2장. 조합논리회로

오늘의 교훈

미래는 준비하는자의 것이다

부울 대수

- 부울대수의 가설과 정리 표 (*표 2-3)
  - 쌍대성 (duality) : . $\Longleftrightarrow$ +, 0 $\Longleftrightarrow$ 1
  - 정리의 증명방법
    - 가설과 정리 이용
    - 진리표 이용
    - Venn-diagram 이용
    - 예) $x + xy = x$
      - 가설이용

$$x + xy = x.1 + xy \text{ (가설 항등원)}$$
$$= x(1 + y) \text{ (가설 배분법칙)}$$
$$= x(y + 1) \text{ (가설 교환법칙)}$$
$$= x.1 \text{ (정리 2(a))}$$
$$= x \text{ (가설 항등원)}$$

부울 대수 (Cont'd)

- 진리표 이용

x	y	xy	x+xy
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

- Venn-diagram 이용

부울 대수 (Cont'd)

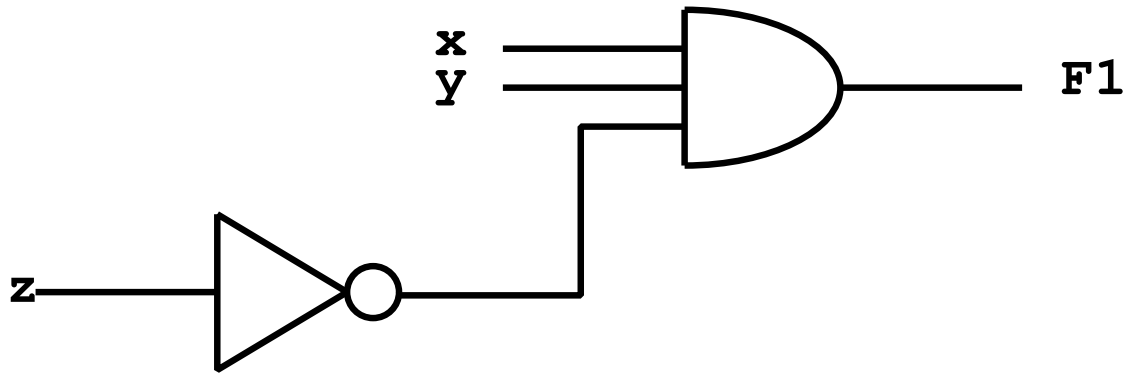
- two-valued boolean algebra

x	y	x.y	x	y	x+y	x	x'
0	0	0	0	0	0	0	1
0	1	0	0	1	1	1	0
1	0	0	1	0	1		
1	1	1	1	1	1		

  * . : AND 와 일치
  * + : OR 와 일치
  * ' : NOT 과 일치

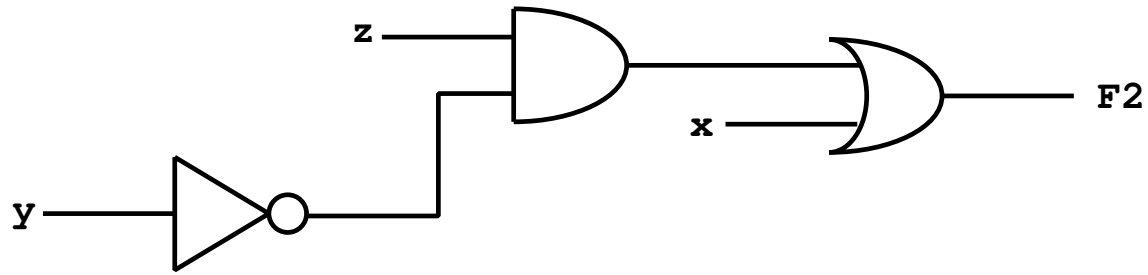
부울 함수 (Boolean functions)

- 2진 변수 : 0 또는 1을 갖는 변수
- 연산자 : AND, OR, NOT, (), =
- 연산자 우선순위 →(), NOT, AND, OR, =
- 예) F1=xyz'
  - x = 1, y = 1, z = 0 일때만 F1=1이된다.
  - 게이트로 구현



부울 함수 (Boolean functions) (Cont'd)

- 예) F2=x+y'z
  - x = 1 이거나 y = 0 이고 z = 1 일때 만 F2=1이된다
  - 게이트 구현



- 디지털 논리게이트에 따른 타이밍 다이어그램 (*그림 2-1)
- 2입력 이상의 게이트 (* 그림 2-2)
- 부울함수의 처리: 가장 간단한 함수표현식을 찾는 문제
- 함수표현식이 간소화 됨에 따라 입력 수와 게이트 수가 축소됨 (*그림 2-4)

부울 함수 (Boolean functions) (Cont'd)

- 대수적 처리에의한 간소화 : 정리와 가설을 이용
  - 예) x + x'y = (x + x') (x + y) = 1 . (x + y) = (x + y) . 1 = (x + y)
- 일반화된 드모르간 법칙
  - (A + B + C + ... + F)' = A'B'C'...F'
  - (A B C ... F)' = (A' + B' + C' + ...+ F')
- 부울함수의 보수 (부정) : 쌍대를 취한 뒤 변수에 부정을 취하면 됨
  - 예) F1 = x(y'z' + yz) 일 때 Fl' = x' + (y + z)(y' + z')

정형과 표준형 (canonical and standard forms)

- 민텀 (minterm) or (standard product) : n bit 를 2^n개의 조합으로 표현할 때 각 항을 AND항으로 나타내어 결과가 1이 되게함
- 맥스텀 (maxterm) or (standard sum) : n bit 를 2^n개의 조합으로 표현할 때 각 항을 OR항으로 나타내어 결과가 0이 되게함

정형과 표준형 (canonical and standard forms) (Cont'd)

- 민텀과 맥스텀의 표현

			민 텀		맥스텀	
x	y	z	항	표시	항	표시
0	0	0	x'y'z'	m_0	x+y+z	M_0
0	0	1	x'y'z	m_1	x+y+z'	M_1
0	1	0	x'yz'	m_2	x+y'+z	M_2
0	1	1	x'yz	m_3	x+y'+z'	M_3
1	0	0	xy'z'	m_4	x'+y+z	M_4
1	0	1	xy'z	m_5	x'+y+z'	M_5
1	1	0	xyz'	m_6	x'+y'+z	M_6
1	1	1	xyz	m_7	x'+y'+z'	M_7

정형과 표준형 (canonical and standard forms) (Cont'd)

- 모든 부울함수는 민텀의합 이나 맥스텀의 곱방식으로 나타낼수 있다 →정형 (canonical form)
  - minterm의 합 : 어떤 함수의 1이되게 하는것들을 OR로 모아놓음
  - maxterm의 곱 : 어떤 함수의 0이되게 하는것들을 AND 로 모아놓음
  - 예)

```
x y F1 F2
---------
0  0 0  0
0  1 1  0
1  0 1  0
1  1 1  1
```

    * F1 을 minterm 으로 : F1 = m1 + m2 + m3 = $\sum$ (1,2,3) = x'y + xy' + xy = x(y'+y) + x'y = x + x'y = (x+x')(x+y) = (x+y) = M0 = $\prod$ (0)
    * F1 을 maxterm 으로 : F1 = M0 M1 M2 = $\prod$ (0, 1, 2) = (x+y)(x+y')(x'+y) =(x+yy')(x'+y) = xx'+xy = xy = m3 = $\sum$ (3)

정형과 표준형 (canonical and standard forms) (Cont'd)

  - 예) F = A + B'C 를 민텀의 합으로 나타내라.
    * 진리표 이용
  - 예) F = xy + x'z 를 맥스텀의 곱으로 나타내라.

- 표준형 (standard forms) : 함수를 하나 또는 그 이상의 변수로 나타냄
  - 곱의 합형 : 1을 나타내기 위해 변수들이 곱(AND) 으로 나타내어지고 이러한 항목이 합(OR)의 형태로 묶임
  - 합의 곱형 : 0을 나타내기 위해 변수들이 합(OR)로 나타내어지고 이러한 항목이 곱(AND)의 형태로 묶임
  - 비표준형 : 곱의합형 과 합의 곱형이 혼재 →표준형으로 고칠수 있음
  - 예) F = (AB + CD) (A'B' + C'D') →F = A'B'CD + ABC'D'
  - 3단계와 2단계 구현 (* 그림 2-6)

정형과 표준형 (canonical and standard forms) (Cont'd)

- 기타 논리 연산 : n 개의 2진 변수에 대해 2^n 개의 함수 존재
  - AND, OR, NOT, XOR (exclusive-OR) 가 중요
  - XOR : 다르면 1
  - XNOR : 같으면 1 (equivalence)
  - NAND : NOT + AND
  - NOR : NOT + OR

맵 간략화

- 대수 식으로 간략화 하는 것은 너무 복잡
- 카르노 맵 이용
- 2변수 카르노 맵

x \ y	0	1
0	x′y′(m0)	x′y(m1)
1	xy′(m2)	xy(m3)

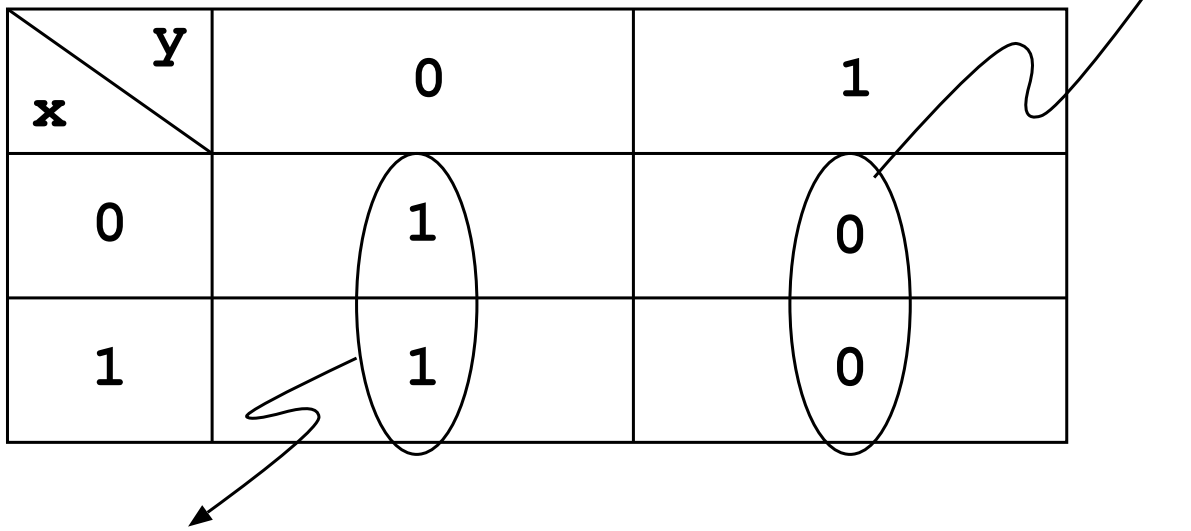
맵 간략화 (Cont'd)

- 함수 간략화의 예

x	y	F1
0	0	1
0	1	0
1	0	1
1	1	0

맵 간략화 (Cont'd)

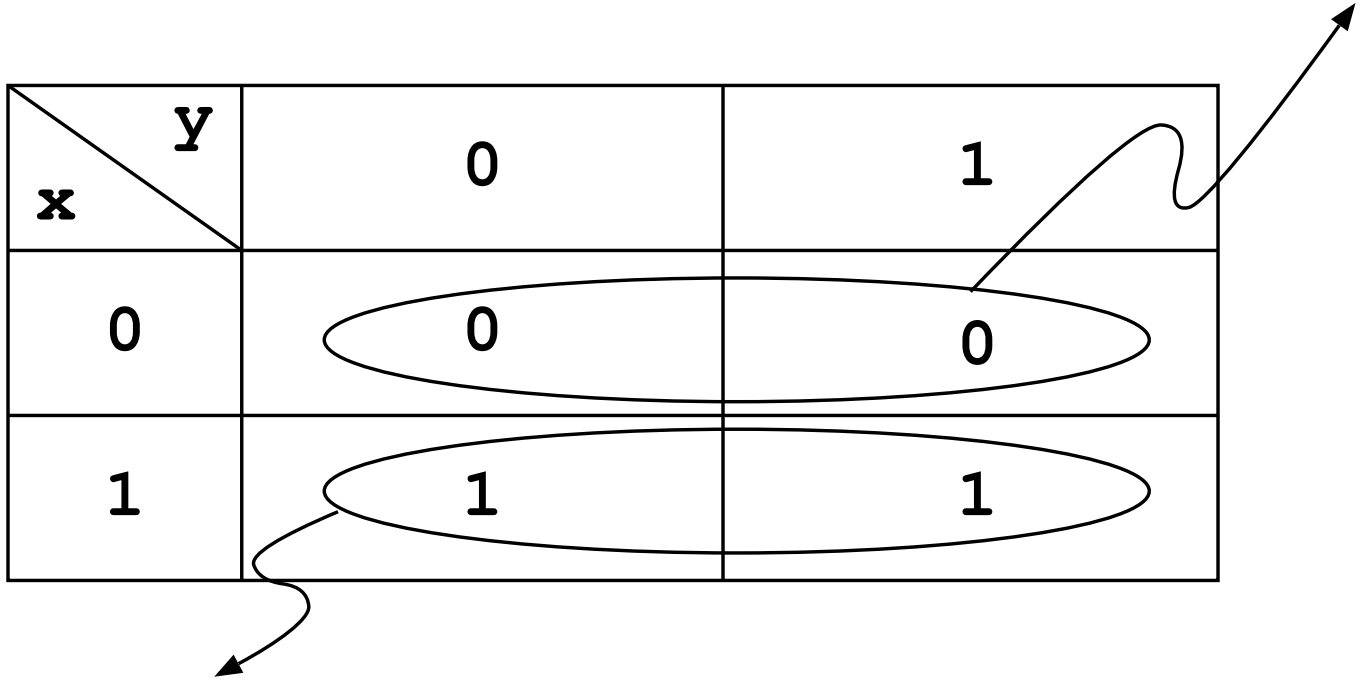
product of maxterms : (x+y′)(x′+y′) = (x.x′ +y′) = y′



sum of minterms : x′y′+xy′ = (x′ + x)y′ = y′

맵 간략화 (Cont'd)

product of maxterms : (x+y)(x+y′) = x+y.y′ = x



sum of minterms : xy′+xy = x(y′+y) = x

- 한변수에 대하여 0과 1에 걸쳐있는 변수는 간략화 된다.

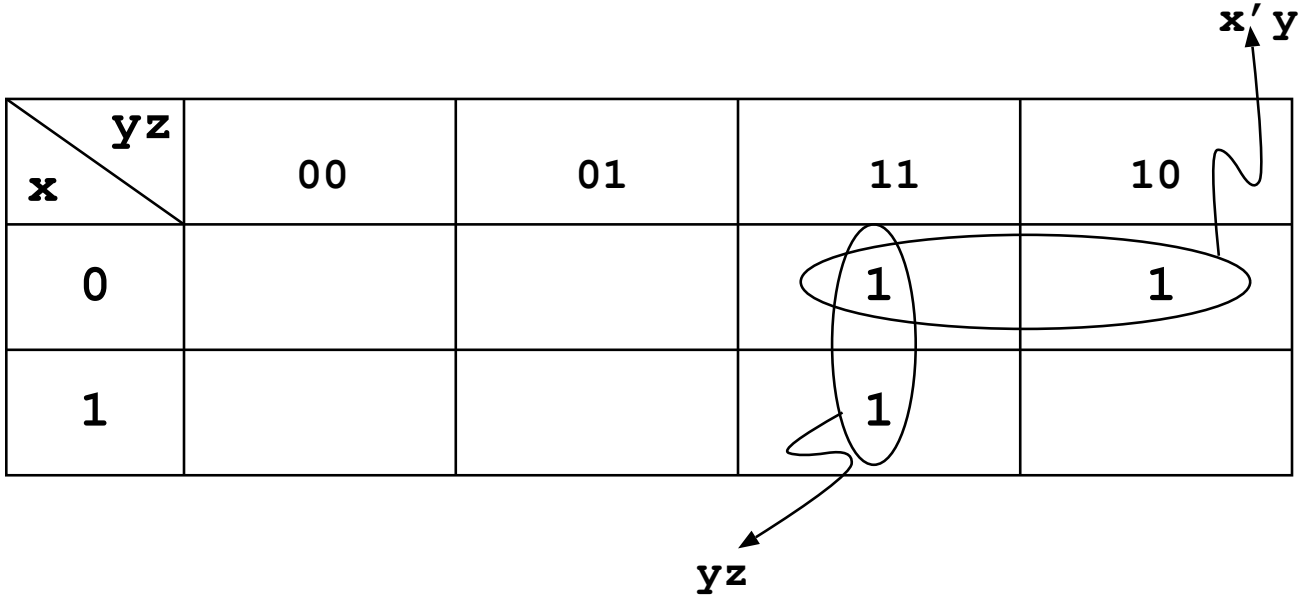
맵 간략화 (Cont'd)

- 3변수 카르노 맵
  - 3변수 카르노 맵부터는 두 개의 변수 값에 대하여 나타낼 때 그레이 코드 순서로 배열한다. 이는 그레이 코드가 인접한 변수에서 오직 하나의 bit 만이 다르기 때문이다. 이는 간략화를 쉽게 한다.

x \ yz	00	01	11	10
0	x′y′z′(m0)	x′y′z(m1)	x′yz(m3)	x′yz′(m2)
1	xy′z′(m4)	xy′z(m5)	xyz(m7)	xyz′(m6)

맵 간략화 (Cont'd)

- 예) F = x'yz' + x'yz + xyz

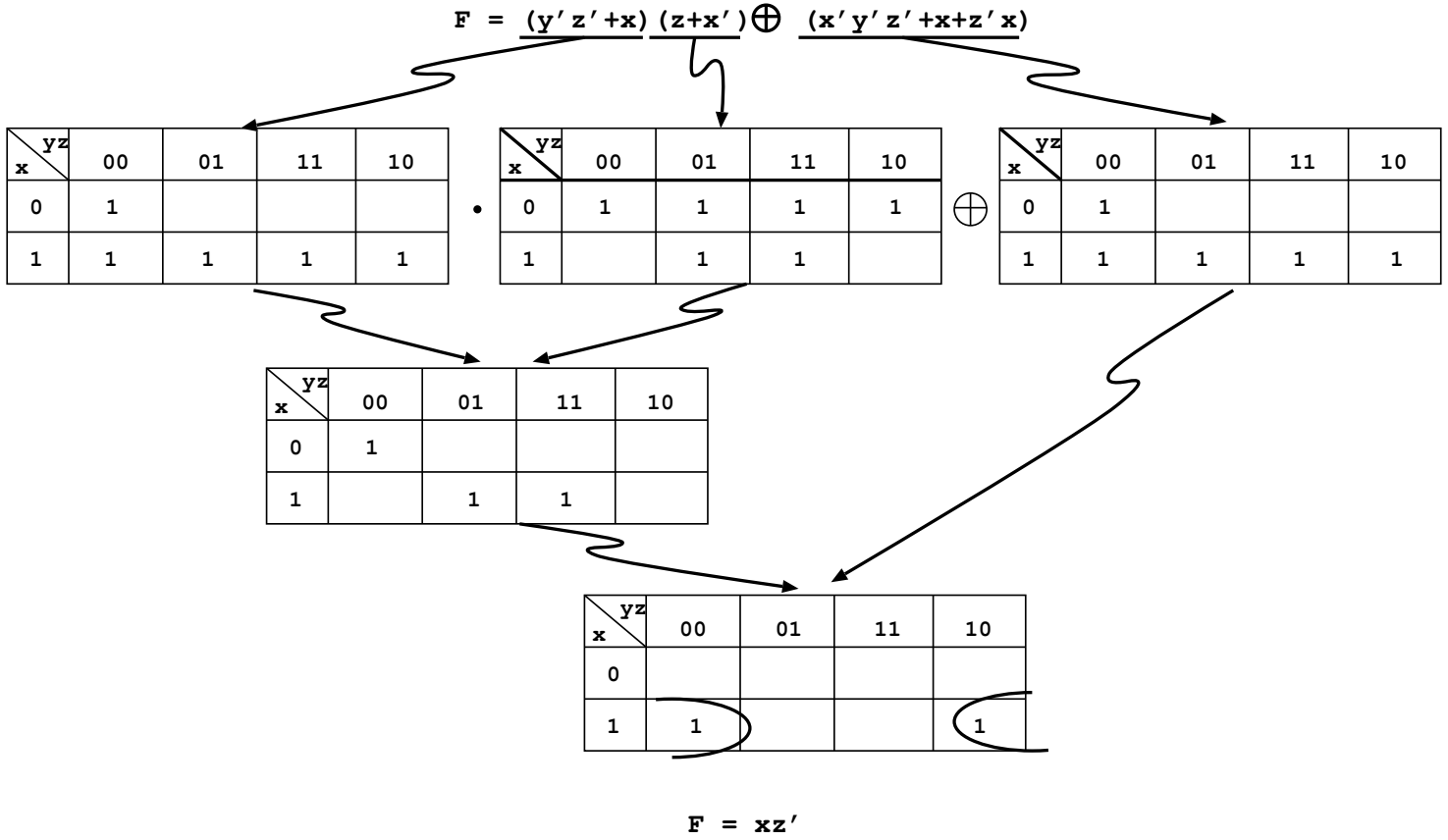


- 결국 F = x'y + yz
- 예제 2-3, 2-4
- 인접한 사각형을 나타내기 위한 평면도 (*그림 2-12)

맵 간략화 (Cont'd)

– 비표준형일 때 : 각각을 카르노 맵으로 그리고 연산을 카르노 맵상에서 하여 결과 맵으로 간략화

∗ 예)



맵 간략화 (Cont'd)

- 4변수 카르노 맵

wx \ yz	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

  - 인접선을 그리기위한 평면도 (*그림 2-18)
  - 예제 2-5, 2-6

맵 간략화 (Cont'd)

- 5변수 카르노 맵

AB \ CDE	000	001	011	010	110	111	101	100
00	0	1	3	2	6	7	5	4
01	8	9	11	10	14	15	13	12
11	24	25	27	26	30	31	29	28
10	16	17	19	18	22	23	21	20

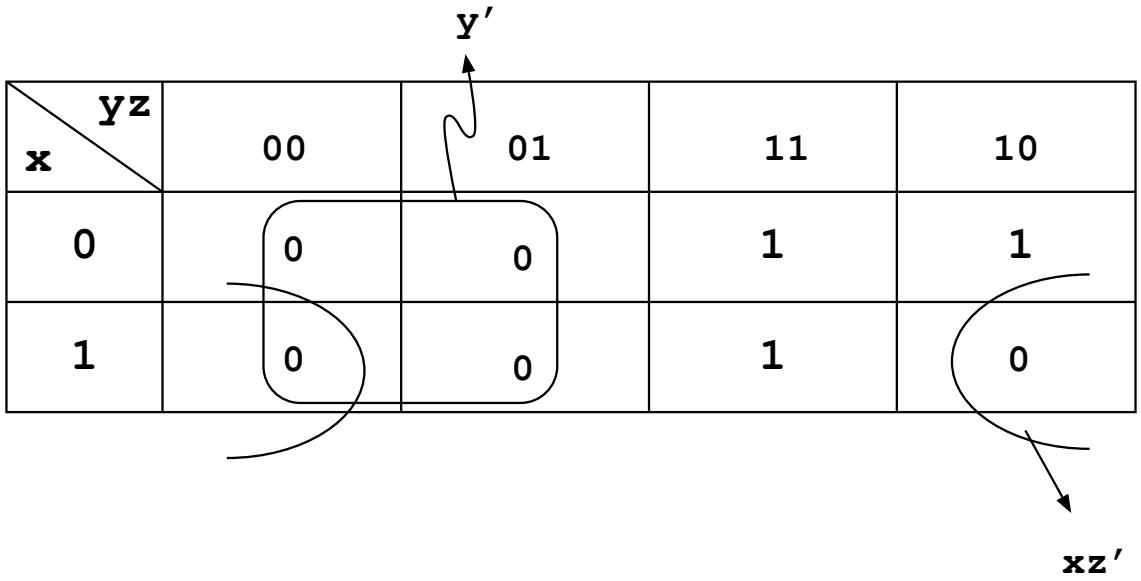
  - 두줄 부분을 접었을 때 마주치는 것도 간략화 됨

맵 간략화 (Cont'd)

- 6변수 카르노 맵
  - 두줄부분 접어서 인접한 것도 간략화됨
  - 6변수 이상은 카르노 맵으로 힘듦 (컴퓨터 이용)
- 카르노 맵과 변수
  - 맵 칸의 반이 1을 차지 $\rightarrow$변수 1개
  - 맵 칸의 이 1을 차지 $\rightarrow$변수 2개

맵 간략화 (Cont'd)

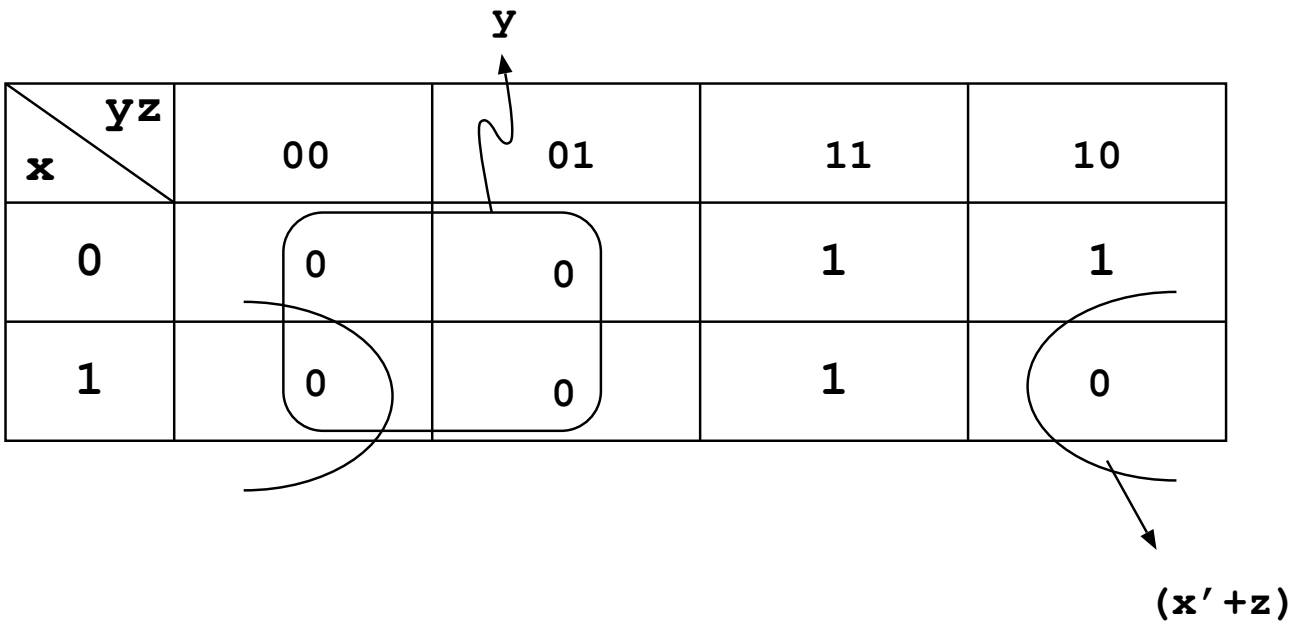
- 합의 곱으로 된 부울함수의 간략화
  - 방법 1
    - 0 을 곱의 합형으로 표시 (F')
    - (F')' : 드모르간 법칙 적용



    - F' = y'+xz'
    - F = (F')' = (y'+xz')' = y (x'+z)

맵 간략화 (Cont'd)

- 방법 2
  - 0 을 합의 곱형으로 표시



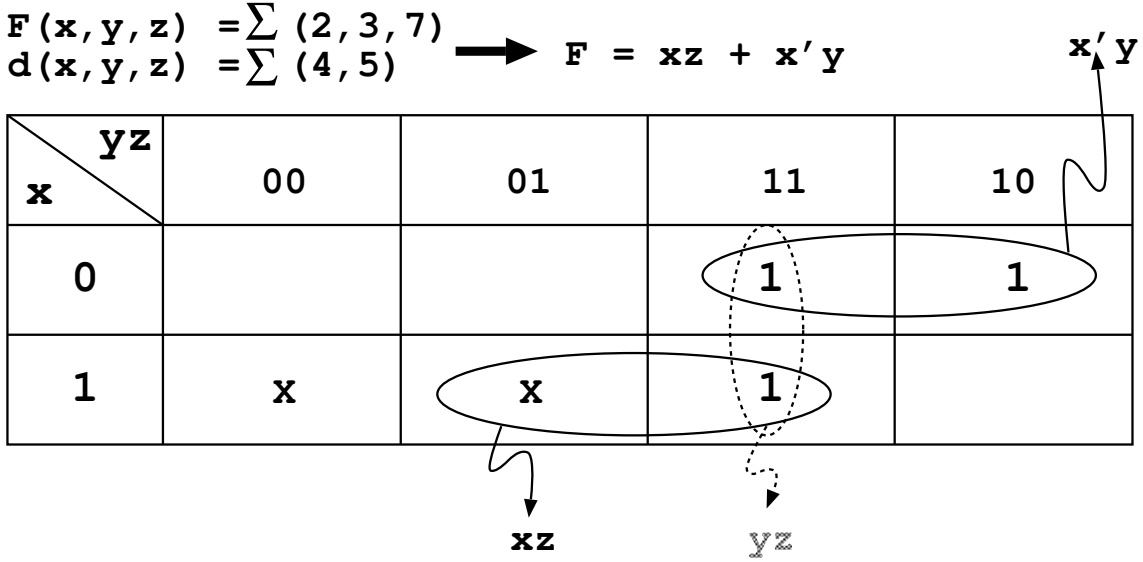
  - F = y (x'+z)
  - 예) : 예제 2-8

맵 조작

- 필요성
  - 모든 최소항 (minterm) 이 간략화 되었는지 확인 필요
  - 중복된 항을 피하여 간략화 되었는지 확인 필요
- 필수주항
  - 항 (implicant)
    - 어떤 함수가 minterm 으로 표현되었다면 표현된 모든 minterm 을 항이라고함
  - 주항 (prime implicant)
    - 두개 이상이 묶여서 하나이상의 문자가 사라진 항
  - 필수주항 (essential implicant)
    - 주항중 다른 주항에 포함되지 않는 minterm 을 가진주항을 말함
  - 예) 2-7

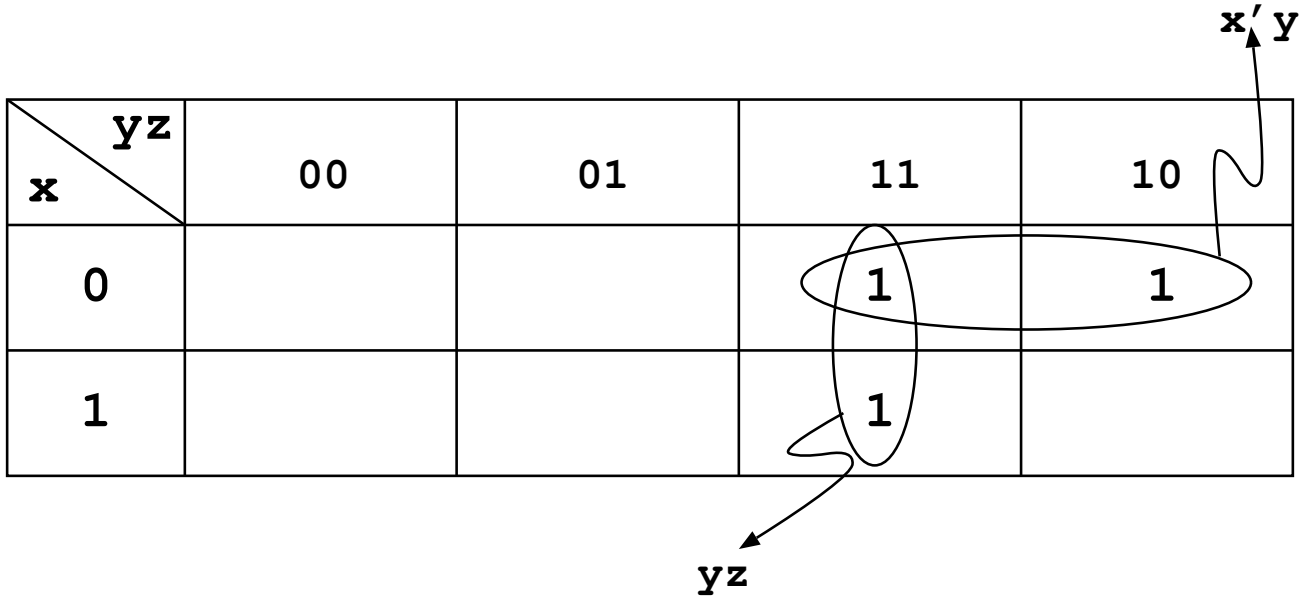
Don't care condition (무정의 조건)

- 입력측 조합에서 전혀 발생될 수 없는 조합에 대한 출력 (X 로 표시)
- 입력측 조합이 발생하나 출력은 고려하지 않을 때
- 간략화시 0, 1 아무것으로 놓아도 상관없다.



NAND 와 NOR 로 설계구현

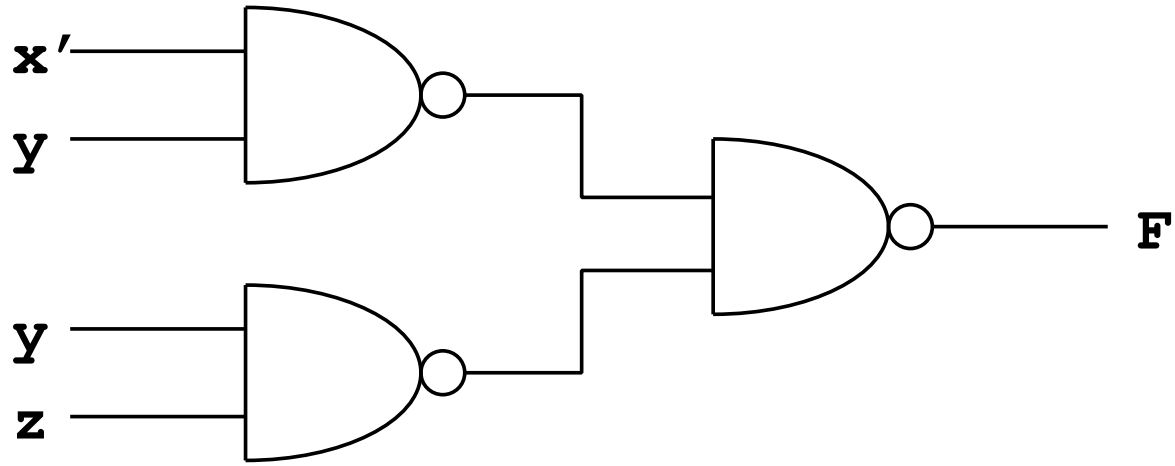
- 디지탈 논리 게이트 (*그림 2-26)
- NAND 와 NOR 게이트가 제작하기 용이
- NAND 게이트로 구현
  - 방법 1 : 곱의 합형으로 표시후 2중 부정을 취함
  - 예)



NAND 와 NOR 로 설계구현 (Cont'd)

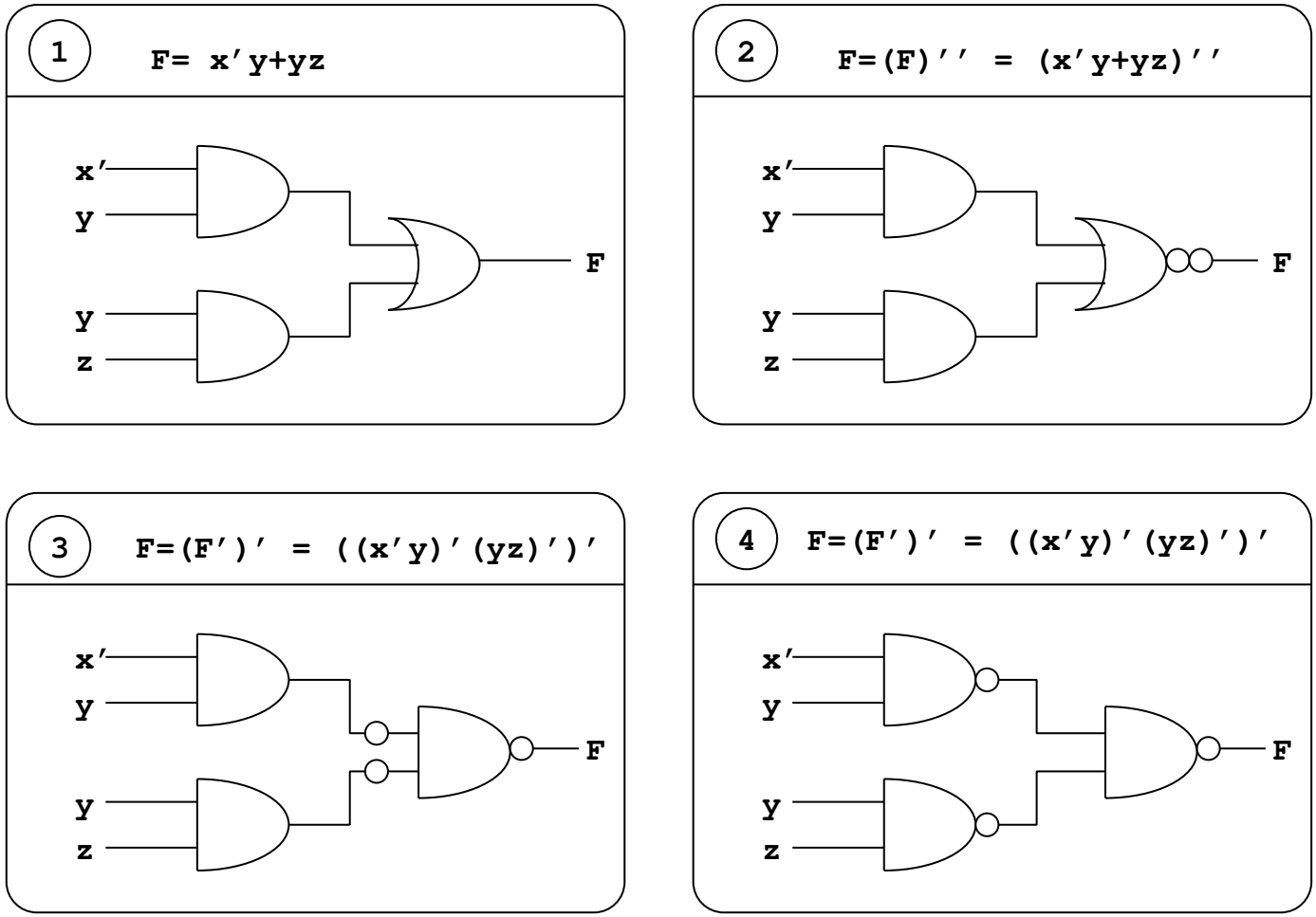
$F = x'y + yz$

$F = (F')' = ((x'y)'(yz)')'$



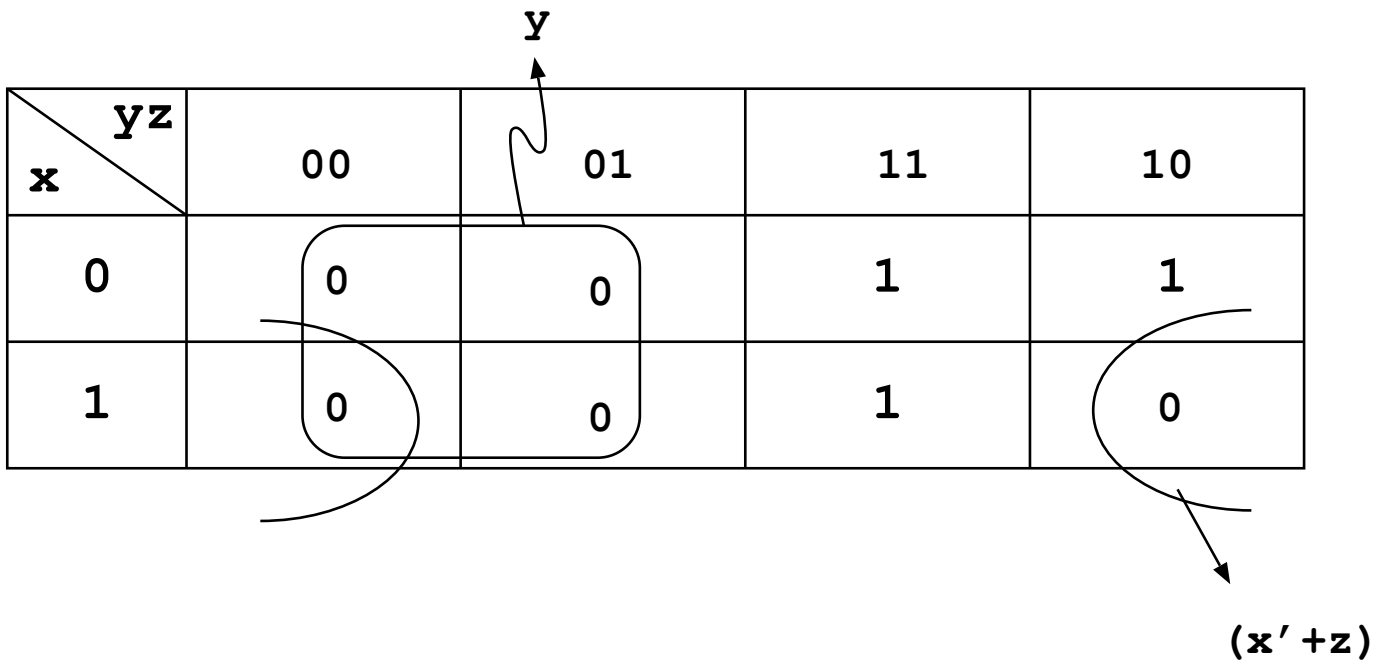
NAND 와 NOR 로 설계구현 (Cont'd)

– 방법 2 : 곱의 합형으로 표시후 OR 앞에 2중 부정을 취함



NAND 와 NOR 로 설계구현 (Cont'd)

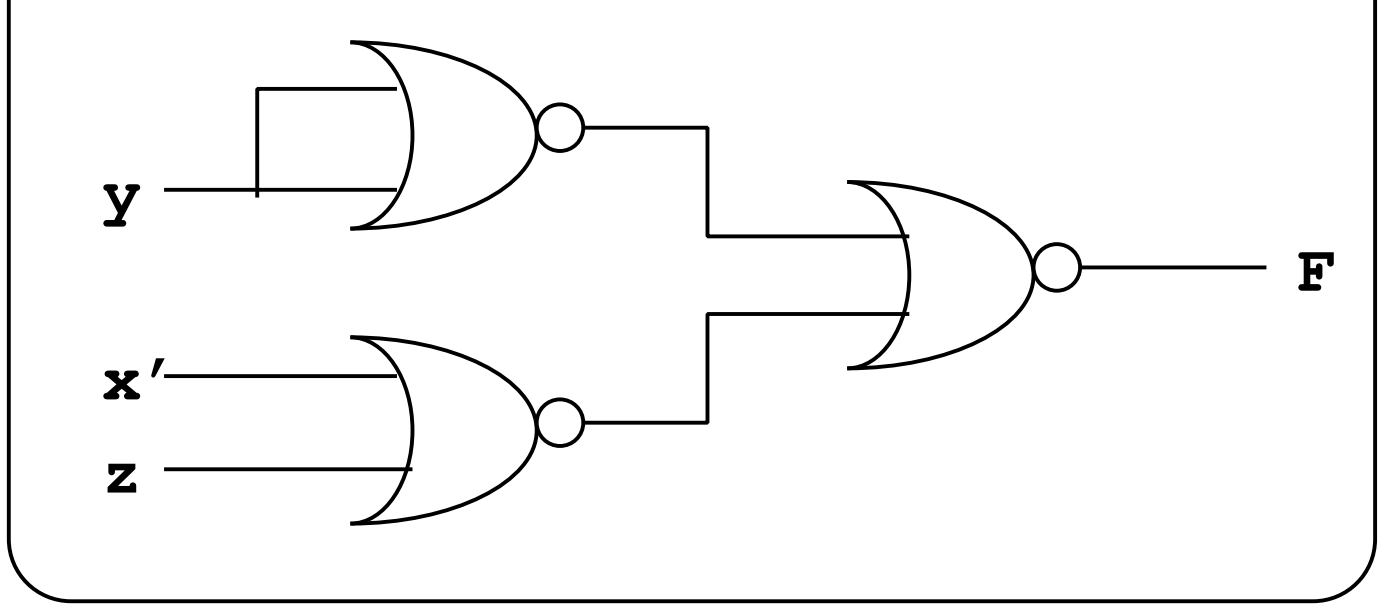
- NOR 게이트로 구현
  - 방법 1 : 합의 곱형으로 표시후 2중 부정을 취함
  - 예)



NAND 와 NOR 로 설계구현 (Cont'd)

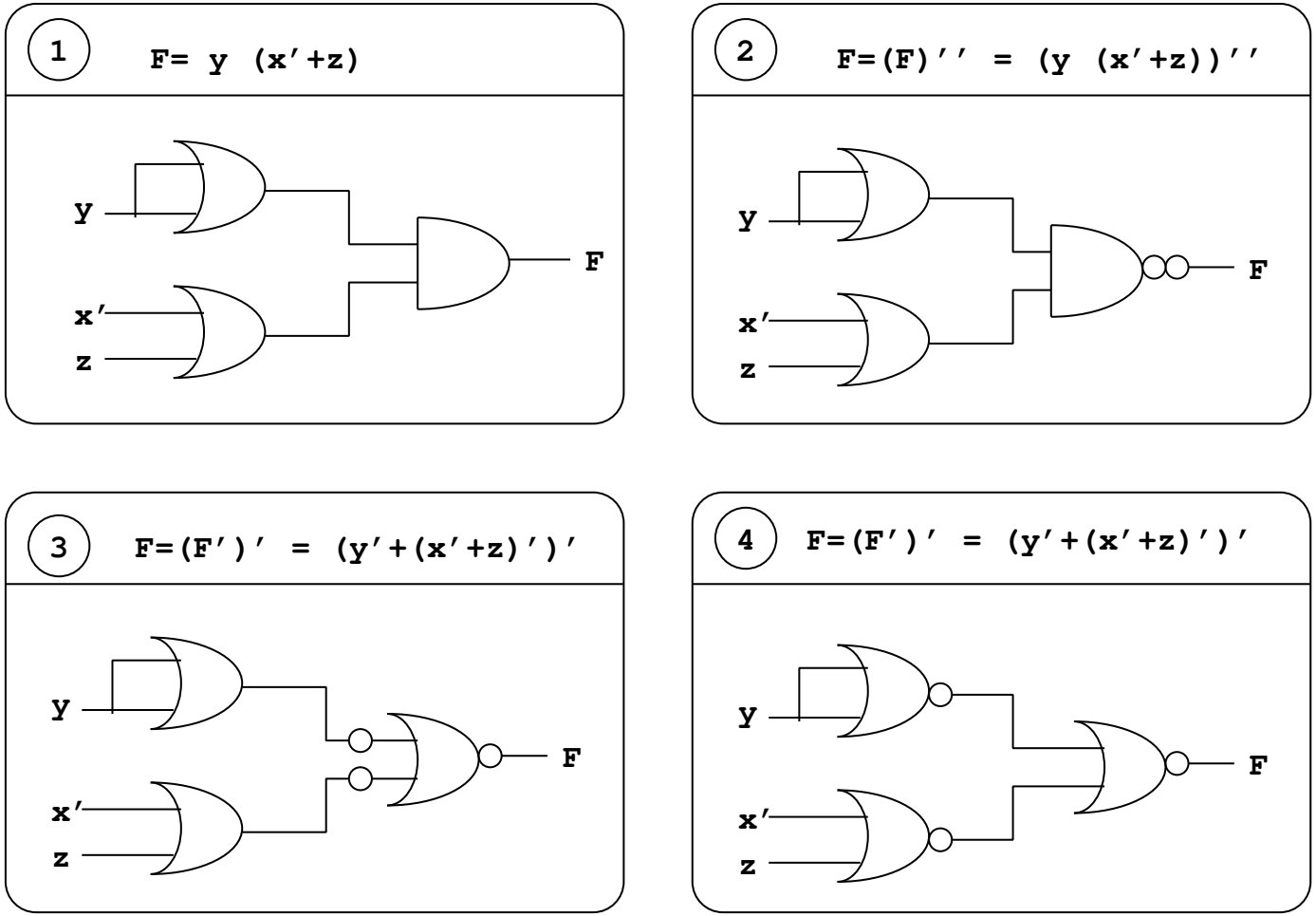
$F = y\,(x' + z)$

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



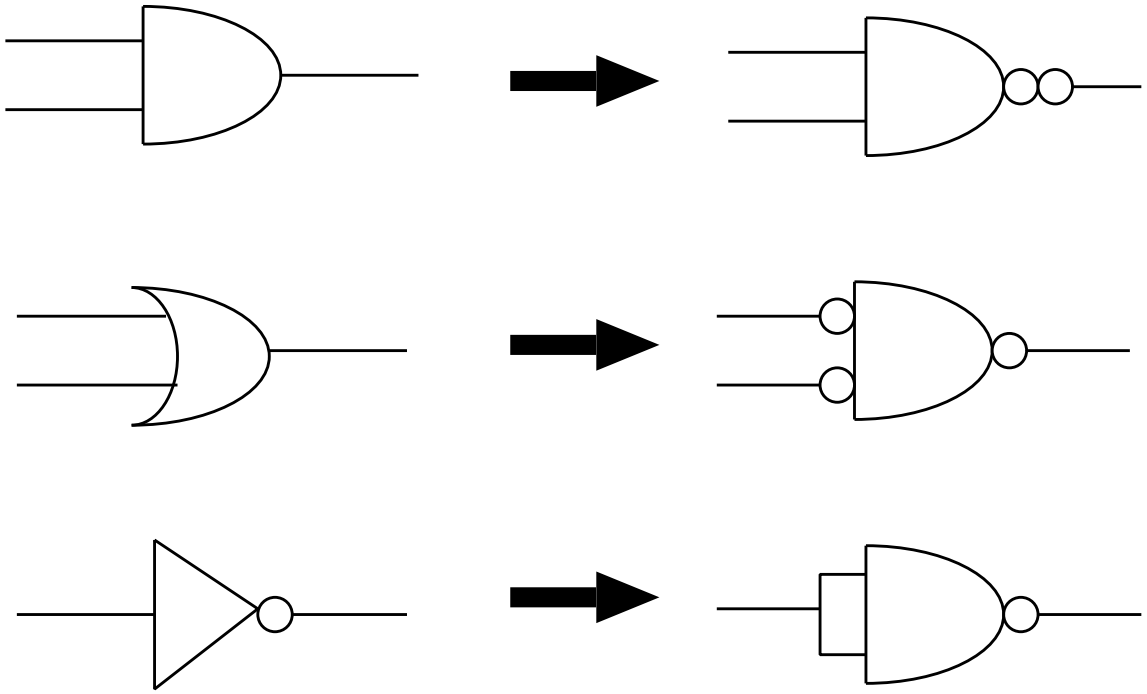
NAND 와 NOR 로 설계구현 (Cont'd)

– 방법 2 : 합의 곱형으로 표시후 AND 앞에 2중 부정을 취함



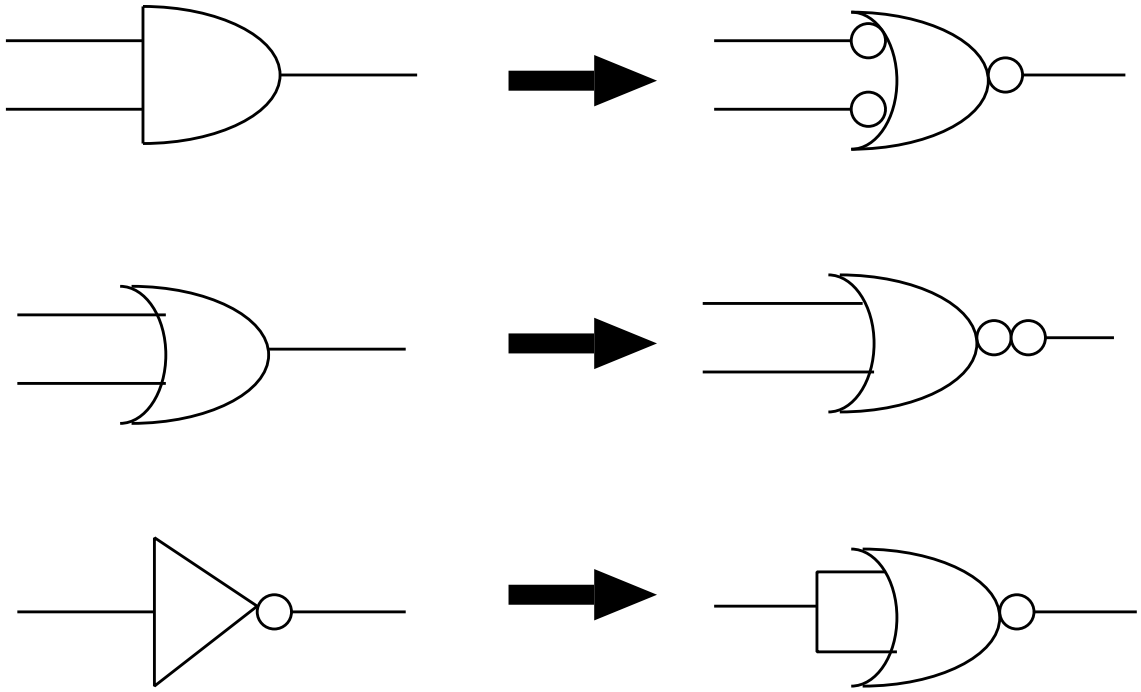
NAND 와 NOR 로 설계구현 (Cont'd)

- 등가회로를 이용한 구현
  - NAND 회로 구현시
    - AB →((AB)')'
    - A+B →(A'B')'
    - A' →A'



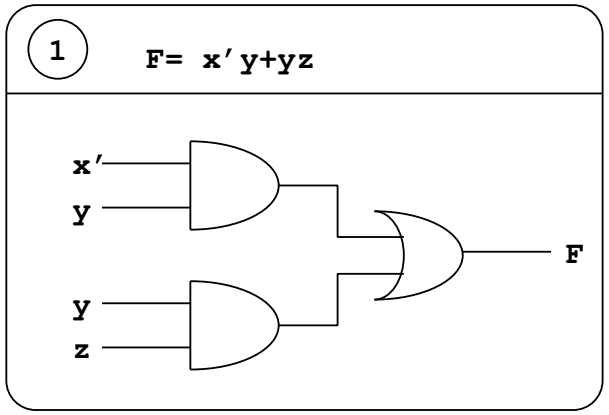
NAND 와 NOR 로 설계구현 (Cont'd)

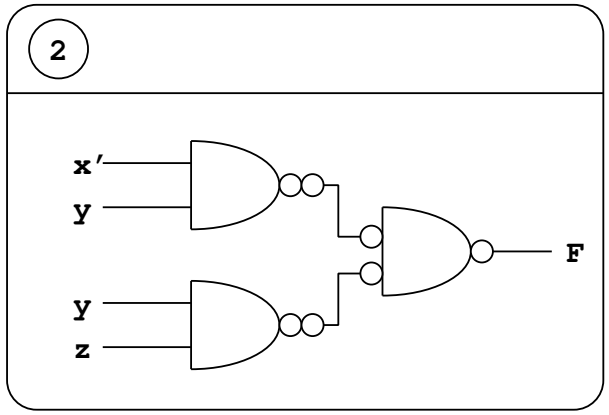
- NOR 회로 구현시
  - AB →(A'+B')'
  - A+B →((A+B)')'
  - A' →A'

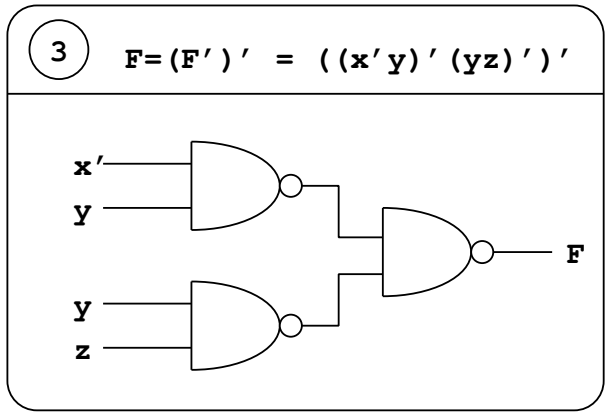


NAND 와 NOR 로 설계구현 (Cont'd)

- 위와 같이 등가회로로 나타내고 2중부정은 생략한다.
- NAND 회로로 바꾸는 예

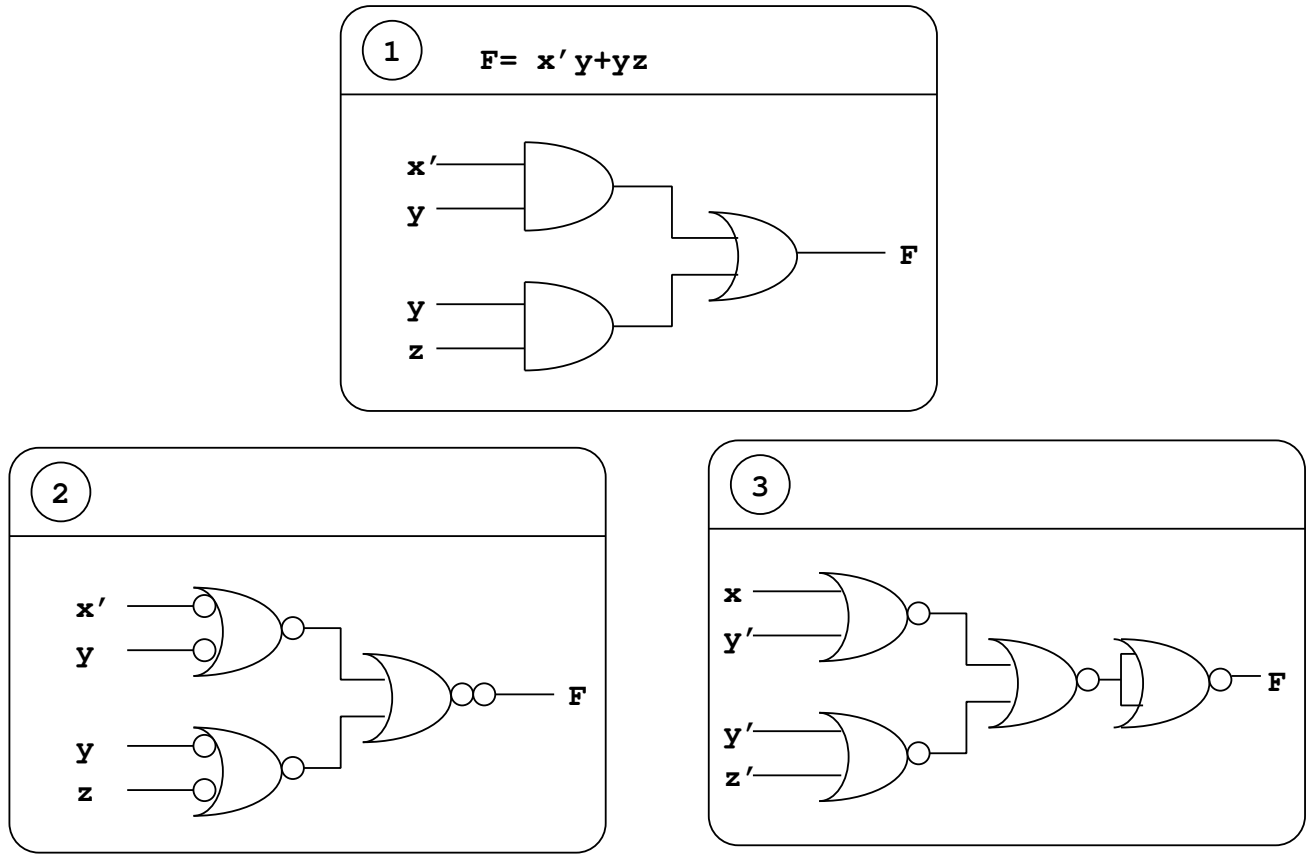






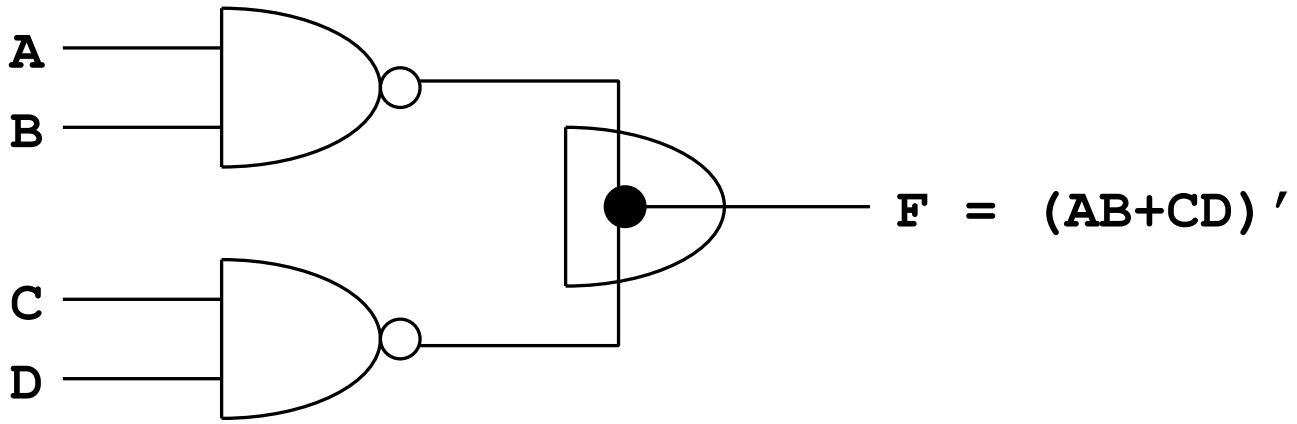
NAND 와 NOR 로 설계구현 (Cont'd)

– NOR 회로로 바꾸는 예



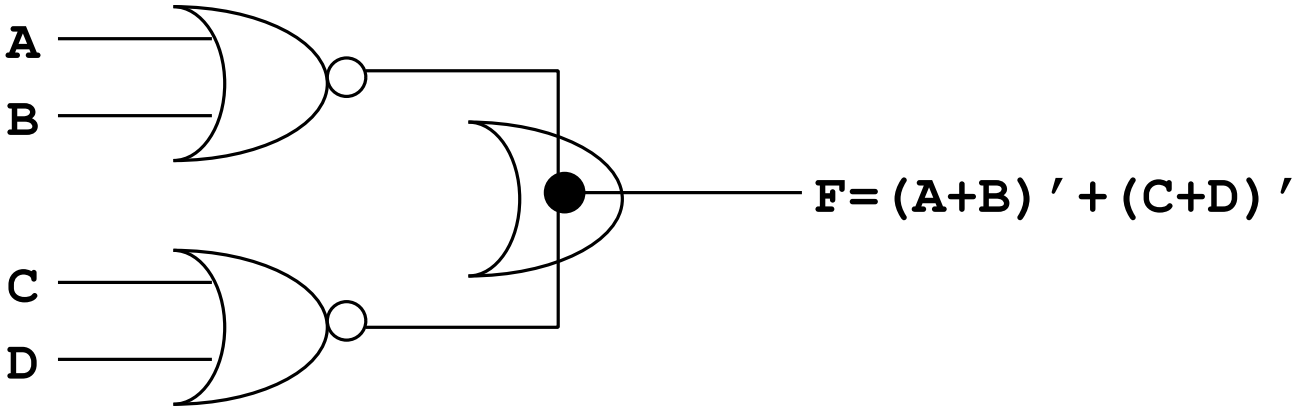
Wired Logic

- Wired AND
  - open-collector TTL NAND 를 선으로 연결 →Wired AND
  - 기호



Wired Logic (Cont'd)

- Wired OR
  - ECL NOR 를 선으로 연결 →Wired OR
  - 기호



배타적 OR(XOR) 게이트

- 교환 결합법칙이 성립
  - A $\oplus$B = A'B + AB' = $\sum$ (1,2) = (A + B)(A' + B') = $\prod$ (0,3)
  - A $\odot$B = A'B' + AB = $\sum$ (0,3) = (A'+ B)(A + B') = $\prod$ (1,2)
  - (A $\oplus$B)' = A $\odot$B
  - A $\oplus$B $\oplus$C = A $\oplus$(B $\oplus$C) = (A $\oplus$B) $\oplus$C
  - $\oplus$: n bit 에서 1의 갯수가 홀수이면 1, 짝수이면 0

    (모든 bit 사이에는 연산자가 있다고 가정)
    - 예) 01101001 $\rightarrow$00001111 $\rightarrow$(00)(00)(11)(11) $\rightarrow$0 0 0 0 $\rightarrow$(00)(00) $\rightarrow$00 $\rightarrow$0

      01001001 $\rightarrow$00000111 $\rightarrow$(00)(00)(11)(10) $\rightarrow$0 0 0 1 $\rightarrow$(00)(01) $\rightarrow$01 $\rightarrow$1

배타적 OR(XOR) 게이트 (Cont'd)

- $\odot$: n bit 에서 0의 갯수가 홀수이면 0, 짝수이면 1

  (모든 bit 사이에는 연산자가 있다고 가정)

  * 예) 01101001 $\rightarrow$00001111 $\rightarrow$(00)(00)(11)(11) $\rightarrow$1 1 1 1 $\rightarrow$(11)(11) $\rightarrow$11 $\rightarrow$1
    01001001 $\rightarrow$00000111 $\rightarrow$(00)(00)(01)(11) $\rightarrow$1 1 0 1 $\rightarrow$(11)(01) $\rightarrow$10 $\rightarrow$0
- 그러므로 (A $\oplus$B $\oplus$C) = (A $\odot$B $\odot$C) 이고

  (A $\oplus$B $\oplus$C $\oplus$D) = (A $\odot$B $\odot$C $\odot$D)' 이다
- 수식적으로 윗식을 유도
  * (A $\oplus$B)' = A $\odot$B
  * (A $\odot$B)' = A $\oplus$B
  * A $\oplus$B = A'B + AB'
  * A $\oplus$1 = A' 1 + A 0 = A'
  * A $\oplus$0 = A' 0 + A 1 = A
  * (A $\oplus$B $\oplus$C) = (A $\oplus$B $\oplus$C) $\oplus$0 = (A $\oplus$B $\oplus$C) $\oplus$(1 $\oplus$1) = ((A $\oplus$B) $\oplus$1 C $\oplus$1) =
    ((A $\oplus$B)' $\oplus$C $\oplus$1)
    = ((A $\odot$B) $\oplus$C $\oplus$1) = ((A $\odot$B) $\oplus$C)' = (A $\odot$B) $\odot$C = A $\odot$B $\odot$C
  * (A $\oplus$B $\oplus$C)' = (A $\odot$B) $\oplus$C = A $\odot$B $\oplus$C

배타적 OR(XOR) 게이트 (Cont'd)

* (A ⊕B ⊕C ⊕D) = (A ⊕B ⊕C ⊕D) ⊕0 = (A ⊕B ⊕C ⊕D) ⊕(1 ⊕1) = ((A ⊕B) ⊕1 ⊕(C ⊕D) ⊕1)
  = (A ⊙B) ⊕(C ⊙D) = ((A ⊙B) ⊙(C ⊙D))' = (A ⊙B ⊙C ⊙D)'

– Parity generator and Parity checker
  * 3bit 홀수 parity generator
    · P = (x ⊕y ⊕z)' = x ⊙y ⊕z
  * 4bit 홀수 parity checker
    · C = (x ⊙y) ⊙(z ⊙P)
    · C = 1 이면 전송 에러
    · C = 0 이면 전송 에러 없음
  * 4bit 홀수 parity checker 는 4bit 홀수 parity generator 와 같다

디지털 게이트 (Digital gates)

- 실제로 IC 화 할 때 AND나 OR보다 NAND, NOR가 잘 사용됨 : 트랜지스터로 쉽게 제작됨
- 다중 입력으로의 확장
  - AND 와 OR 게이트는 교환법칙과 결합법칙이 성립하므로 여러개의 입력으로 확장 가능
  - NAND 와 NOR는 결합법칙이 성립안함으로 바로 불가능
  - XOR 와 XNOR 는 가능

집적회로 (IC)

- 디지탈 논리군
  - TTL (Transistor-transistor logic) : 가장 널리쓰임
  - ECL (Emitter-coupled logic) : 고속동작 요구시
  - MOS (Metal-oxide semiconductor) : 고 집적도 요구시
  - CMOS (complementary metal-oxide semiconductor) : 저전력 요구시
  - (Integrated-injection logic) : 고 집적도 요구시
- 양논리 와 음논리 (* 그림 2-42)

집적회로 (IC) (Cont'd)

- IC 의 특수한 성질
  - 팬아우트 (fan-out) : 정상동작으로 출력에 달수 있는 표준 부하의 숫자, 게이트 출력에 연결될수 있는 최대의 입력수 (버퍼나 증폭기 를 이용 늘림)
  - 전력 소모 : 게이트를 작동하기 필요한 전력
  - 전파지연 시간: 입력 이진신호의 변화가 출력까지 전달되는데 걸리는 평균시간
  - 잡음 허용치 : 출력의 값을 변화시키지 않는 입력의 최대 잡음허용치

CMOS 회로

- 게이트를 CMOS 회로로 구혐하는 방법
- n channel 트랜지스터 와 p channel 트랜지스터를 이용함
- CMOS 를 이용한 게이트 구현 (*그림 2-46)
- 예) 2-10 (*그림 2-47)
- 전송게이트 (*그림 2-48, 2-49)

Homework

- 1장 연습문제
  - 2, 3, 4, 6, 8, 11, 20, 25
- 2장 연습문제
  - 1(c), 2, 5, 6, 7, 8(d), 11(d), 13(b)(d), 14(c), 18(d), 22(c), 25, 28(a)