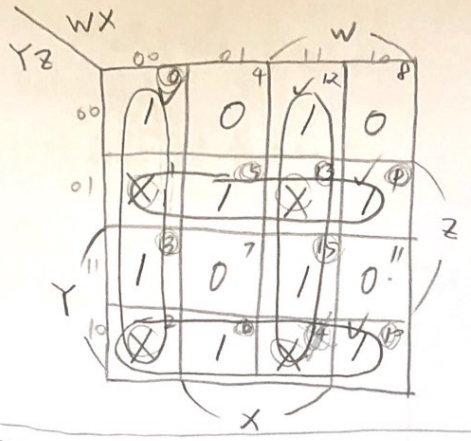
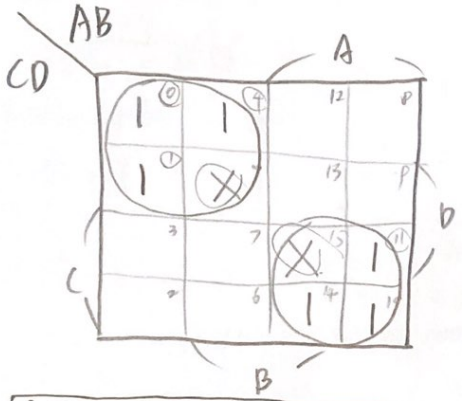
3.3 (a) $f(w,x,y,z) = \Sigma m(3,5,6,8,10,12,15) + \Sigma d(1,2,13,14)$

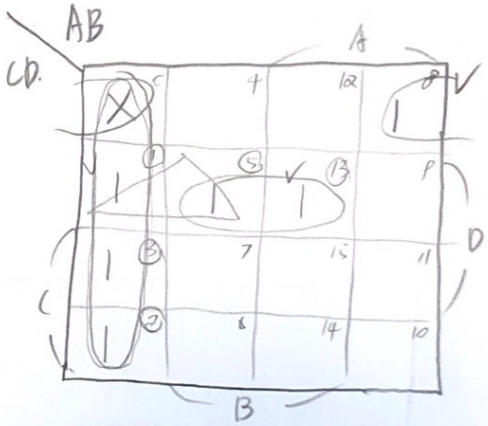


$f(w,x,y,z) = wx + y'z + yz' + w'x'$

(b) $f(A,B,C,D) = \Sigma m(0,1,4,10,11,14) + \Sigma d(5,15)$



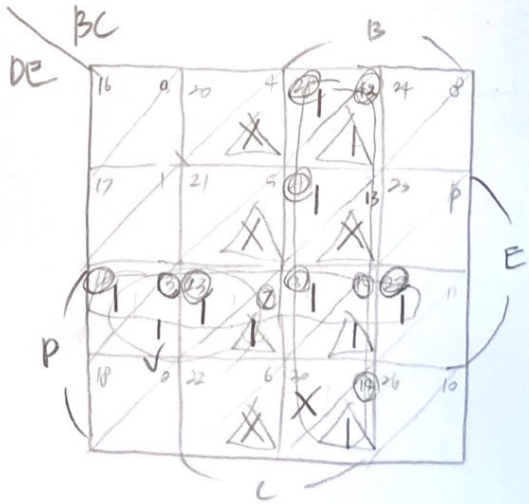
$f(A,B,C,D) = A'C' + AC$



$f(A,B,C,D) = BC'D + B'C'D' + A'B'$

(c) $f(A,B,C,D) = \Sigma m(1,2,3,5,8,13) + \Sigma d(0)$

(d) $f(A,B,C,D,E) = \Sigma m(3,7,12,14,15,19,23,24,28,29,31) + \Sigma d(4,5,6,13,30)$



$f(A,B,C,D,E) = BC + ADE + B'DE$

"~~for A is 1~~"

~~white~~ : [0,15] ~~black~~ : [16,31]

3-4 (a)

- PIs : $w'x'$, $y'z$, yz', xy,
- E·PIs : $w'x'$, $y'z$, yz', xy.
- redundants : 0
- n of DCs set to 1 : 4

(b)

- PIs : $A'C'$, AC
- EPIs : $A'C'$, AC
- redundants : 0
- n of DCs "1" : 2.

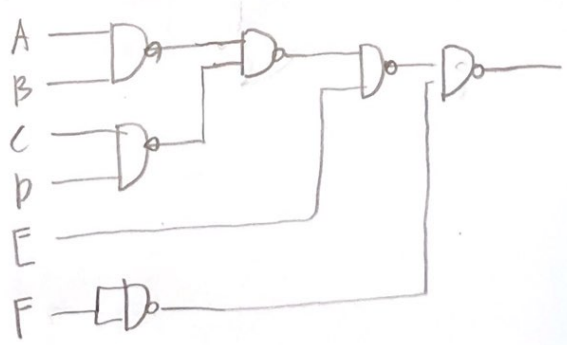
(c)

- PIs : $A'B'$, $BC'D$, $A'C'D$, $B'C'D'$
- EPIs : $A'B'$, $BC'D$, $B'C'D'$
- redundants : 1
- n of DCs "1" : 1

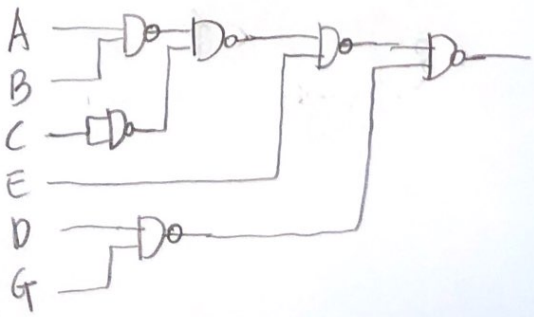
(d)

- PIs : $B'DE$, $A'C$, BC, ADE.
- EPIs : $B'DE$, ADE, BC
- redundants : 1
- n of DCs "1" : 2.

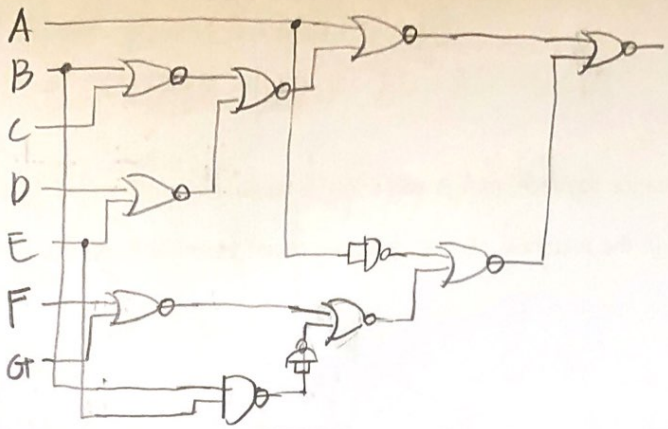
3-12 (a) $(AB+CD)E+F$



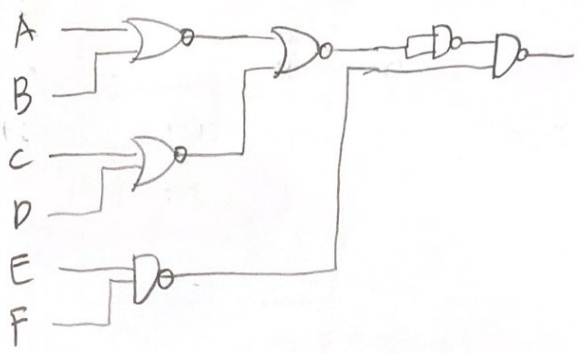
(b)



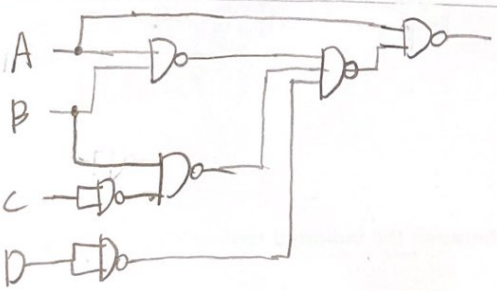
3-12 (c)



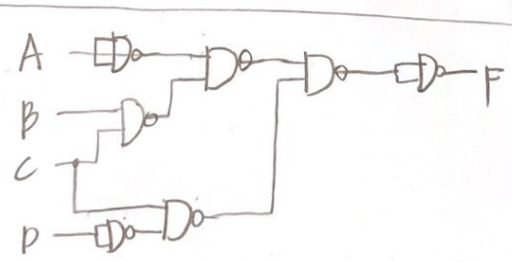
(d)

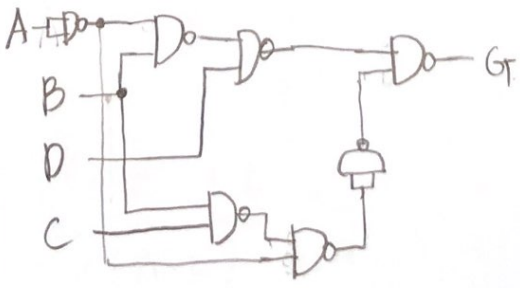


(e)



3-16 (a)

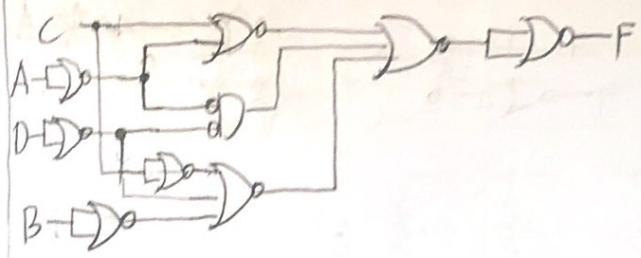




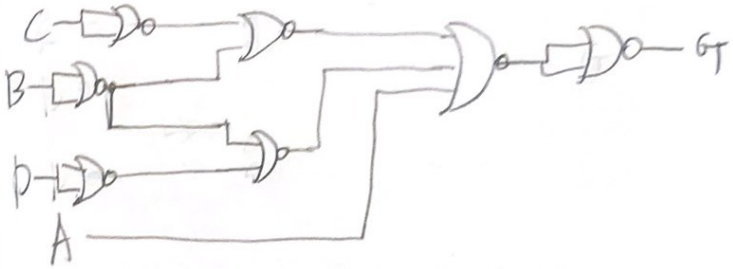
3-16 (b) $F(A,B,C,D) = (A+BC)(C'+D)$

$= AC'+AD+\cancel{B}+BCD = AC'+AD+BCD$

$AC'+AD+BCD$



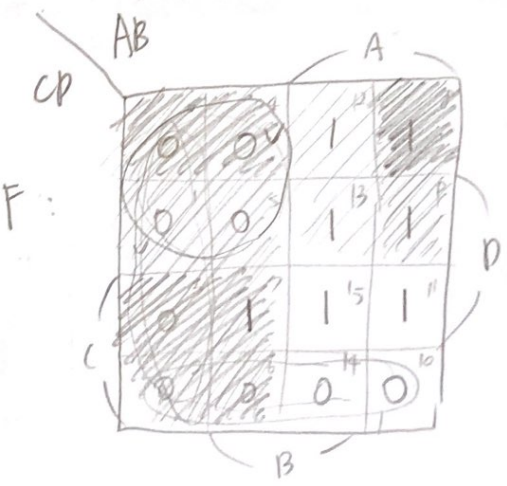
$G(A,B,C,D) = (\cancel{AD}+B'D)+(A+BC) = A+BC+B'D.$



(c) $F = AC'+AD+BCD.$

$G = A+BC+B'D.$

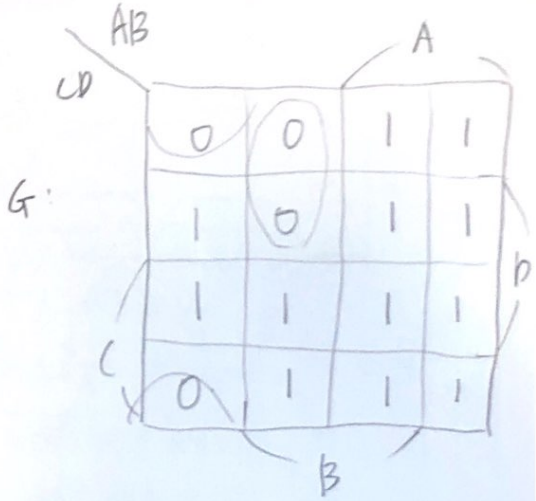
(d)



~~$F = (A'+C')\cdot(A'+B')\cdot(C+D')$~~

$F' = A'C'+A'B'+CD'$

$\boxed{F = (A+C)\cdot(A+B)\cdot(C'+D)}$



$G' = A'B'D'+A'BC'$

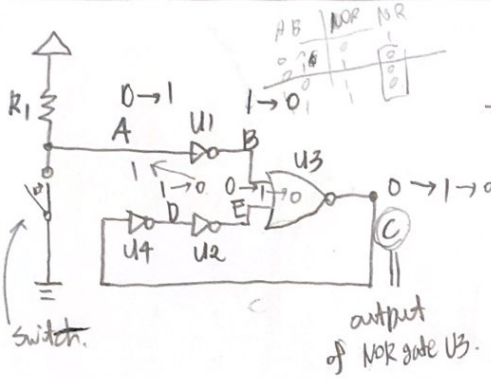
$G = (A+B+D)\cdot(A+B'+C)$

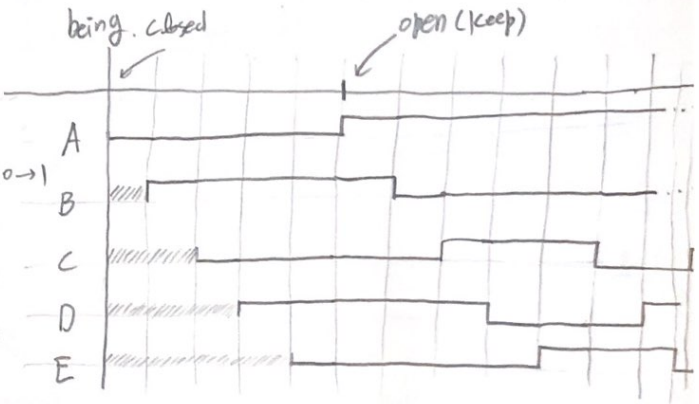
3-16 (e) gates, literal.

	n(gate)		n(literal)	
(a)	7	7	4	4
(b)	7	7	4	4
(c)	4	4	4	4
(d)	4	4	4	4

∴ simplest implementation is minimized forms like (c), (d) in terms of n of gates & literals.

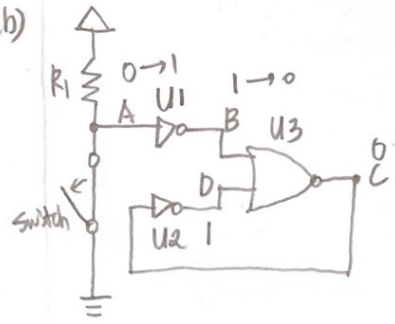
3-21 (a)

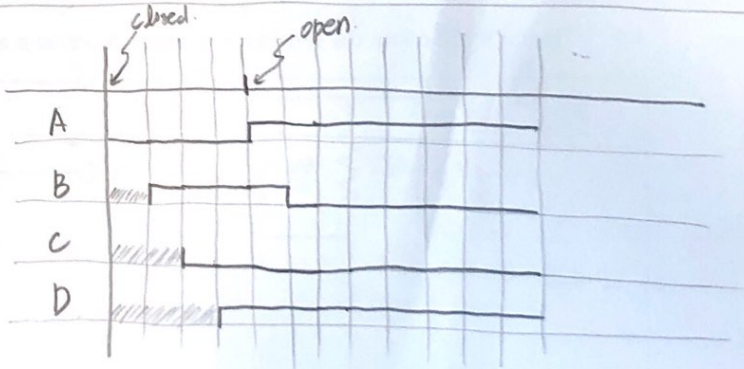




∴ As a result, if switch (being close → being open), output of U1 (B) keep '0' and output of U3 changing 0→1→0→1→ ···. Because input of U3 be (0,0), (0,1) ⇒ **oscillate**

(b)



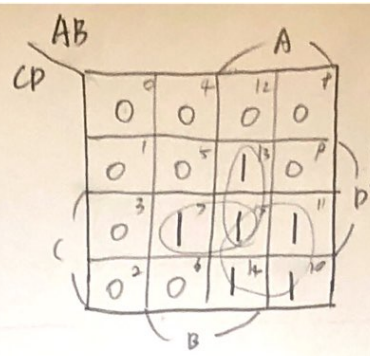


∴ As a result, U3's 1st input B changes 1→0, but, 2nd input D kept its value 1. ⇒ C does not oscillates.

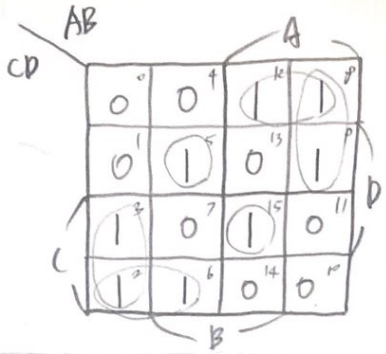
3-26. (a)

input				output			
A	B	C	D	X	Y	Z	
0	0	0	0	0	0	0	(0)
0	0	0	1	0	0	1	(1)
0	0	1	0	0	1	0	(2)
0	0	1	1	0	1	1	(3)
0	1	0	0	0	0	1	(4)
0	1	0	1	0	1	0	(5)
0	1	1	0	0	1	1	(6)
0	1	1	1	1	0	0	(7)
1	0	0	0	0	1	0	(8)
1	0	0	1	0	1	1	(9)
1	0	1	0	1	0	0	(10)
1	0	1	1	1	0	1	(11)
1	1	0	0	0	1	1	(12)
1	1	0	1	1	0	0	(13)
1	1	1	0	1	0	1	(14)
1	1	1	1	1	1	0	(15)

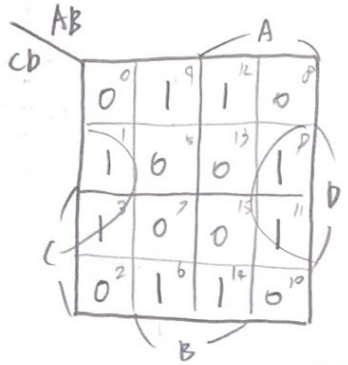
(b)



$X = BCD + ABD + AC$

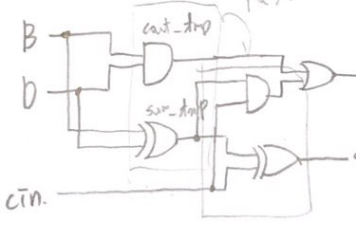


$Y = A'BC'D + ABCD + AB'C' + AC'D' + A'B'C + A'CD'$



$Z = B'D + BD' = B \oplus D$

(c) full-adder circuit.



$cout = B \cdot D + [(B \oplus D) \cdot cin]$ ($BD' + B'D$)

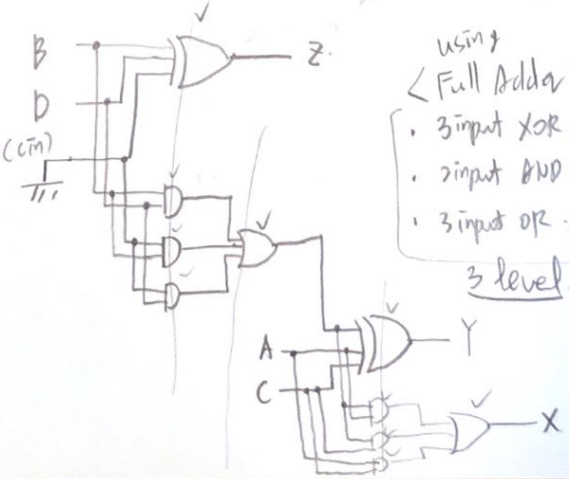
$= B \cdot D' \cdot cin + B' D cin + BD$

$sum = cin \oplus (B \oplus D)$

$cin \oplus BD' + B'D$

$= cin(BD + B'D') + cin'(BD' + B'D)$

∴ Using (a) & (b) is relatively faster in terms of delay. but it requires 6-input and 4 input gates. so it's safe to use Full Adder.



using < Full Adder >

- 3 input XOR : T (3)
- 2 input AND : 正 — (6)
- 3 input OR : T (2)

3 level imple, 11 gates.

vs.

using < (a) & (b) Adder >

- 3 input AND : 正 — (6)
- 2 input AND : — (1)
- 4 input AND : T (2)
- 2 input XOR : — (1)
- 3 input OR : — (1)
- 6 input OR : — (1)

2 level imple, 12 gates