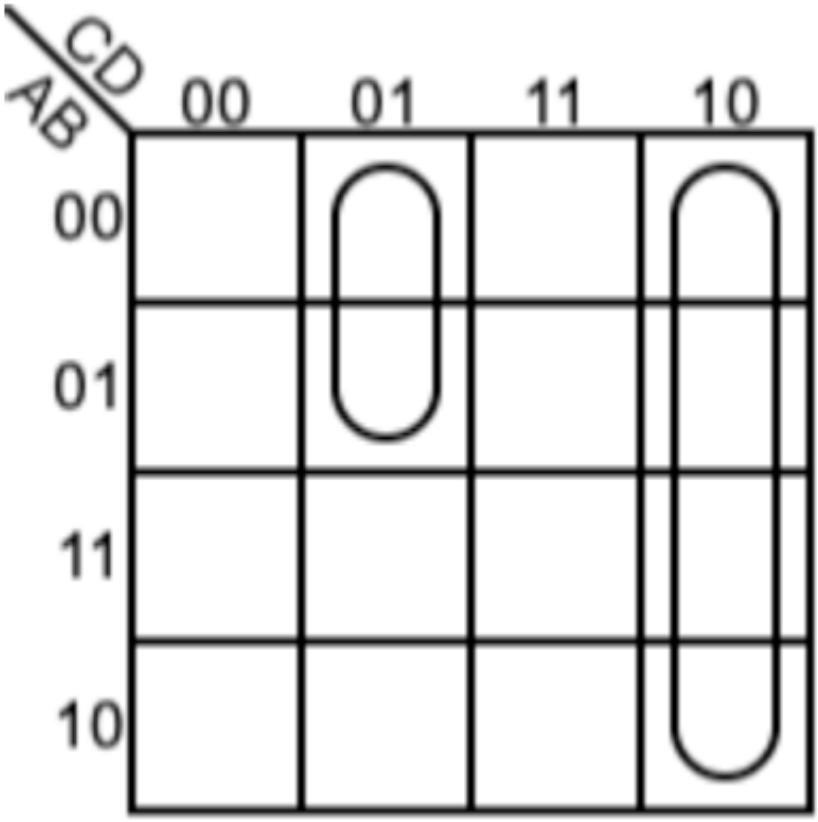
A(t)	B(t)	C(t)	D(t)	A(t+1)	B(t+1)	C(t+1)	D(t+1)	D_A	D_B	D_C	D_D
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	1	0	0	0	1	0
0	0	1	0	0	0	1	1	0	0	1	1
0	0	1	1	0	1	0	0	0	1	0	0
0	1	0	0	0	1	0	1	0	1	0	1
0	1	0	1	0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	1	0	1	1	1
0	1	1	1	1	0	0	0	1	0	0	0
1	0	0	0	1	0	0	1	1	0	0	1
1	0	0	1	0	0	0	0	0	0	0	0



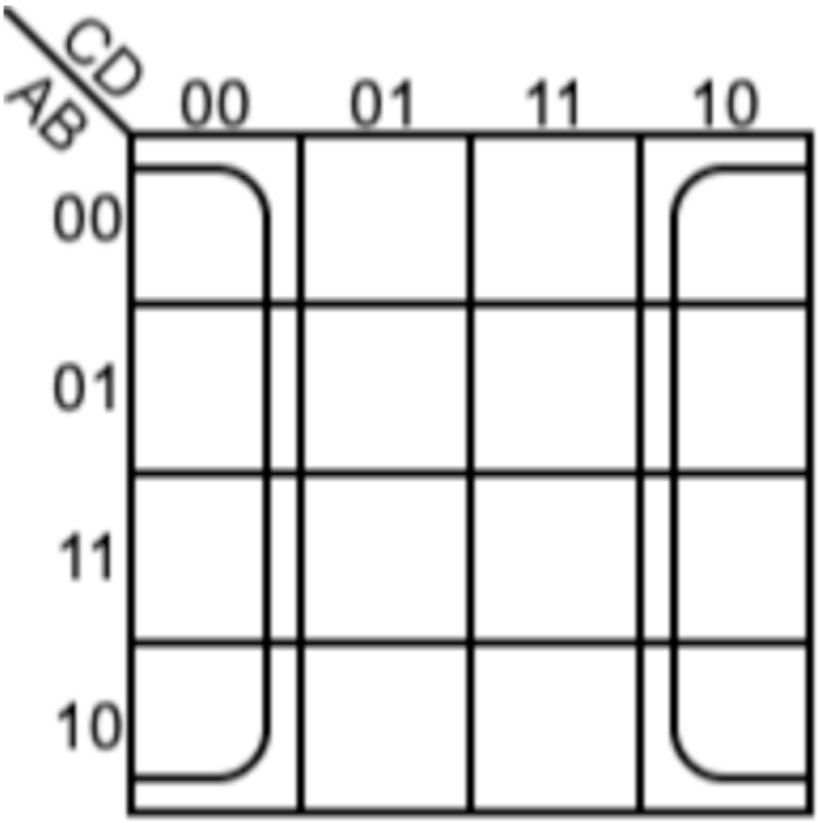
$$D_A = AB'C'D' + A'BCD \tag{51}$$



$$D_B = BC' + BD' + B'CD \tag{52}$$



$$D_C = CD' + A'C'D \tag{53}$$



$$D_D = D' \tag{54}$$

the most effective one is the one that made by JK ff as it only use 3 AND gates, which uses less gates than other implementation method.