

챕터7 정리

Ch7: MULTI-LEVEL GATE CIRCUITS / NAND AND NOR GATES

7.0 들어가며

- 7장의 주요 내용

- Boolean Expression으로 표현된 어떤 방정식을 게이트를 이용하여 구현함

근데 2단으로 할 건지 혹은 3단으로 할 건지

혹은 NAND만으로 NOR만으로 구현할 것인지에 대한 챕터임

- 산업 현장에서 실질적인 구현에서는 의미 있는 장

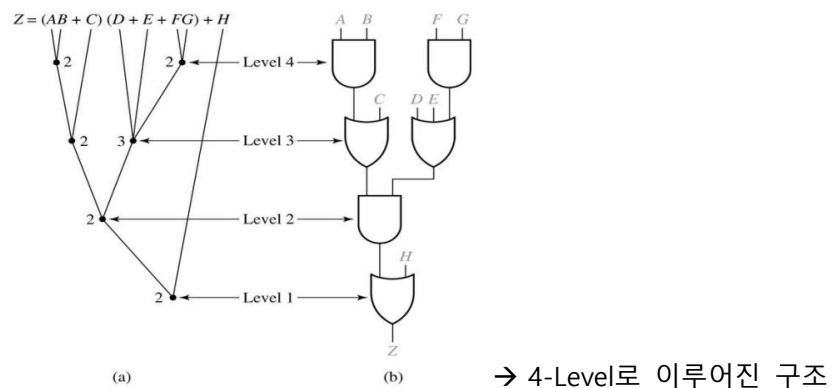
논리회로 배운다는 입장에서 7장 8장은 그리 중요도가 높은 챕터는 아님

7.1 Multi-Level Gate Circuits

- Multi-Level Gate Circuits

- Terminology: AND-OR, OR-AND, OR-AND-OR, AND and OR

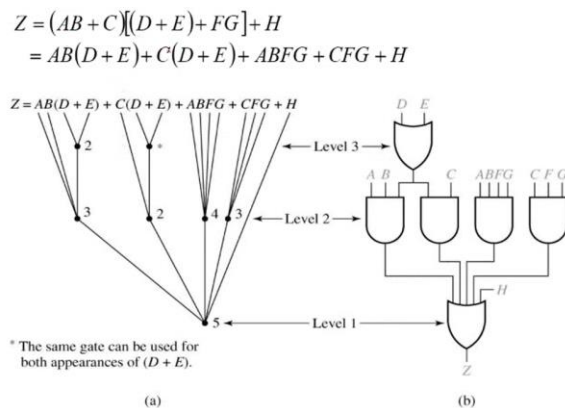
- Four-Level Realization of Z



→ 레벨 4부터 보면 AND-OR-AND-OR 순으로 되어있음; AND-OR-AND-OR 구조

또한 각 레벨의 입력의 개수를 카운팅 할 수 있다. 위의 경우는 13개의 입력

- Three-Level Realization of Z



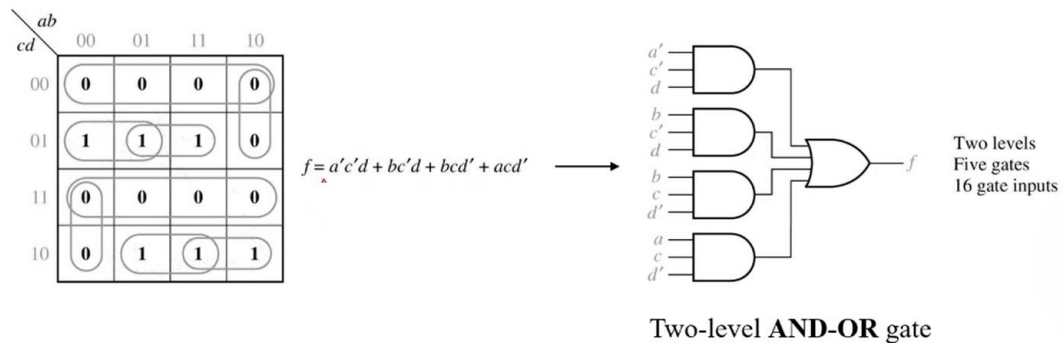
→ 3-Level Realization, OR-AND-OR Structure, 입력의 개수는 19개

- Example

- Multi-Level Design Using AND and OR Gates

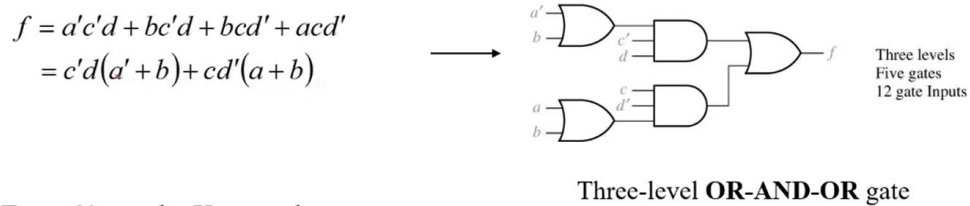
우리에게 친숙한 SOP 또는 POS형태는 2레벨 realization임

다음 예시를 봅시다: $f(a, b, c, d) = \sum m(1, 5, 6, 10, 13, 14)$



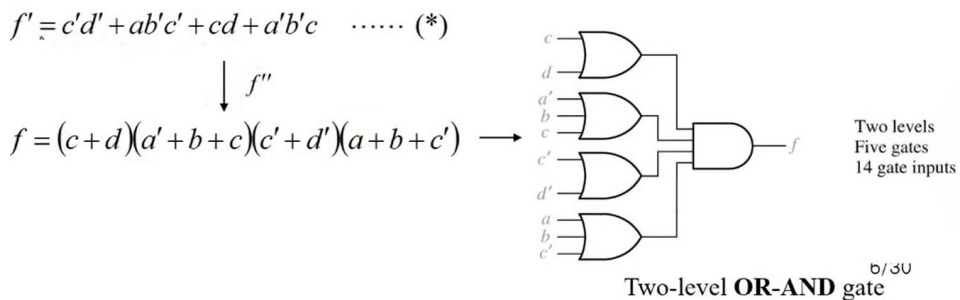
→ 적절한 그루핑을 통해 표현식을 얻을 수 있음. SOP 형태임.

각각의 항은 3 input OR 게이트임. AND-OR구조, 16개의 입력을 받음



→ 3 Level OR-AND-OR 게이트, 12개의 입력

From 0's on the Karnaugh map



→ 0에 초점을 맞추고 드모르간 취해 POS 형태를 얻음

2-Level OR-AND Structure, 14 input

- 이런 것들을 왜 구현, 표시하는가?

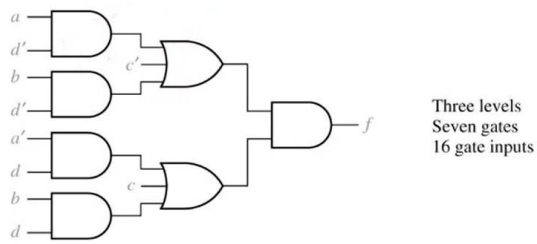
실제로 구현한다는 측면에선 게이트와 게이트에 들어가는 입력 하나하나의 측면에서 봤을 때는 구현 비용(cost)으로 간주할 수 있음. 즉, 게이트와 입력의 개수가 같은 경우가 적은 cost가 든다고 간주할 수 있음

- 2-Level 뿐만이 아니라 다중 Level로도 구현이 가능하다

위의 예시 식을 계속 갖고 와서 변형하면

Using $(X + Y)(X + Z) = X + YZ$, $f = [c + d(a' + b)][c' + d'(a + b)]$

이 식을 전개하면 $f = (c + a'd + bd)(c' + ad' + bd')$ → 3 레벨로 구현 가능함



Three-level **AND-OR-AND** gate

→ 위의 경우들보단 더 많은 cost를 요구함

· 정리

- 다양한 레벨의 Structure로 동일한 함수 표현을 할 수 있다.
- 다양한 형태에 따라 게이트의 수나 입력, 즉 cost가 달라진다.

7.2 NAND and NOR Gates

- 이 내용을 다시 배우겠다는 것이 아니라..
- NAND 또는 NOR밖에 없을 때 전체 표현식을 NAND, NOR만으로 구현하는 것을 초점으로 회로 설계를 NAND 또는 NOR 만으로 설계가 가능하다
- 트랜지스터의 면에선 AND, OR보다 NAND, NOR가 더 이득을 본다
 - NAND, NOR만으로 게이트를 구성하면 훨씬 이득을 보겠지?

· NAND gate

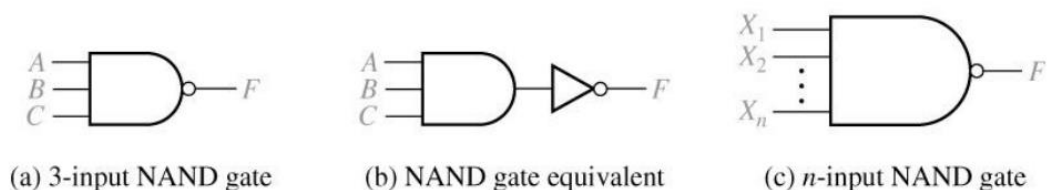
- $F = (ABC)' = A' + B' + C'$

→ $F = (X_1 X_2 \dots X_n)' = X_1' + X_2' + \dots + X_n'$

만약 3-Input NAND 게이트에서 2개의 Input만을 다루고 싶다면 → C'을 0으로 만들면 됨

즉, C에 High-Voltage(1)를 걸어주면 됨

- NAND = AND + Inverter



게이트 면에서는 인버터가 추가되었지만, 트랜지스터의 기본적인 특성에 인버터가 있음.

트랜지스터를 가지고 게이트를 구성하다 보면, NAND 게이트를 구성하는 트랜지스터의 개수가 AND 게이트에 사용되는 트랜지스터보다 적음. 트랜지스터의 입장에선, AND게이트는 NAND + 인버터의 개념이다.

· NOR gate

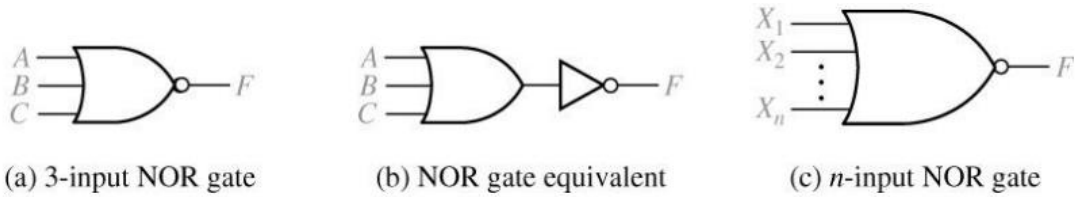
- $F = (A + B + C)' = A'B'C'$

→ $F = (X_1 + X_2 + \dots + X_n)' = X_1' X_2' \dots X_n'$

만약 3-Input NOR 게이트에서 2개의 Input만을 다루고 싶다면 → C'을 1로 만들면 됨

즉, C에 0을 보내주면 됨

- NOR = OR + Inverter

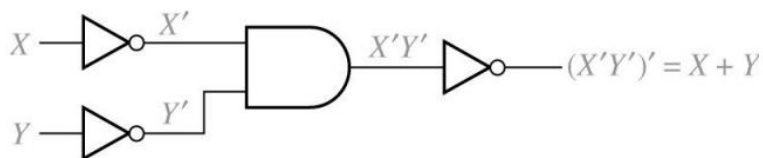


NOR 게이트 또한 마찬가지임. NOR 게이트가 OR 게이트보다 더 적은 개수의 트랜지스터를 필요로 함

- 다른 게이트를 이용하여 원하는 게이트 만들기

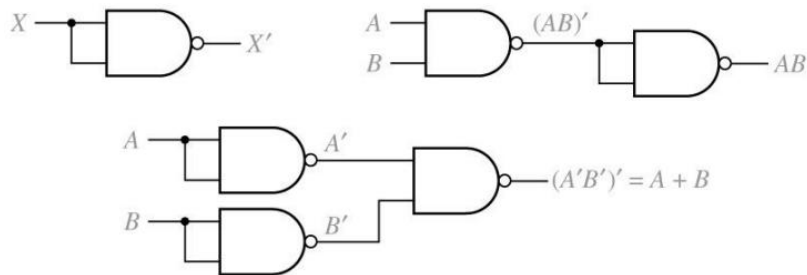
- OR realized by using AND and NOT

AND와 NOT 게이트만으로 OR 게이트를 구현하는 방법



→ 필요로 하는 게이트를 기존에 있는 다른 게이트들을 이용해 마음대로 만들 수 있다

- NAND gate realization of NOT, AND, and OR



→ 좌 상단: OR과 버블을 이용하여 인버터를 구상하는 방법(NAND로 NOT게이트)

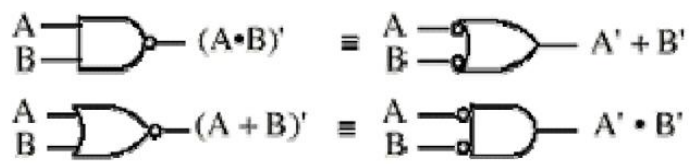
→ 우 상단: NAND로 AND 게이트 구성

→ 하단: NAND로 OR 게이트 구성

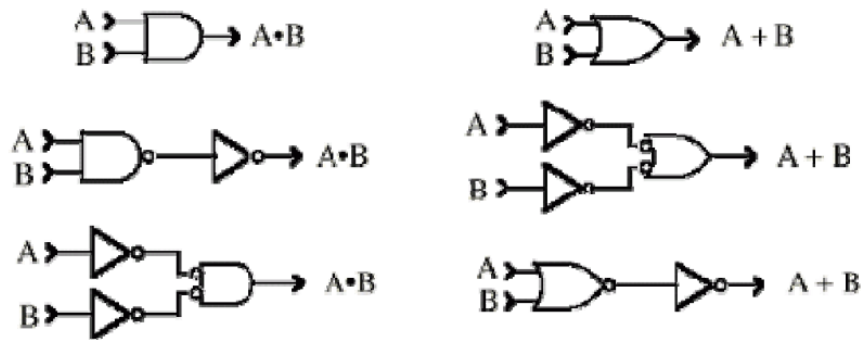
- AND realized by using OR and NOT

$$XY = (X' + Y')'$$

- NAND and NOR Equivalent Symbols



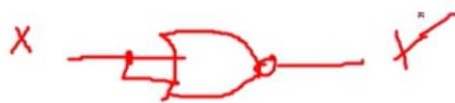
- NAND and NOR Equivalents of AND and OR Gates



→ AND와 OR을 만드는 다양한 방법

- 추가로 생각해 봐야 할 점

NOR 게이트로 인버터는 어떻게 만들까?



→ 똑같다

7.3 Design of Two-Level Circuits Using NAND and NOR Gates

- DeMorgan's laws

$$(X_1 + X_2 + \dots + X_n)' = X_1' X_2' \dots X_n'$$

NAND 게이트

$$(X_1 X_2 \dots X_n)' = X_1' + X_2' + \dots + X_n'$$

NOR 게이트

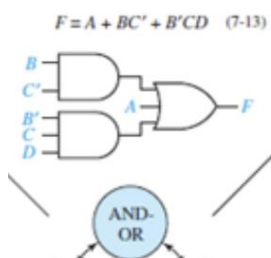
- 이를 이용하여 다양한 2-Level Structure를 구현 가능함

- Conversion of a sum-of-products to several other two-level forms

- Eight Basic Forms for Two-Level Circuits: 참고로 이런 내용이 있다는 것만 기억하자

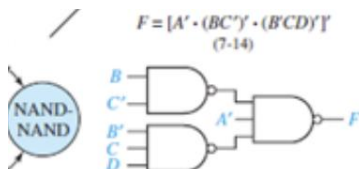
다음과 같은 구조가 있다고 하자

$$F = A + BC' + B'CD \text{ (Standard SOP Form, AND-OR Structure)}$$

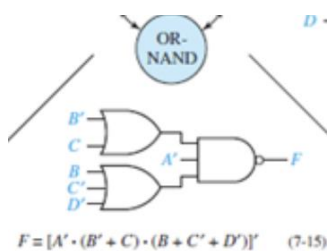


$$= [(A + BC' + B'CD)'] \text{ (표현의 complement의 complement 형태)}$$

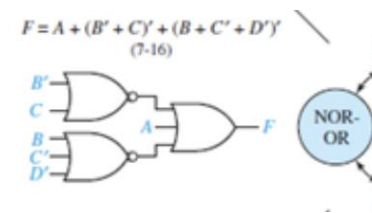
$$= [A'(BC')'(B'CD)'] \text{ (NAND-NAND Structure)}$$



$$= [A'(B' + C)(B + C' + D')] \text{ (OR-NAND 2-Level Structure)}$$

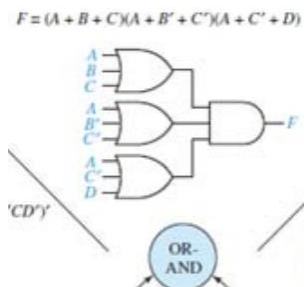


$$= A + (B' + C)' + (B + C' + D')' \text{ (NOR-OR 2-Level Structure)}$$



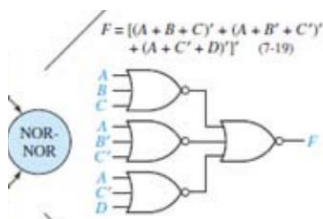
$$= \{[A + (B' + C)' + (B + C' + D')']'\}' \text{ (NOR-NOR-INVERT)}$$

$$= (A + B + C)(A + B' + C')(A + C' + D) \text{ (maxterm 구한 POS 형태, OR-AND Structure)}$$

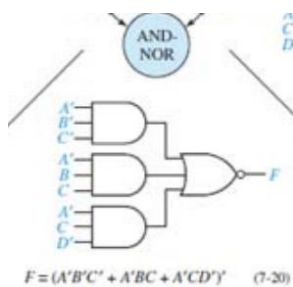


$$= \{[(A + B + C)(A + B' + C')(A + C' + D)]'\}' \text{ (표현의 complement의 complement 형태)}$$

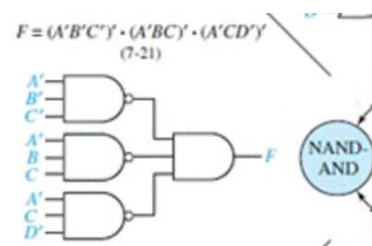
$$= [(A + B + C)' + (A + B' + C')' + (A + C' + D)]' \text{ (NOR-NOR Structure)}$$



$$= (A'B'C' + A'BC + A'CD')' \text{ (AND-NOR 2-Level Structure)}$$



$$= (A'B'C')(A'BC)(A'CD')' \text{ (NAND-AND 2-Level Structure)}$$



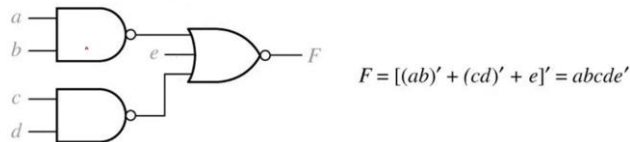
→ 여기서 주의해야 할 점은 NAND-NAND랑 NOR-NOR임!

NAND-NAND는 SOP로부터 Complement 2번 취하고 한번 전개하면서 얻어졌고

NOR-NOR는 POS로부터 Complement 2번 취하고 한번 전개하면서 얻어짐

- 기본적인 8개의 형태에 들어가지 않는 형태는 2레벨로 구현할 수 없다(degenerate)

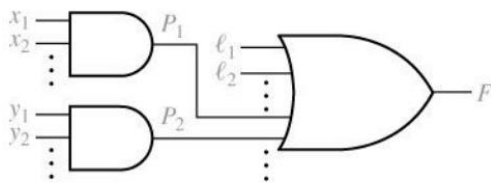
EX) NAND-NOR



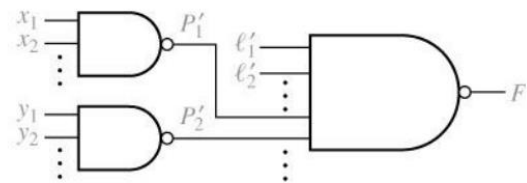
- AND-OR to NAND-NAND Transformation

$(l_1, l_2 \dots)$: literals $(P_1, P_2 \dots)$: product terms

$$F = l_1 + l_2 + \dots + P_1 + P_2 + \dots = \left(l_1' l_2' \dots P_1' P_2' \dots \right)'$$



(a) Before transformation

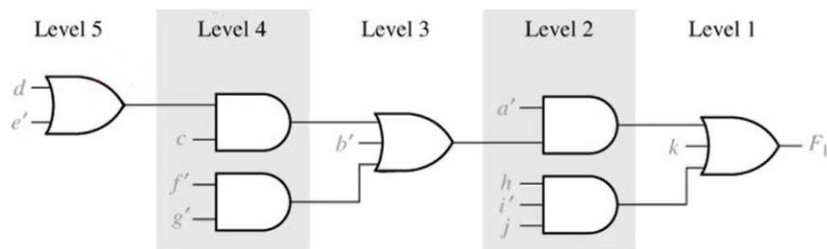


(b) After transformation

수식적인 표현. Complement를 2번 취해준다

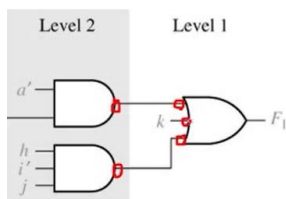
7.4 Design of Multi-Level NAND- and NOR-Gate Circuits

- Multi-Level NAND-Gate 만들기 절차
- 설명보다 과정을 통해 이해하는 것이 빠름
- 예시: $F_1 = a'[b' + c(d + e') + f'g'] + hi'j + k$

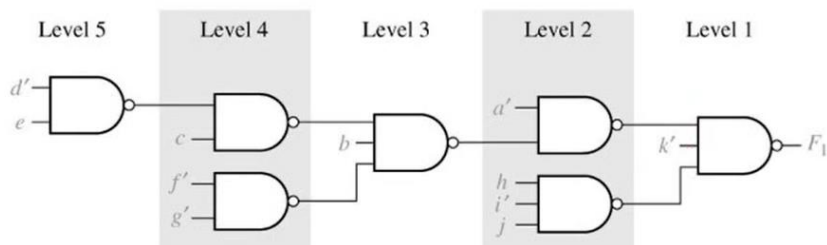


→ 이 멀티레벨 게이트를 NAND 게이트 만을 이용하여 구성하고 싶다고 하자.

여기서 짝수 레벨을 보자. 버블을 추가하고 상쇄시키기 위해 버블을 추가한다면



→ 이런 식으로



→ 그림과 같이 NAND만으로 변환이 가능하다

- 말로 과정을 정리하면

Simplify – 단순화한다

AND와 OR로 멀티 레벨 게이트를 구성한다

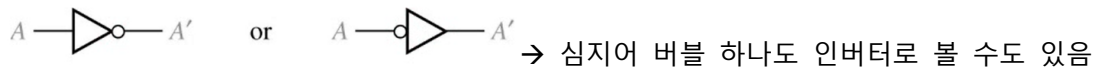
Output 게이트를 레벨 1로 한다

레벨 2, 4, 6로 옮겨가면서 NAND 게이트로 만든 다음에 interconnections를 변함이 없게 적절히 버블을 추가한다.

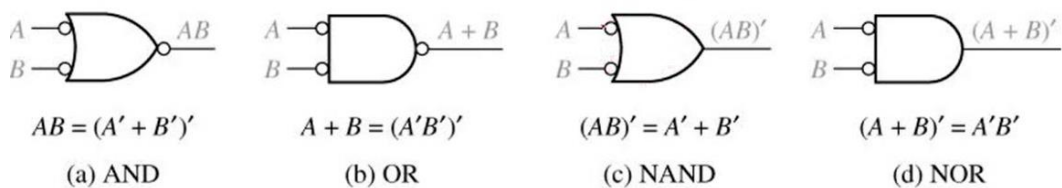
7.5 Circuit Conversion Using Alternative Gate Symbols

• Circuit Conversion Using Alternative Gate Symbols

- Inverter



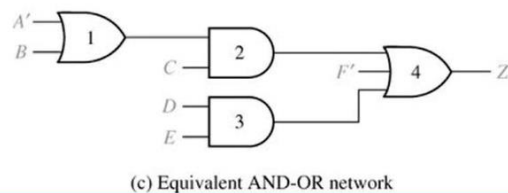
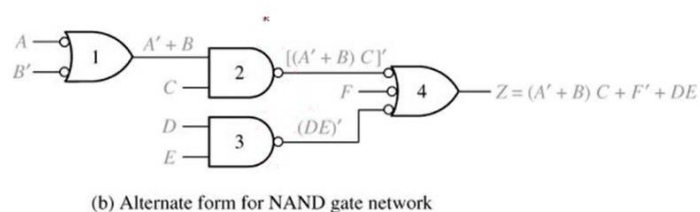
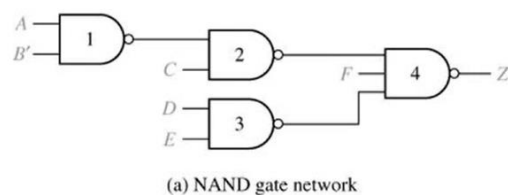
- Alternative Gate Symbols



→ OR에 버블 들어가면 NAND, AND에 버블 들어가면 NOR

- NAND Gate Circuit Conversion

최종적으로 NAND만으로 게이트를 어떻게 구성하는가?

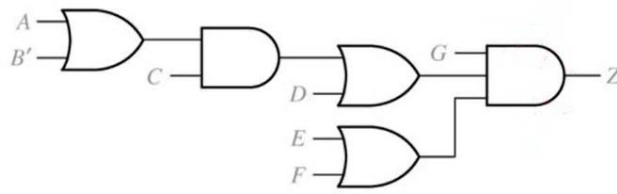


아래 → 위 순서로. 버블 추가하고 상쇄 위해 또 버블 추가

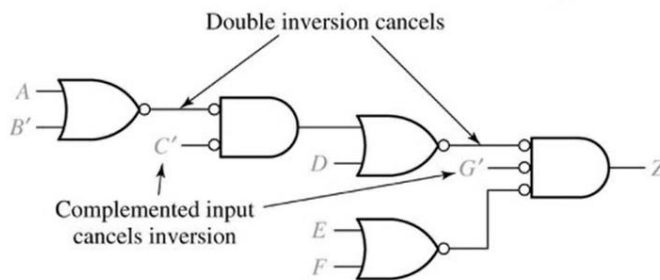
- Conversion to NOR Gates

NOR만으로 게이트를 어떻게 구성해야 하는가? → POS 형태로 출발해야 한다

즉 아웃풋 나오는 마지막이 AND여야 한다는 소리



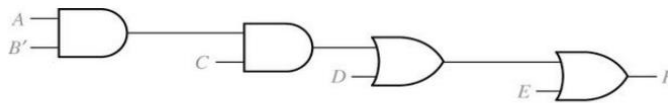
(a) Circuit with OR and AND gates



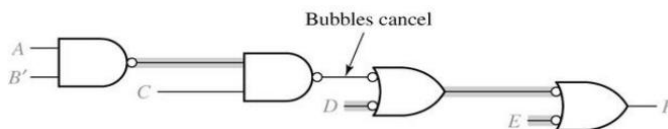
(b) Equivalent circuit with NOR gates

버블 추가하고 상쇄 위해 또 버블 추가

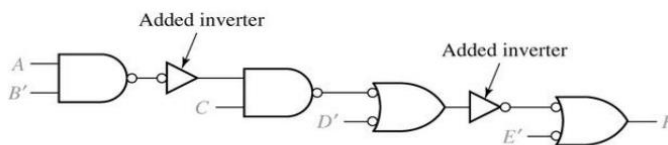
- Conversion of AND-OR Circuits to NAND Gates



(a) AND-OR network



(b) First step in NAND conversion



(c) Completed conversion

NAND만으로 구현하는 것. 좀 변형된 형태이긴 하다.

따라서 버블의 수를 맞추기 위해 인버터를 중간에 추가함

추가: 인버터도 NAND로 구현이 가능하다

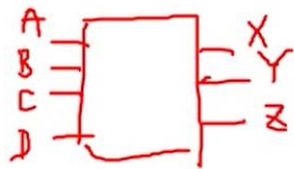


→ 이런 형태로

→ 결론: 임의의 구조라도 NAND랑 NOR만을 이용하여 구현할 수 있다

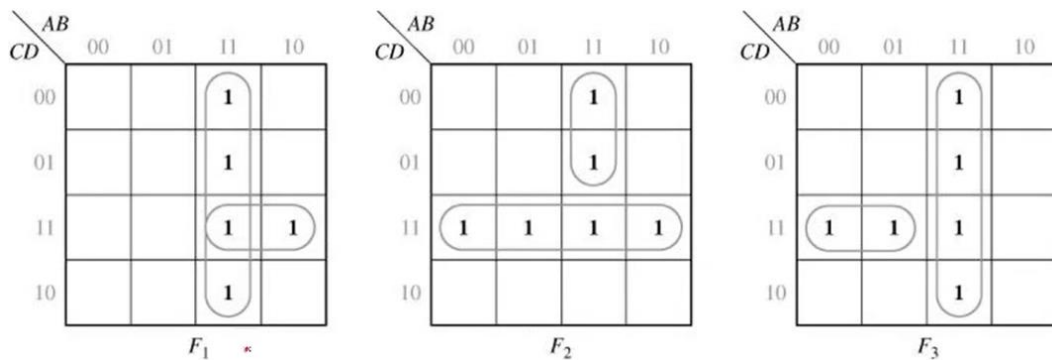
7.6 Design of Two-Level, Multiple-Output Circuits

- Multiple-Output Circuit
- 입력이 여러 개이고 출력이 여러 개인 회로
- Example: Design a circuit with four inputs and three outputs

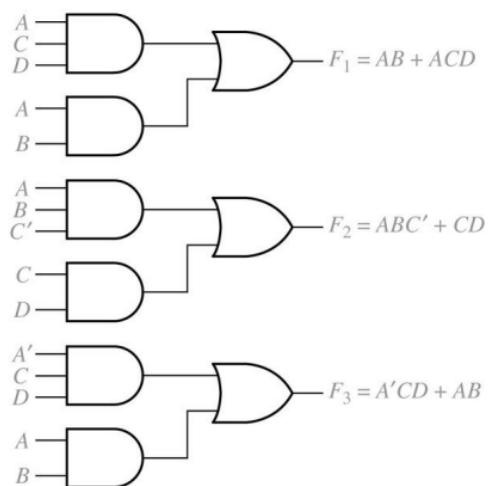


$$\begin{aligned}
 X &= F_1(A, B, C, D) = \sum m(11, 12, 13, 14, 15) \\
 Y &= F_2(A, B, C, D) = \sum m(3, 7, 11, 12, 13, 15) \\
 Z &= F_3(A, B, C, D) = \sum m(3, 7, 12, 13, 14, 15)
 \end{aligned}$$

Karnaugh Maps for Equations



Realization of Equations

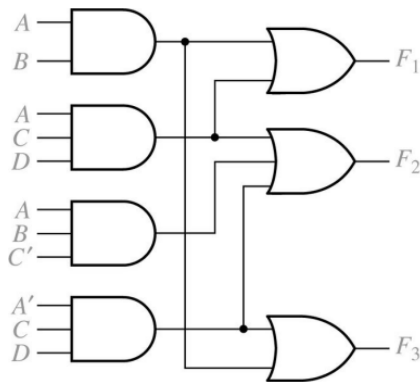


→ Multiple-Output인 경우, 각각을 최적화하는 것이 전체 회로 입장에선 최적일 아닐 수도 있다

위의 경우에선, CD는 $ACD + A'CD$ 와 같으므로 이 둘의 결합으로 대체될 수 있다

또한 AB가 중복되므로 하나를 두 번 사용하면 된다

Multiple-Output Realization of Equations



→ 2개의 게이트를 save: 최적의 Circuit

그러면 Multiple-Output Circuit의 경우 항상 cost 측면에선 이렇게 처리해야 하는가?

→ 꼭 그렇지만은 않다

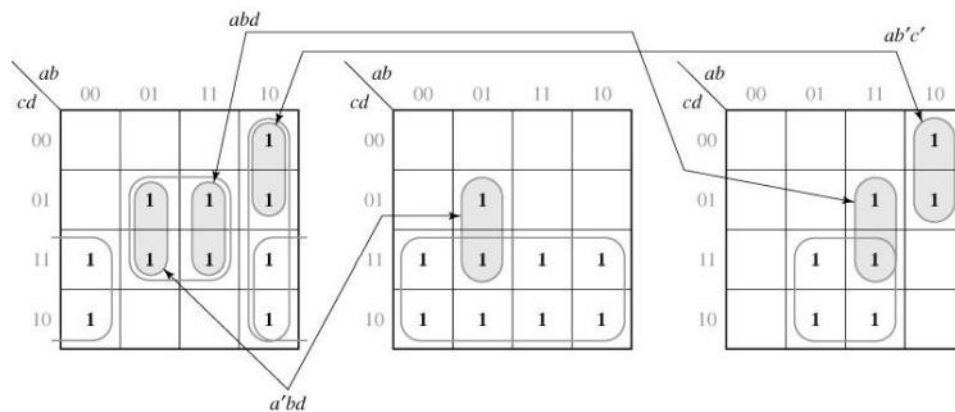
- Example: Design a multiple-output circuit with 4-inputs and 3-outputs

$$f_1 = \sum m(2,3,5,7,8,9,10,11,13,15)$$

$$f_2 = \sum m(2,3,5,6,7,10,11,14,15)$$

$$f_3 = \sum m(6,7,8,9,13,14,15)$$

Karnaugh Maps for Equations



Minimized Equations

$$f_1 = bd + b'c + ab'$$

$$f_2 = c + a'bd$$

$$f_3 = bc + ab'c' + \begin{cases} abd \\ or \\ ac'd \end{cases} \begin{matrix} 10 \text{ gates,} \\ 25 \text{ gate inputs} \end{matrix}$$

→ 카운팅에 문제 있는 듯함 확인 요망

The Minimal Solution

$$f_1 = \underline{a'bd} + \underline{abd} + \underline{ab'c'} + b'c$$

$$f_2 = c + \underline{a'bd}$$

$$f_3 = bc + \underline{ab'c'} + \underline{abd}$$

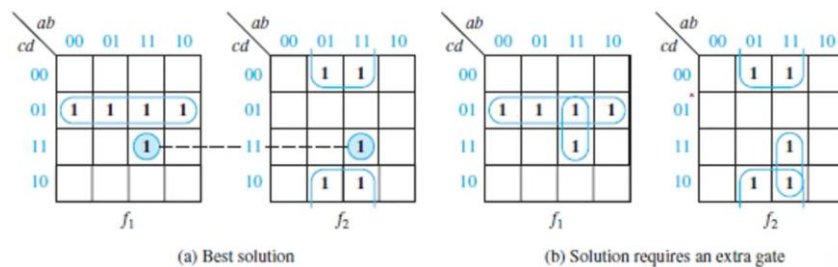
8 gates
22 gate inputs

→ 위의 카르노 맵에서 색칠한 부분

공통인 것을 가급적 많이 만들어서 입력과 게이트의 수를 줄인다

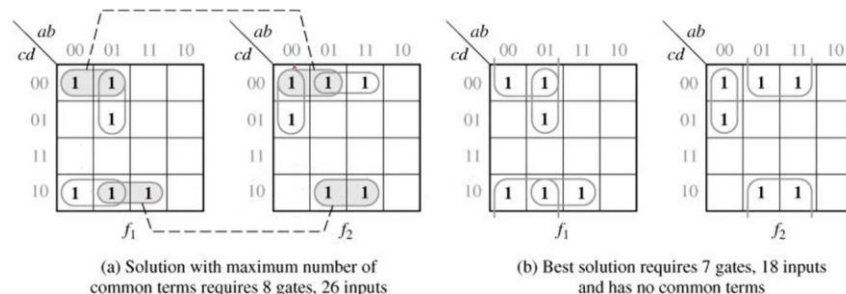
근데 이런 경우도 있지만, 공통인 경우를 가급적 많이 만들어서 하는 것이 무조건 좋은 것은 아니다. 꼭 그렇지만은 않은 것이 문제이다. 그렇지 않을 수도 있다

- Determination of Essential Prime Implicants for Multiple-Output Realization



(a): 공통인 부분을 만들어서 cost를 감소

(b): 각각의 최선의 그루핑. 최적 아님



(a): 공통인 요소를 많이 만들었음. 근데? 오히려 각각을 최소로 했을 때 보다 더 많이 들

(b): 각각의 최선의 그루핑. 근데 최적임

→ 공통의 요소를 무지성으로 막 만든다고 좋은 것이 아님

- 그럼 어떻게 해야 하는가? - 개략적인 cost를 감소시키는 가이드라인

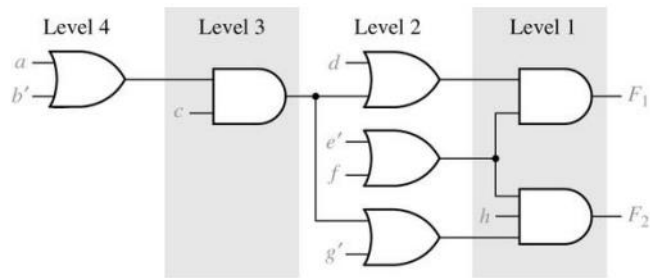
1. 어느 한 쪽에 있는 EPI를 취함. F1이든, F2이든 일단 취함
2. EPI를 다 취한 후에, 각각의 멀티 아웃풋이 있다면, 멀티 아웃풋에 각각의 function에 대한 EPI는 다 표현식에 넣은 다음에, EPI가 아닌 나머지를 최대한 간소화시키는 식으로 함

7.7 Multiple-Output NAND and NOR Circuits

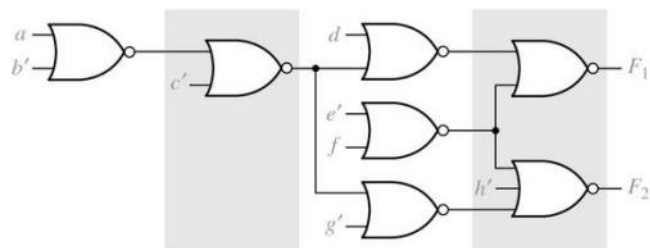
- Multi-level Circuits Conversion to NOR Gates

$$F_1 = [(a + b')c + d](e' + f)$$

$$F_2 = [(a + b')c + g'](e' + f)h$$



(a) Network of AND and OR gates



(b) NOR network

→ 멀티 아웃풋 이어도 NOR 만으로 구성 가능함