1. In complement number system, a constant M is used to define a negative number, i.e. negative number of x is represented by M-x, which is non-negative.

   (a) If we use M = 2^k-1 (1's complement), where k is the number of bits, there are two representations of 0 (one +0 another -0), which is a waste. What happens if we use M = 2^k+1?

If we use M = 2^k-1, then M-x is 2^k+1 when x = 0. But, 2^k+1 is beyond the range of numbers that can be represented by k bits. It is also applied to the negative number of 1. So we are not able to represent -1 in this complement number system.

The least negative number will be represented by 2^{k-1} (100….00). This number satisfies 2^k+1-x = 2^{k-1}. This equation tells us x = 2^{k-1}+1.

Hence, we can represent positive numbers from 0 to 2^{k-1}-1, and negative numbers from -2 to –(2^{k-1}+1).

The objective of complement number system is to represent negative numbers using positive numbers because there is no negative number in the computer system.

So if you didn't write down "it can't represent negative number of -1," then your score was marked down -2.

If you only write down a waste of 0, your score was marked down -4.

   (b) Now, we want to represent integers in the range [-N, +P] using complement number systems. What is the value for M such that all the numbers within the range are uniquely represented?

After representing integers [0, P] using positive numbers [0, P], we want to represent negative integers [-N, -1] using complement numbers.

We have to represent negative integers using M-x, where x in the range [1, N], so M-N is the least complement number of all complement numbers. After representing positive numbers, the least number of remained positive numbers is P+1. Because we want to be uniquely represented, M-N must be equal to P+1. Thus, M = N+P+1.

If your answer is M ≥ N+P+1, then your score was marked down -1.

If you assumed that we use k-bits, then your score is 0.

2. Answer the followings:

   (a) Simplify three expressions of three variables ($f_1 = xz'$, $f_2 = xz' + x'z + x'y'z'$, $f_3 = x'yz + y'z'$) using Quine-McClusky method. [15 pts]

				f_1	f_2	f_3
0	0	0	0	0	1	1
1	0	0	1	0	1	0
4	1	0	0	1	1	1
3	0	1	1	0	1	1
6	1	1	0	1	1	0

				f_1	f_2	f_3
(0,1)	0	0	–	0	1	0
(0,4)	–	0	0	0	1	1
(1,3)	0	–	1	0	1	0
(4,6)	1	–	0	1	1	0

				f_1		f_2					f_3		
				4	6	0	1	3	4	6	0	3	4
3	0	1	1					X				X	
4	1	1	1	X					X				X
(0,1)	0	1	0			X	X						
(0,4)	0	1	1			X			X		X		X
(1,3)	0	1	0				X	X					
(4,6)	1	1	0	X	X				X	X			

Essential prime implicants

→ f_1: xz'

f_2: xz'

f_3: $x'yz$, $y'z'$

➔ $f_1 = xz'$

$f_2 = xz'+y'z'+x'z$

$f_3 = x'yz+y'z'$

-5 points for not optimized answer

-5 points for wrong prime implicant chart

(b) Implement f_1 = b'd + a'b' + c'd and f_2 = a'd' + bc' + bd' with a two-level NAND-NAND circuit. [10 pts]

f_1 = b'd+a'b'+c'd

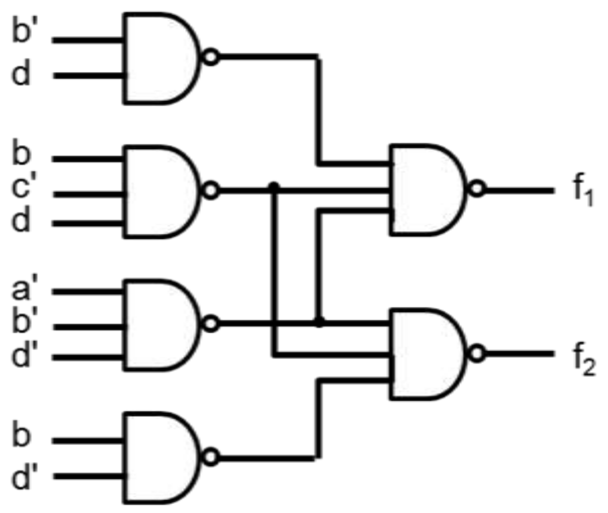
cd \ ab	00	01	11	10
00	1	0	0	0
01	1	1	1	1
11	1	0	0	1
10	1	0	0	0

f_2 = a'd'+bc'+bd'

cd \ ab	00	01	11	10
00	1	1	1	0
01	0	1	1	0
11	0	0	0	0
10	1	1	1	0

➔ f_1 = b'd + bc'd + a'b'd'

f_2 = bd' + bc'd + a'b'd'



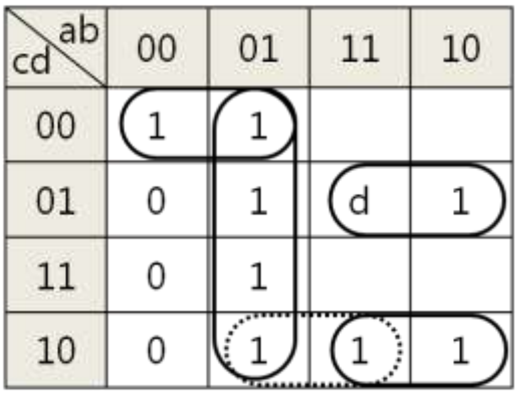
5 points: correct NAND-NAND conversion with not optimized function

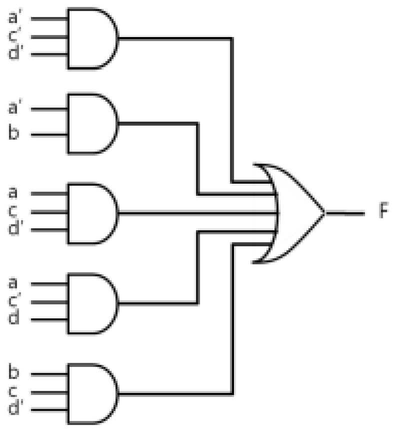
3 points: incorrect function but correct NAND-NAND conversion

0 points for incorrect NAND-NAND conversion

3. Answer the followings:

   (a) Implement F(a, b, c, d) = ∑m(0, 4, 5, 6, 7, 9, 10, 14) + ∑d(13) as a hazard-free circuit. The number of gates and the number of gate inputs have to be minimized. [10 pts]





F= a'c'd' + a'b + ac'd + acd' + bcd' [10 points]

※ If solution has POS expression, this is not minimum	→ 7 points
※ If solution includes a bc'd term, which is useless	→ 5 points
※ Solution don't consider hazard-free constraint	→ 2 points
※ **Final solution was written incorrectly**	→ **-1 points**

   (b) Gate delay typically increases as load capacitance increases. Therefore, smaller capacitance is preferred, which however is not always possible. Explain why. [5 pts]

Answer: Wire capacitance restricts load capacitance minimization

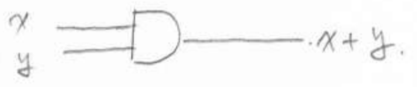
※ Intrinsic capacitance, driving current problem	→ 2 points
※ **The number of fanout gates**	→ **0 points**

4. Answer the followings.

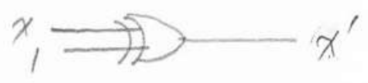
(a) Show that XOR and AND are functionally complete. You can use “1” as a gate input, but only for a single gate. [10 pts]

AND

(INV : 5points, OR : 5points)

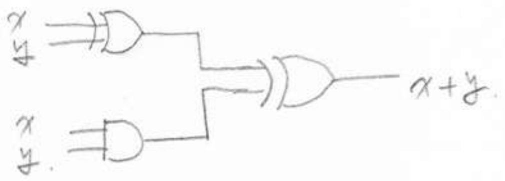


INV



$x'1 + x \cdot 0 = x'$

OR



$(x'y + xy') \oplus xy$

$= (xy' + x'y)'xy + (x'y + xy')(xy)'$

$= (x'+y)(x+y')xy + (x'y + xy')(x'+y')$

$= (xy + x'y')xy + (x'y + xy')$

$= xy + x'y + xy'$

$= x + x'y = x + y$

INV: 5 points, OR: 5 points

(b) DeMorgan’s law (x + y)’ = x’y’, (xy)’ = x’ + y’ can be extended to arbitrary number of variables. Prove (xyzw)’ = x’ + y’ + z’ + w’. [5 pts]

$(xyzw)' = (xy \cdot zw)'$ ↓ $A = xy$, $B = zw$.

$= (A \cdot B)'$ ↓ $(xy)' = x' + y'$ 적용.

$= A' + B'$

$= (xy)' + (zw)'$ ↓ $(xy)' = x' + y'$ 적용

$= (x' + y') + (z' + w')$

$\therefore (xyzw)' = x' + y' + z' + w'$

5. Answer the followings:

   (a) To use Q-M method to minimize a given expression, a list of on-sets (vertices in a Boolean space) is identified first. This seems somewhat redundant because the expression may be given as a simplified form (even though it is not the minimized one). We therefore want to see whether we can directly apply Q-M method to a given expression without generating all the on-sets (i.e. we try to combine 000-, 0-01, -111, 111-, and 10-- in some way or the other to minimize f = a'b'c' + a'c'd + bcd + abc + ab'). Is it possible to obtain the minimum expression? Explain how if it works; explain why if it does not. [10 pts]

   This is not possible.

   If we use the Q-M method we obtain all of the prime implicants. This allows us to select a minimum set of literals which will result in a minimum sum of products expression.

   If we start the Q-M method with a simplified form function, the problem is we cannot find all of the prime implicants, due to the fact that we only combine terms instead of minterms.

   For example in the given function, there are no terms that can be combined. However, the actual minimized function form is f = ac + b'c' + a'bd, which shows how limited the minimization process is when directly applying the Q-M method to a function.

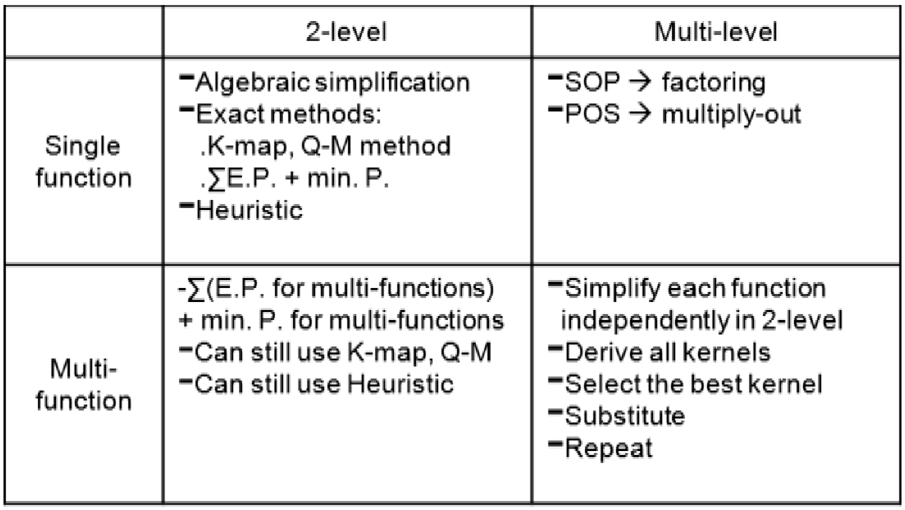
   5 points if reason is wrong

   (b) Show that (a' + b + c)(a + b) = (a + b)(b + c). Specify which Boolean theorems are used. [5 pts]

   (a'+b+c)(a+b) = (a'+b+c)(a+b)(b+c) -> Consensus theorem

   = (a+b)(b+c) -> Simplification theorem

(c) Explain the basic steps of minimizing a single Boolean expression in 2-level, multiple expressions in 2-level, a single expression in multi-level, and multiple expressions in multi-level. [5 pts]



	2-level	Multi-level
Single function	-Algebraic simplification -Exact methods: .K-map, Q-M method .∑E.P. + min. P. -Heuristic	-SOP → factoring -POS → multiply-out
Multi-function	-∑(E.P. for multi-functions) + min. P. for multi-functions -Can still use K-map, Q-M -Can still use Heuristic	-Simplify each function independently in 2-level -Derive all kernels -Select the best kernel -Substitute -Repeat

-1 points for each wrong answer

(d) We are given an expression *f*, which we want to minimize in 2-level. Imagine that somehow we discovered all its essential prime implicants, and denote them as SOP f_1. In the expression *f*, we assume that all the minterms that belong to f_1 are now don't cares, and denote the modified expression by f_2. If we minimize f_2 and add it to f_1 (i.e. $f_1 + f_2$), it is a minimum SOP for *f*. If this is true, explain; otherwise, suggest a counter-example. [5 pts]

A minimum SOP expression is found by first finding all essential prime implicants and then selecting a minimum subset of the remaining prime implicants to cover the remaining minterms.

To select this subset, we set the minterms covered by essential prime implicants as don't cares (because they are already included but can be used to reduce the expression) and find the minimum expression. If we look at the prime implicants found for this expression, they are the prime implicants for the original function.

Because the expression f2 is simplified, this is the minimum expression representing the remaining minterms. f1 is also a minimum expression because all terms are essential prime implicants. If a term in f2 can be reduced by a term in f1 this means that it is an essential prime implicant which should be in f1. Thus such term cannot exist.

Therefore adding f1 and f2 forms a minimum SOP for f.

-1 points if the answer does not include essential prime implicants