디지털 기초 이론

서울시립대학교
컴퓨터과학부

양 정 훈

Part I

산술 연산

2진수 표기

1) 정보의 크기

2진수 1자리 : 둘 중의 하나를 선택할 수있는 크기의 정보비트(bit)

2진수 8자리 : 256 가지 정보 표현가능 - 1바이트(byte)

2) 진법 변환

(1) n 진법의 수를 10 진법 수로 변환

n진법의 수 $a_3a_2a_1a_0.b_1b_2b_3$ (각 a, b는 n보다 작은 수)

=> $a_3*n^3+a_2*n^2+a_1*n^1+a_0*n^0+b_1*n^{-1}+b_2*n^{-2}+b_3*n^{-3}$

10진법의 수 1234.56

=> $1x10^3 + 2x10^2 + 3x10^1 + 4x10^0 + 5x10^{-1} + 6x10^{-2}$

2진수 1011.01

=> $1x2^3 + 0x2^2 + 1x2^1 + 1x2^0 + 0x2^{-1} + 1x2^{-2} = 11.25$

2진수 표기

(2) 10진수를 2진수로 변환

(십진수)13.75 = $a_3a_2a_1a_0.b_1b_2b_3$...(2진수)(각 a, b는 0 또는 1)

$$\mathbf{13.75}=a_3x2^3+a_2x2^2+a_1x2^1+a_0x2^0+b_1x2^{-1}+b_2x2^{-2}+b_3x2^{-3}$$

<정수부>

2	13		
2	6	… 1	(a_0)
2	3	… 0	(a_1)
2	1	… 1	(a_2)

십진수 13은 2진수 1101

< 소수부>

0.75
x 2
1.50 … 1
0.5
x 2
1.0 … 1

십진수 0.75는 2진수 0.11

★ 앞에서 구한 정수부와 소수부를 결합하면, 13.75의 2진 표현은 **1101.11**이 된다

수의 표현

1) 보수

음의 정수를 나타내기 위해 보수의 표현 사용
2진수에는 1의 보수와 2의 보수 표현

(1) 1의 보수

k자리 2진수 N의 1의 보수 : $(2^k - 1) - N$ = 1111....1111(1이 k자리) - N

예) 10101010인 수의 1의 보수 표현은 11111111 - 10101010 = 01010101이 된다.

♣ **N의 1의 보수는 각 자리수의 보수를 취한 결과와 같다.**

(2) 2의 보수

k자리 2진수 N의 2의 보수 : $2^k - N$ = 1000....0000(0이 k자리) - N

예) 10101010인 수의 2의 보수 표현은 100000000 - 10101010 = 01010110이된다.

♣ **N의 2의 보수는 1의 보수에 1을 더한 값과 같다.**

수의 표현

2) 정수의 표현

♣ 음의 정수를 표현하는 방법

(1) 부호-크기 표현 방식 (signed magnitude)

1비트로 부호를, 나머지 비트들로 수의 절대값

(2) 부호와 1의 보수 표현 방식 (signed 1's complement)

1비트로 부호를, 나머지 비트들로 수의 절대값에 대한 1의 보수값

(3) 부호와 2의 보수 표현 방식 (signed 2's complement)

1비트로 부호를, 나머지 비트들로 그 수의 절대값에 대한 2의 보수값

수의 표현

-7을 부호를 포함하여 **5**비트로 표현하라.

부호크기 : 1 0111

1의 보수 : 1 1000

2의 보수 : 1 1001

위와 같이 **8**비트로 정수를 표시할 때 세 방식에서 각각 표현 가능한 수의 범위?

부호 - 크기 : $+(2^7 - 1)$; 01111111 ~ $-(2^7-1)$; 11111111 즉, +127 ~ -127

1의 보수 : $+(2^7 - 1)$; 01111111 ~ $-(2^7-1)$; 1000000 즉, +127 ~ -127

2의 보수 : $+(2^7 - 1)$; 01111111 ~ $-(2^7)$; 1000000 즉, +127 ~ -128

위와 같이 **8**비트로 정수를 표시할 때 세 방식에서 **0**의 표현을 적어라.

부호 - 크기 | : 1000000(음의 0), 0000000(양의 0)

1의 보수 : 11111111(음의 0), 0000000(양의 0)

2의 보수 : 0000000(유일한 0)

수의 표현

정수 **x**의 **2**의 보수는 **–x,** 이것의 **2**의 보수는 원래 값이다**.**

음수의 **2**의 보수를 취하면 절대값이다**.**

2의 보수회로로 감산기를 대치한다**.**

비부호 정수의 부호확장

캐리 **(c** 비트**) vs** 오버플로우 **(v** 비트**)**

		비부호정수 (2)	정수 (2)
	0000 0010		
+	1000 0001	(129)	(-127)

	1000 0011	(131)	(-125)

수의 표현

3) 부동 소숫점에 의한 실수의 표현

(1)구성

가수부 : 수의 유효 자릿수들을 부호화된 수로 나타낸다.

: 가수는 부호 비트와 크기부분으로 구성

지수부 : 실제 소숫점의 위치를 나타낸다.

예) 1234.56 = 0.123456×10^4 가수부 123456 과 지수부 4로 표시가능

♣ **정규화 표현 : 크기부분의 MSB가 1이 되도록 표현**

(예제) 2진수 1010.101을 8비트의 지수와 8비트의 가수(부호 비트 포함) 로 표현하라.

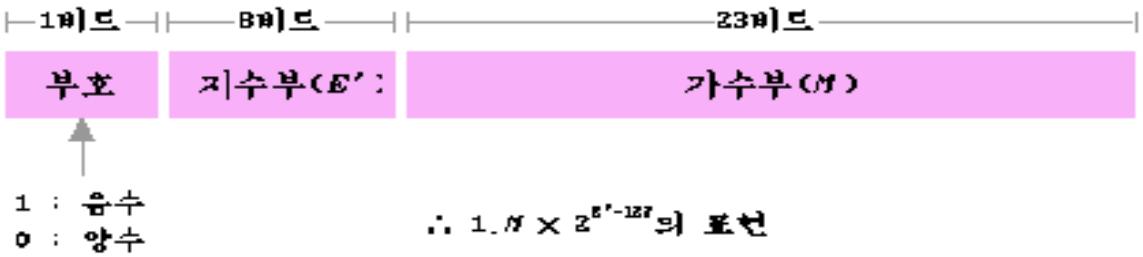
$1010.101 = 0.1010101 \times 2^4$이므로 가수는 부호비트를 포함하여 **01010101**으로, 지수는 **00000100**으로 표현된다.

수의 표현

(2) IEEE의 Computer Society 규정

◑ **지수 표현 :** 지수는 실제 지수에 127을 더한 값으로 양수로만 표현

◑ **가수부 표현 :** 정규화로 표현된 가수의 첫째 비트를 제외한 나머지 비트들로 표현



[그림 3-3] IEEE의 부동 수점 수의 표준안 형식

(예제) 1.001010000 x 2^{-87} 을 IEEE 표준 형식으로 표현하라.

(풀이) 지수부 : -87 + 127 = 40 = 00101000

가수부 부호 : 0 가수: 001010000.....0 (23비트)

00010100000010100000.....0 (32비트)

(예제) -118.625를 IEEE 표준 형식으로 표현하라.

정수의 연산

1) 정수의 곱셈

(1) shift 기능을 이용하여 곱셈처리

A x 01100(=A x 12) 는 A x 10000 (=A x 16) - A x 00100 (=A x 4) 와 동일하다는 원리를 이용 A의 왼쪽 쉬프트 기능과(실제 의미는 2의 배수)와 덧셈(뺄셈 포함)만으로 가능

(2) 정수를 한자리만이 1인 수의 합으로 표현 방법

- 1이 1개이상 연속된 부분에서의 처음 1이 있는 비트가 i번째 비트라면 -2^i를 추가
- 0이 1개이상 연속된 부분에서 처음 0이 있는 비트가 j번째 비트라면 $+2^j$를 추가
- 그외의 비트는 0으로 한다.

정수의 연산

(예제) 10011(-13) ?

(1) 가장 오른쪽 0번째 위치의 비트가 1이므로 -2^0 항

(2) 오른쪽에서 2번째 위치의 비트에서 0이 1개이상 연속되므로 $+2^2$ 항

(3) 가장 왼쪽의 4번째 비트에서 1이 시작하므로 -2^4항을 갖는다.

따라서, 10011은 $-2^0 + (+2^2) + (-2^4)$과 같게 된다.

(예제) 11001(-7)을 한 자리만이 1인 수들의 연산으로 표현하라.

★ 11001에서 오른쪽에서 0번째 비트부터 1이 연속 : $2^0=1$

오른쪽에서 1번째 비트부터 0이 연속 : $+2^1=10$

오른쪽에서 3번째 비트부터 1이 연속 : $2^3=1000$

따라서, **11001=-1+10+(-1000)**이 된다.

정수의 연산

(예제) 두 수 (6, −7)의 곱을 구하라. **2**의 보수를 사용하고 각 정수는 부호비트를 포함하여 **5**비트로 표현된다.

풀이 ▪ 앞의 예제에서, $-7 = 11001 = -1 + 10 + (-1000)$이므로,

$$\begin{aligned} 6 \times (-7) &= 00110 \times 11001 \\ &= 00110 \times (-1 + 10 + (-1000)) \\ &= 00110 \times (-1) + 00110 \times (10) + 00110 \times (-1000) \\ &= 11010 + 001100 + 11010000 \end{aligned}$$

이때, 이 덧셈은 부호를 고려하여 10비트로 확장하여 수행한다.

$$\begin{array}{r} 1111111010 \\ 0000001100 \\ +\ 1111010000 \\ \hline 1111010110 \end{array}$$

♣ 곱셈에서는 오버플로우가 없으므로 캐리를 무시하면, 결과는 **1111010110 (-42)**이된다.

실수의 연산

1) 부동 소수점 수의 덧셈과 뺄셈

(1) 지수 조정 (큰 지수를 기준으로)

(2) 가수의 덧셈과 뺄셈 연산 실행

(3) 결과의 정규화

(예제) 0.1011×2^4 과 0.1110×2^6의 덧셈을 수행하라.

1. 지수 **6**을 기준으로 하여, 0.1011×2^4 을 0.001011×2^6으로 지수 조정한다.

2. 각 가수를 더한다.

0.001011

+ 0.1110

1.000011 이므로, 결과는 1.000011×2^6이다.

3. 이를 정규화 하면 0.1000011×2^7이된다.

실수의 연산

2) 부동 소수점 수의 곱셈

◑ 승수 또는 피승수가 **0**이면 곱의 결과는 **0**이다.

◑ 그 이외에는 다음 순서로 계산한다.

(1) 승수와 피승수의 가수를 곱셈

(2) 승수와 피승수의 지수는 서로 덧셈

(3) 결과를 정규화

(예제) $(0.101 \times 2^4) \times (0.11 \times 2^6)$ 을 계산하라.

1. 가수의 곱 : **$0.101 \times 0.11 = 0.01111$**

2. 지수의 합 : **$4+6=10 \Rightarrow 2^{10}$**

=> 결과 : 0.01111×2^{10}

3. 정규화 **=> 0.1111×2^9**

Part II

논리 연산

CPU 구조

- CPU : 연산을 수행하는 디지털 회로, 수백만개 이상의 트랜지스터로 이루어져 있음.
- 트랜지스터 : 켜짐 상태와 꺼짐 상태를 나타낼 수 있는 소형 전자 스위치
- CPU 트랜지스터 회로의 기본 기능
  - 덧셈 (adding) : 수학 함수
  - 디코딩 (decoding) : 메모리 위치 선택 등
  - 시프팅 (shifting) : 덧셈과 함께 수학 함수
  - 저장 (storing) : latch, flip-flop
- 실제적인 계산 작업을 위해서는 트랜지스터를 특수한 회로로 묶어야 함 : 논리 회로 (logic circuit) → 부울 대수 수행
  - 부울 연산자의 기능을 수행하는 회로 : 게이트 (gate)
  - 기본 게이트를 조합하여 가산기, 디코더, 시프터, 플립플롭 회로를 구성할 수 있음.
  - IC, LSI, VLSI

부울 연산과 부울 대수

1) 부울 연산

(1) 부울 연산식

이름이 부여된 2진 변수(x, y, z 등)와 논리 연산자 +, *, '로 구성되는 연산식 예) E = xy + y'z등은 부울 연산식이다.

(2) 진리표

모든 입력들의 조합에 대해 연산 결과의 출력 값을 표로 표시

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

(a) x+y

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

(b) xy

x	x′
0	1
1	0

(c) x′

[그림 2-2] 논리 연산의 진리표

부울 연산과 부울 대수

(3) 리터럴(literal)

각 변수 또는 보수화된 변수 (예를 들어 **x, y', z**등)

(4) 기본곱 (fundamental product 또는 minterm)

리터럴 또는 동일한 리터럴이 반복되지 않는 리터럴들의 곱

예) x, xy, xy', xyz 등은 기본곱

★ 기본곱의 포함관계

기본곱 (P1)이 기본곱(P2)의 한 부분인 경우, P1 + P2 = P1 이된다.

예) xy + xyz = xy

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

[그림 2-31] xy+xyz의 진리표

(5) 분리 정규형(dnf, disjunctive normal form)

하나 이상의 기본곱들(이들 사이에는 서로 포함되는 경우가 없는경우)의 논리합

부울 연산과 부울 대수

2) 부울 대수

논리 연산자 +, *, ' 및 두 개의 성분 **0**과 **1**에 대해, 집합 **B**가 다음 **4**개의 법칙들을 만족할 때, 집합 **B**를 부울대수라 한다.

[1] 교환법칙(commutative law)	$a+b=b+a$	
	$a \times b=b \times a$	
[2] 분배법칙(distributive law)	$a \times (b+c)=(a \times b)+(a \times c)$	
	$a+(b \times c)=(a+b) \times (a+c)$	
[3] 항등원법칙(identity law)	$a+0=a$	$a \times 1=a$
[4] 보수법칙(complement law)	$a+a'=1$	$a \times a'=0$

부울 대수는 다음여러 법칙(공리)들을 만족한다.

[5] 등멱 법칙(idempotent law)	$a+a=a$
[6] 경계 법칙(boundness law)	$a+1=1$
[7] 흡수 법칙(absorption law)	$a+(a \times b)=a$

부울 연산과 부울 대수

[8] 연관 법칙(associative law)	$(a + b) + c = a + (b + c)$
[9] 대합 법칙(involution law)	$(a')' = a$
[10] 드모르강 법칙(Demorgan's law)	$(a + b)' = a' \times b'$

◈ 쌍대성 특성을 적용한 항등식 : + ➔ × , × ➔ + , 0 ➔ 1 , 1 ➔ 0 으로 변환

3) 부울식의 분리 정규형화 또는 식의 간단화 과정

(1) 드모르강 법칙과 대합 법칙이 적용되면 적용, 보수 연산들을 처리한다. 예) (a'b')'c = (a + b)c

(2) 분배법칙의 적용으로 리터럴의 곱의 합으로 전환하고, 교환법칙, 등역법칙, 흡수 법칙중 가능한 법칙을 적용 기본곱들의 합 형식인 분리 정규식을 얻는다. 예) (a + b)c = ac + bc

부울 연산과 부울 대수

(예제) E=(a(bc)')'(a+b)을 기본곱들의 합으로 표현하라.

E =(a'+((bc)')')(a+b) (드모르강의 법칙)

=(a'+bc)(a+b) (대합 법칙)

위의 식에 단계 2를 적용해보면,

E=(a'+bc)(a+b)

=aa'+abc+a'b+bbc (분배 법칙)

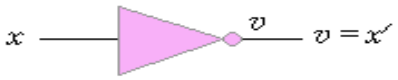
=abc+a'b+bc (보수 법칙, 등역 법칙)

=a'b+bc (흡수 법칙)

논리 게이트

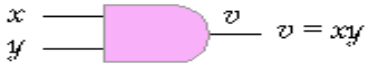
이진 정보에 대한 **AND, OR** 및 **NOT**등의 논리 연산을 실행하는 논리 회로

① Inverter(NOT 연산)



x	v
0	1
1	0

② AND 게이트

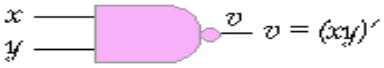


xy	v
00	0
01	0
10	0
11	1

③ OR 게이트

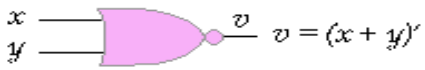
xy	v
00	0
01	1
10	1
11	1

④ NAND 게이트



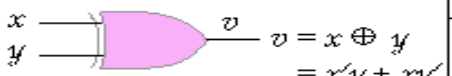
xy	v
00	1
01	1
10	1
11	0

⑤ NOR 게이트



xy	v
00	1
01	0
10	0
11	0

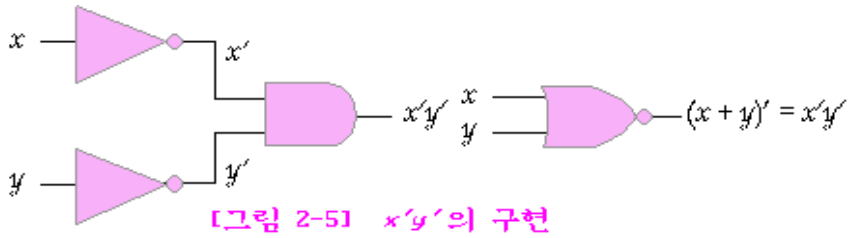
⑥ XOR 게이트



xy	v
00	0
01	1
10	1
11	0

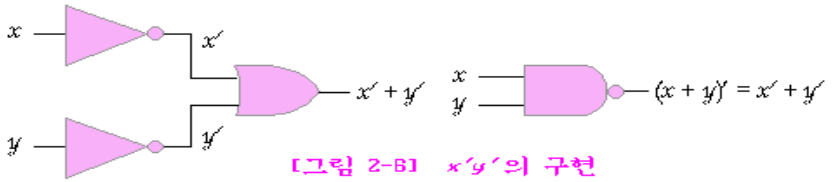
[그림 2-4] 논리 게이트(이름, 그림부호, 논리 연산 및 진리표)

논리 게이트



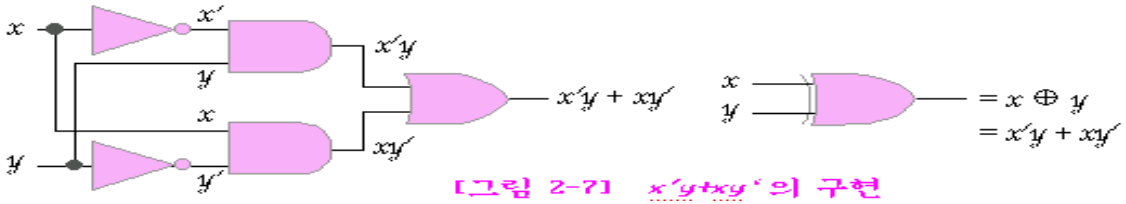
[그림 2-5] x′y′의 구현

1) NOR 게이트의 활용



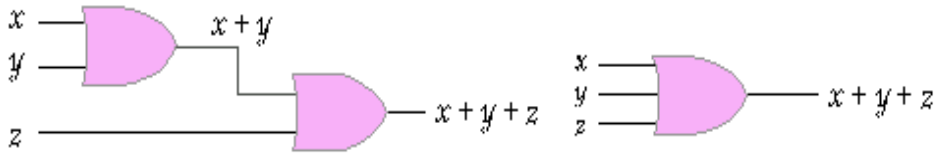
[그림 2-6] x′y′의 구현

2) NAND 게이트의 활용



[그림 2-7] x′y+xy′의 구현

3) XOR 게이트의 활용



[그림 2-8] 3입력 OR 게이트

4) 3입력 게이트

논리 게이트

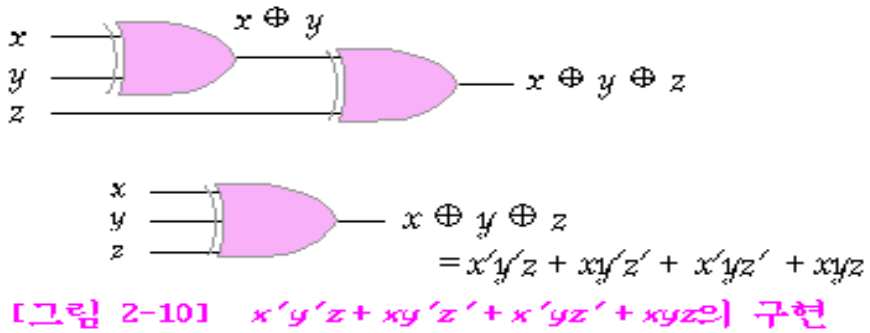
(예제) 다음과 같은 **3**변수 연산을 가장 간단하게 구현하라.

v=x'y'z+xy'z'+x'yz'+xyz

x	y	z	$x'y'z$	$xy'z'$	$x'yz'$	xyz	v
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1
0	1	0	0	0	1	0	1
0	1	1	0	0	0	0	0
1	0	0	0	1	0	0	1
1	0	1	0	0	0	0	0
1	1	0	0	0	0	0	0
1	1	1	0	0	0	1	1

$v = x \oplus y \oplus z = x'y'z + xy'z' + x'yz' + xyz$

[그림 2-9] v의 진리표



[그림 2-10] $x'y'z + xy'z' + x'yz' + xyz$의 구현

논리게이트의 구현

1) 논리게이트의 실제 회로

5V를 입력으로 사용하는 TTL 회로

입력 단자: 0 ~ 0.8V을 0, 2 ~ 5V를 1

출력 단자: 0 ~ 0.4V을 0, 2.4 ~ 5V을 1

2) 논리 회로 구현에 사용되는 여러 소자

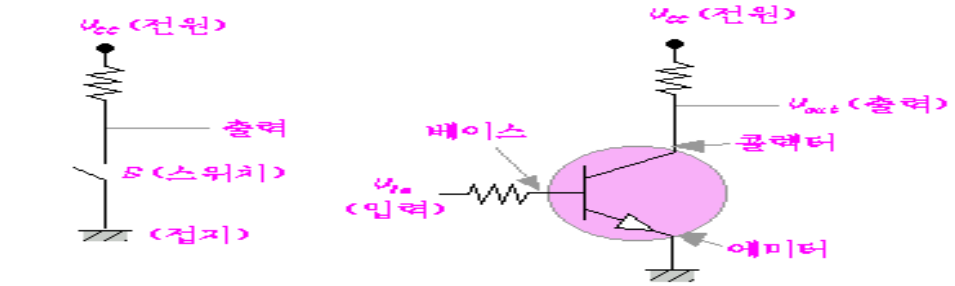
(1) TTL : 양극성 트랜지스터로 구현되며 논리 게이트 구현에 가장 많이 사용된다.

(2) ECL(emitter-coupled logic) : 게이트 지연 시간(1-2nsec)이 매우 적어 특성상 고속회로에 사용

(3) MOS(metal oxide semiconductor) TR(transistor) : 낮은 속도에서 동작하는 VLSI에 적합

(4) CMOS(complementary metal oxide semiconductor) : 잡음에 대한 영향이 적고, 전력 소비가 적고 회로의 밀도가 높다.

논리게이트의 구현



(a) 스위치 이용 (b) 트랜지스터 이용

[그림 2-11] 인버터의 구현

(a) 스위치 회로

S가 연결(1상태) : 출력 전압은 GND와 같은 0V(GND)가 되어 0

S가 비연결(0상태) : 출력 전압은 Vcc와 같게 되어 1

(b) 트랜지스터 회로

입력= 0(0V): 트랜지스터는 OFF상태, 출력은 Vcc(1 상태)

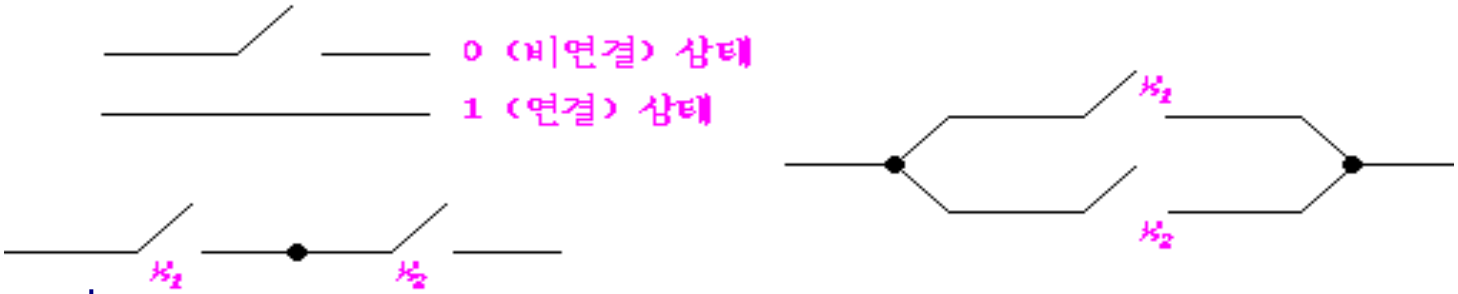
입력= 1(5V): 트랜지스터는 ON상태, 출력은 GND(0 상태)

논리게이트의 구현

◑ 스위치로 **2**진 정보를 표현하거나 논리 연산을 실행

스위치의 연결된 상태 : 1 (x)

스위치의 연결되지 않은 상태 : 0 (x')



(a) 직렬 연결 (b) 병렬 연결

[그림 2-1] 스위치의 직렬 및 병렬 연결

◑ 직렬 연결된 두 스위치

AND 논리 연산

◑ 병렬 연결된 두 스위치

OR 논리 연산

논리 연산식의 간단화

1) 최소 연산식
- ◈ 부울 대수의 여러가지 법칙을 이용하는 방법
- ◈ 카노맵을 이용하는 방법

2) 카노맵(Karnaugh map) 이용 방법

변수들의 모든 가능한 기본곱을 사각형으로 표현하는그림적 표현 방법

(1) 카노맵의 표현

만약 변수가 n개라면 카노맵은 2^n 개의 사각형으로 구성

각 인접 사각형은 하나의 변수만이 서로 달라야한다

출력이 1인 기본곱에 해당하는 사각형은 1로, 나머지는 0으로 표시

	y	y'
x	xy	xy'
x'	$x'y$	$x'y'$

[그림 2-12] 두 변수의 카노맵

	xy	$x'y$	$x'y'$	xy'
z	xyz	$x'yz$	$x'y'z$	$xy'z$
z'	xyz'	$x'yz'$	$x'y'z'$	$xy'z'$

[그림 2-16] 세 변수의 카노맵

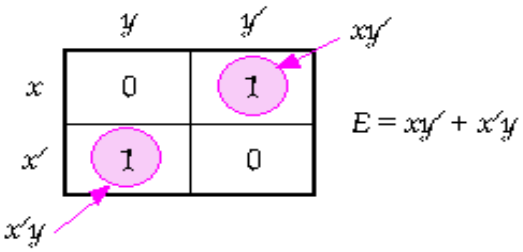
	xy	$x'y$	$x'y'$	xy'
zw	$xyzw$	$x'yzw$	$x'y'zw$	$xy'zw$
$z'w$	$xyz'w$	$x'yz'w$	$x'y'z'w$	$xy'z'w$
$z'w'$	$xyz'w'$	$x'yz'w'$	$x'y'z'w'$	$xy'z'w'$
zw'	$xyzw'$	$x'yzw'$	$x'y'zw'$	$xy'zw'$

[그림 2-19] 4변수의 카노맵

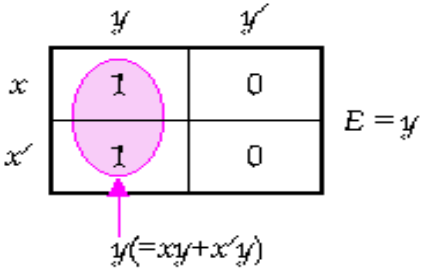
논리 연산식의 간단화

(2) 카노맵을 통한 간단화

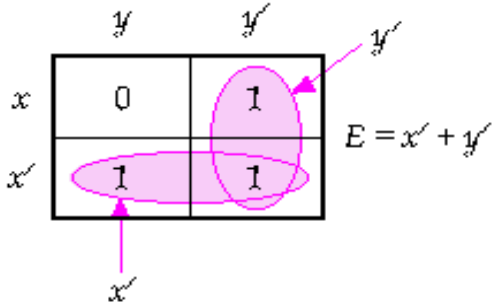
인접한 두개의 사각형을 묶어서 새로운 더 큰 사각형을 만들면 변수의 수가 하나 줄어든 새로운 기본곱



[그림 2-13] xy′ + x′y의 카노맵



[그림 2-14] xy + x′y의 카노맵

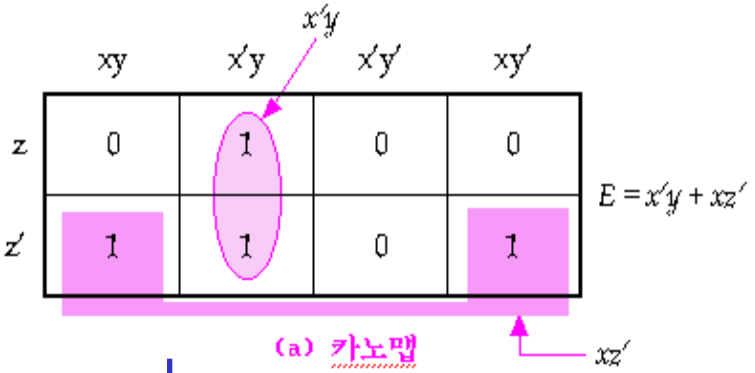


[그림 2-15] xy′ + x′y + x′y′의 카노맵

논리 연산식의 간단화

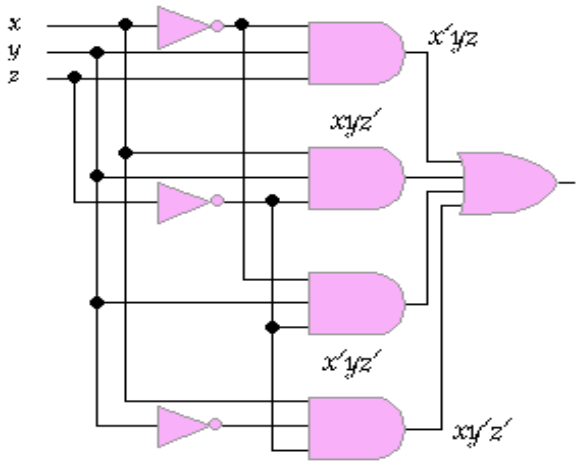
(3) 논리회로구현

$E = x'yz + xyz' + x'yz' + xy'z'$



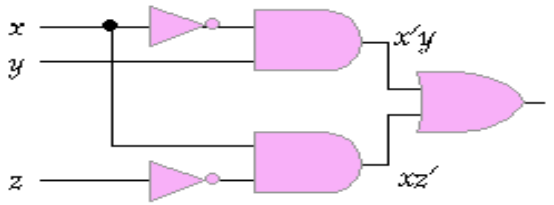
(a) 카노맵

논리회로



(b) $x'yz + xyz' + x'yz' + xy'z'$의 논리 회로

논리회로

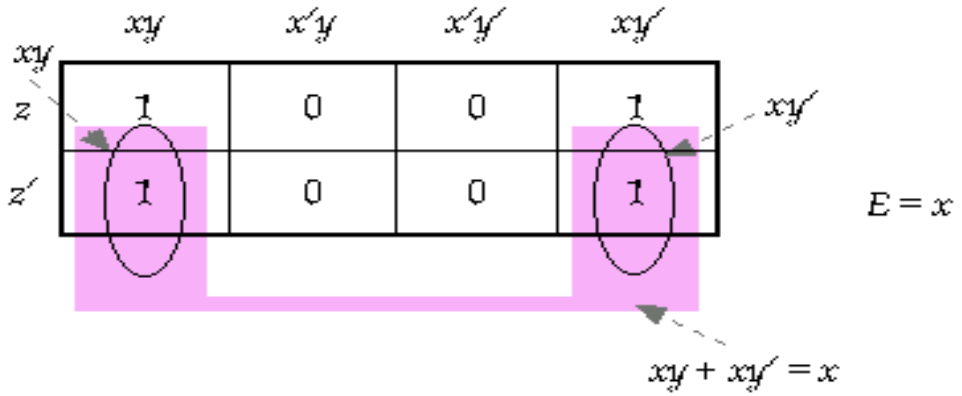


(c) $x'y + xz'$의 논리 회로

[그림 2-17] 3변수 카노맵과 논리 회로 구현

논리 연산식의 간단화

(예제) [그림 2-18]과 같은 카노맵으로 표현되는 연산식 E=xyz+xy'z+xyz'+xy'z'을 최소화하라.

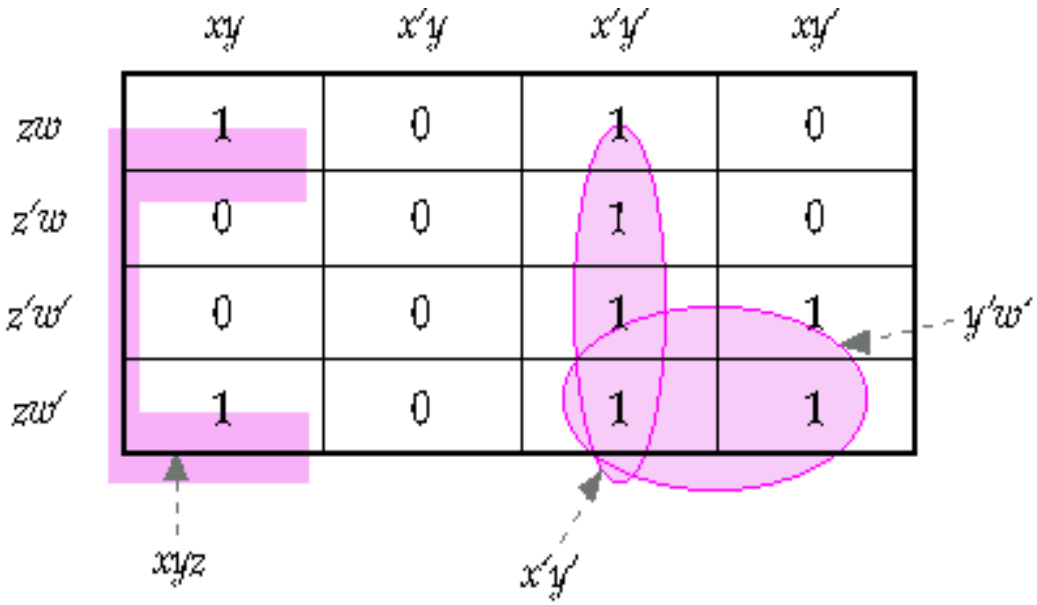


[그림 2-18] 세 변수 카노맵의 예제

논리 연산식의 간단화

◑ 4변수 카노맵

$E = xyzw + xyzw' + xy'zw + xy'z'w + xy'z'w' + xy'zw' + xyz'w' + xyzw'$



$E = xyz + x'y' + y'w'$

[그림 2-20] 4변수 카노맵의 예

논리 연산식의 간단화

◑ **Implicant :** 곱의 합 식 형태의 부울식에서 하나의 곱 항을 **Implicant**라 한다

◑ **PI(Prime Implicant) :** 한 함수에 속한 **Implicant** 중에 다른 **Implicant**에 의해 커버되지 않는 **Implicant**를 뜻한다.

◑ **EPI(Essential Prime Implicant) :** 다른 **PI**에 의해 커버되지 않는 **1**을 커버하는 유일한 **PI**를 뜻한다. 즉, 하나의 **PI**에 의해서만 커버하는 **1**이 있는 경우에 해당 **PI**를 **EPI**라 한다.

◑ 카노맵에서 최소곱의 합 식은 다음의 절차로 찾는다.

1. **EPI**를 모두 찾는다
2. **EPI**를 찾은 후에 남은 **1**들을 커버하기 위한 **"충분한"** 다른 **PI**들을 찾는다. **PI**를 선택할 때에 될 수 있으면 커버되지 않고 남은 **1**들 중에서 가장 많은 **1**을 커버하는 **PI**를 선택한다.

조합 회로

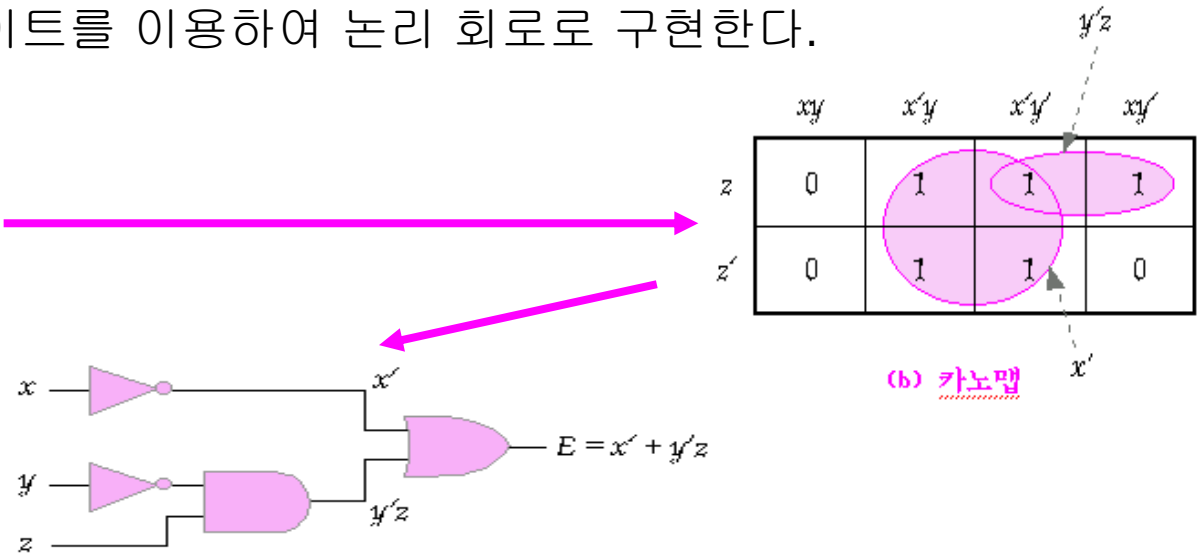
입력 값에 의해서만 출력이 결정되어지는 논리회로

◈ 진리표로부터 논리 회로를 구현하는 방법

① 진리표에서 출력 값이 1인 모든 경우에 대해 부울 연산식을 표현한다.
② ① 에서 구한 식에 대한 카노맵 구성한다.
③ 최소의 연산식을 얻는다.
④ 이 연산식을 게이트를 이용하여 논리 회로로 구현한다.

x	y	z	$E(x,y,z)$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

(a) 진리표



[그림 2-22] 논리 회로 구현의 예

조합 회로

1) 가산기

2 비트 반가산기

입력 : 2개의 입력(x, y)

출력 : 두 개의 출력(합 출력인 S와 캐리 출력인 C)

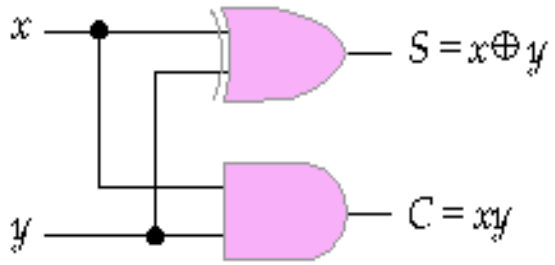
```
  0      0      1      1
+ 0    + 1    + 0    + 1
----   ----   ----   ----
  0      1      1     (1)0
                       캐리(C)
```

x	y	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

(a) 진리표

S = x'y + xy'

C = xy



(b) 회로도

[그림 2-23] 반가산기의 진리표와 회로도

조합 회로

2) 디코더

n 비트의 입력으로 2^n개의 출력중 하나를 **1**로 설정하도록 하는 회로

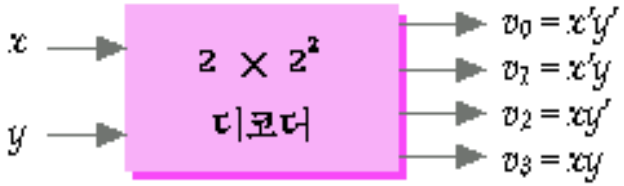
♣ **2입력 4출력 디코더 (2x4 디코더)**

입력: x, y

출력: vo, v1, v2, v3

x	y	v_0	v_1	v_2	v_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

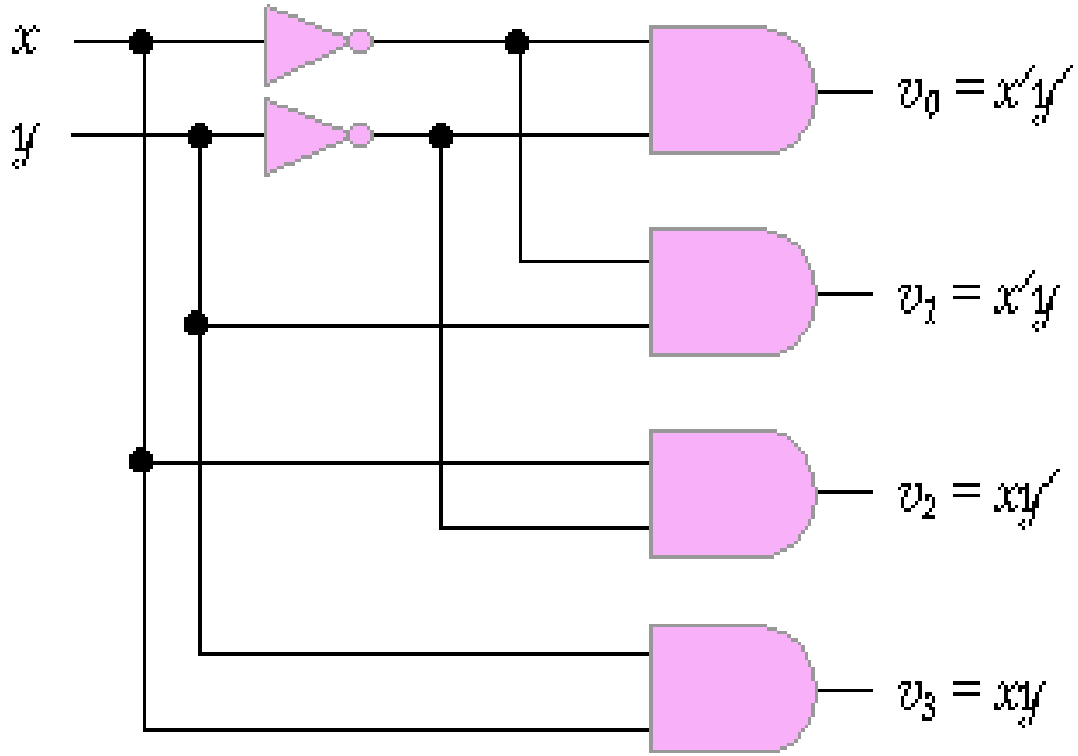
(a)



(b)

[그림 2-26] 디코더의 진리표와 입출력선

조합 회로



[그림 2-27] 2×4 디코더 회로도

조합 회로

3) 멀티플렉서 (먹스, MUX)

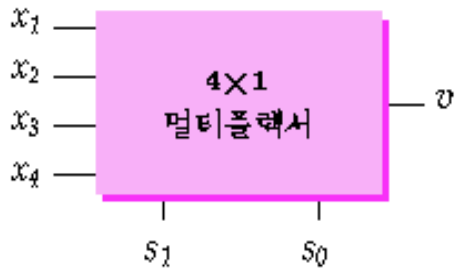
2^n 개의 입력값 중의 하나를 출력으로 연결하는 논리회로

2^n 개의 입력 중 하나를 선택하기 위한 별도의 **n**개의 선택 입력선

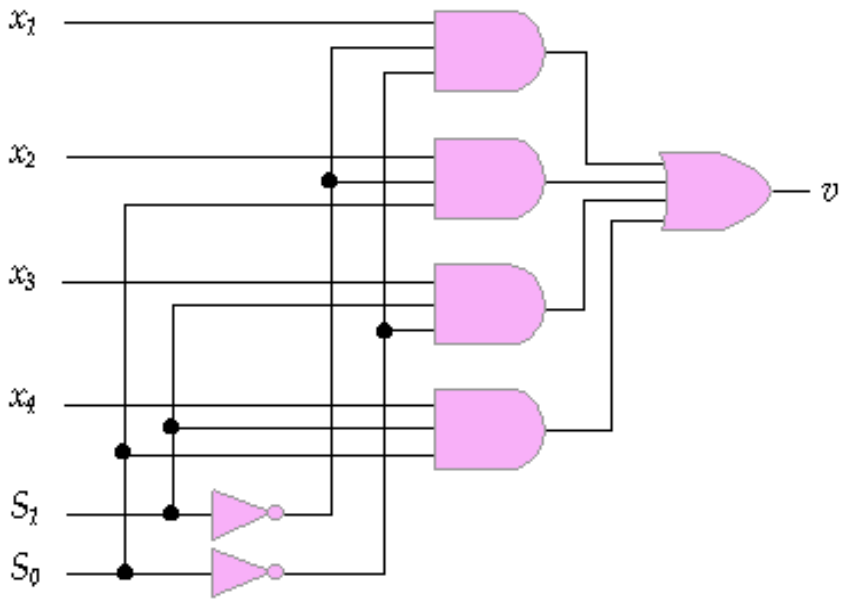
♣ 4x1 멀티플렉서

입력선 x1, x2, x3, x4, 선택선은 s1, s2, 출력선 v

S_1	S_0	v
0	0	x_1
0	1	x_2
1	0	x_3
1	1	x_4



[그림 2-28] 멀티플렉서의 변형된 진리표와 입출력선



[그림 2-29] 4×1 멀티플렉서의 회로도

조합 회로

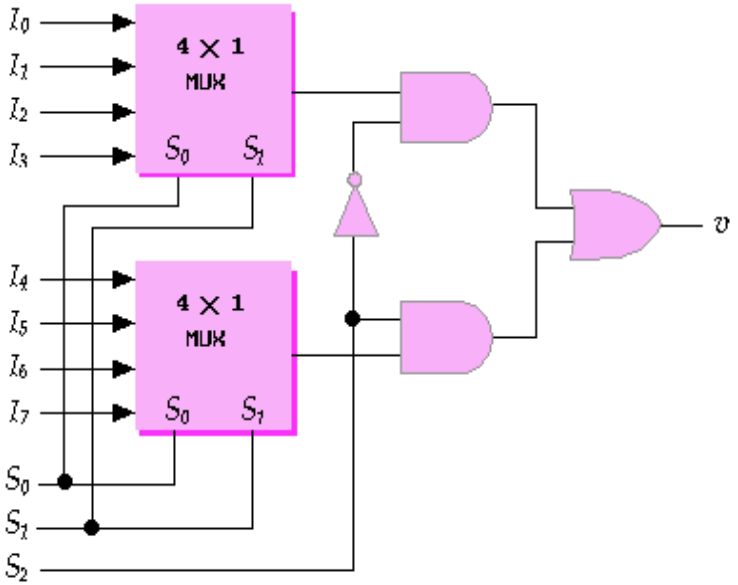
(예제) 4×1 멀티플렉서를 2개 사용하여 8×1 멀티플렉서를 구현하라.

입력선 8개($I_0..I_7$)　　출력선 v

각 멀티플렉서 공통 선택선 S_0,S_1　　멀티플렉서 선택선S_2

S_2	S_1	S_0	v
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

(b) 8 × 1 MUX의 진리표



(a) 8 × 1 MUX의 회로도

[그림 2-30] 8 × 1 멀티플렉서의 구현

조합 회로

4) 패리티 검사 회로

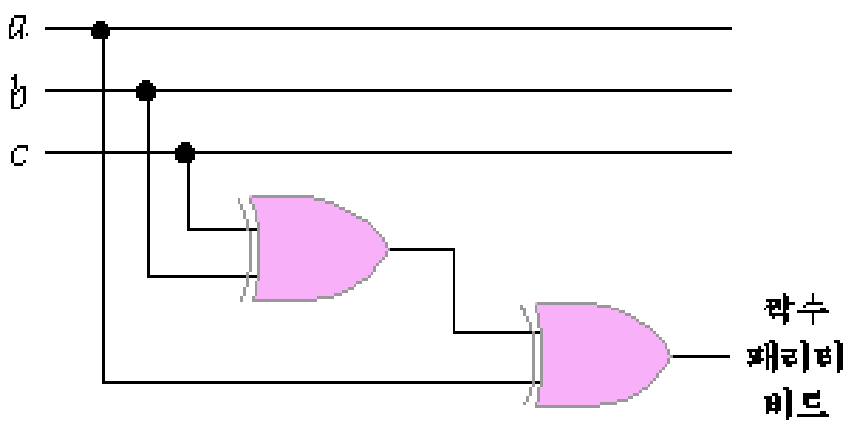
전송되는 정보의 오류를 검출하는 방법

♣ 짝수 패리티 검사

정보의 비트들 중에서 '1'인 비트의 수가 짝수되도록 패리티 비트를 설정

a	b	c	짝수 패리티 비트
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

[그림 2-31] 패리티 비트의 진리표

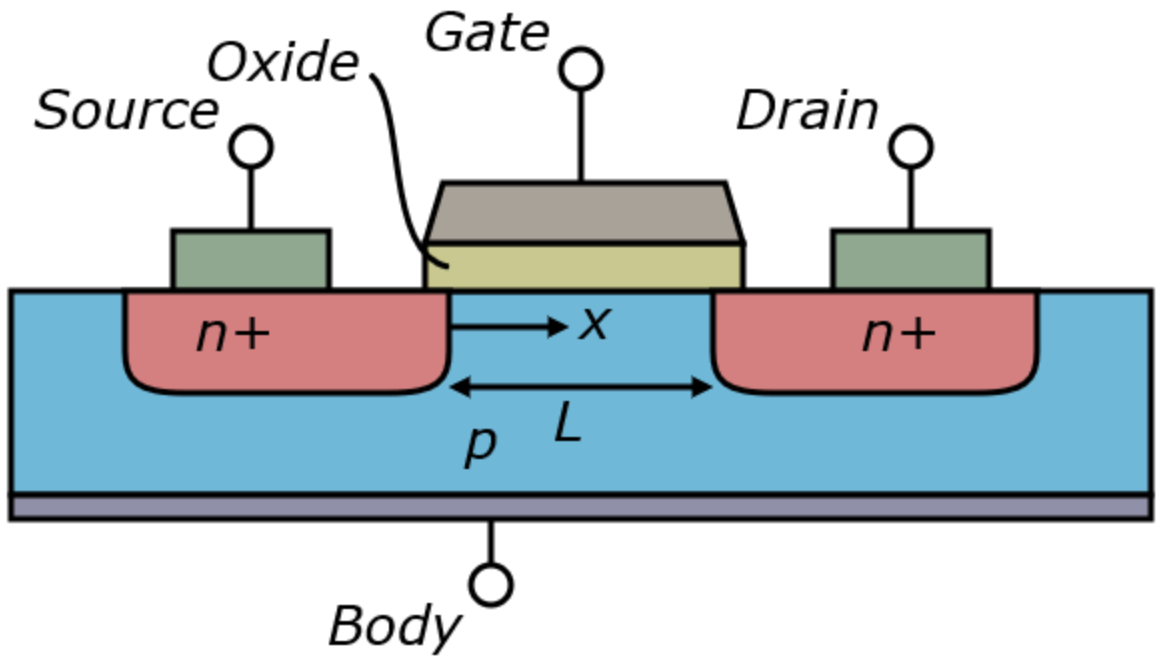


[그림 2-32] 짝수 패리티 비트 생성

ANNEX

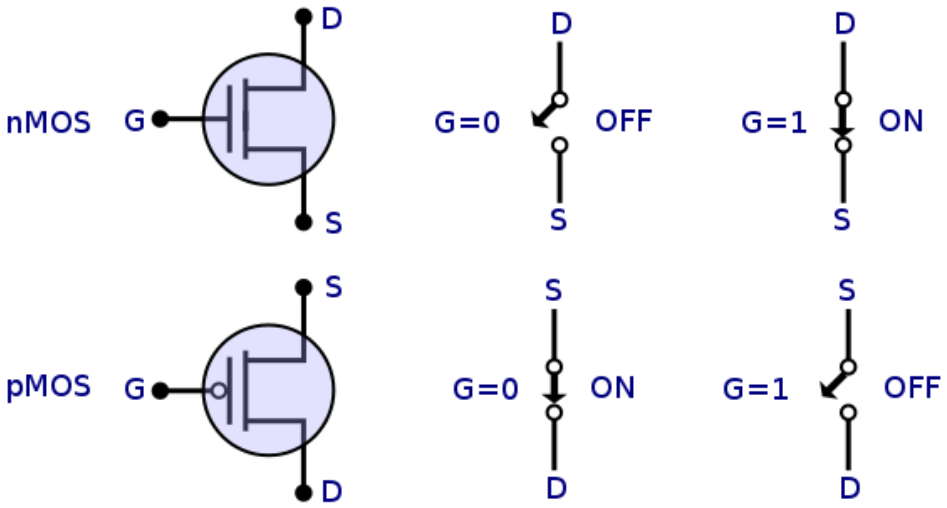
CMOS, 순차논리회로

MOSFET



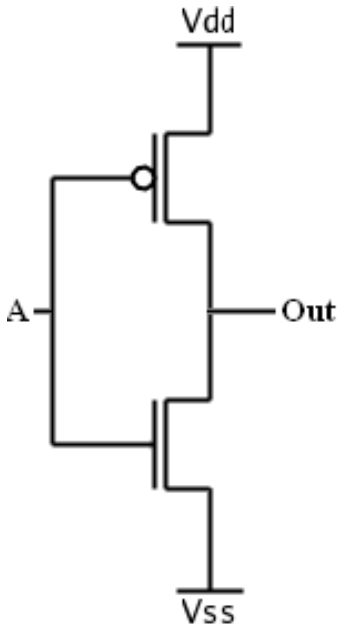
[그림 2-33] n-channel MOSFET

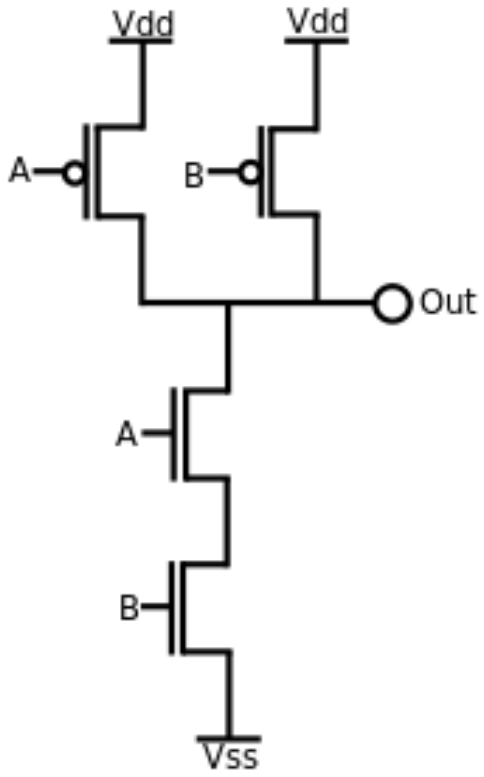
MOSFET

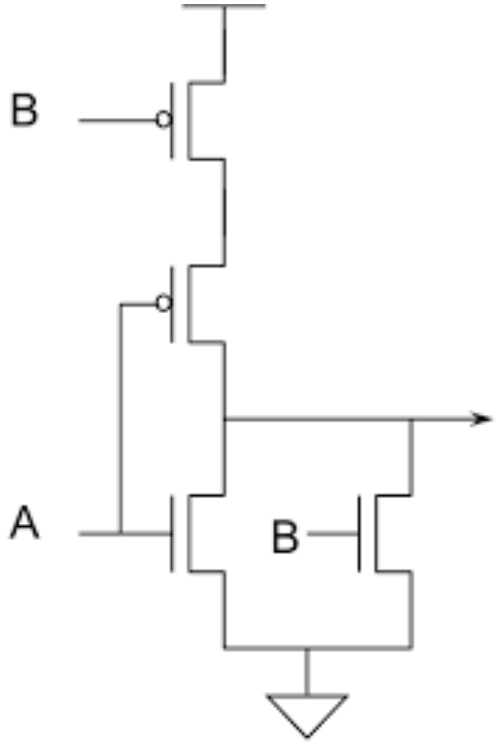


[그림 2-34] MOSFET 스위치 동작

CMOS gate



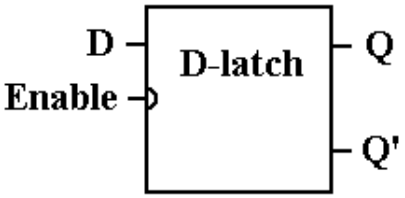




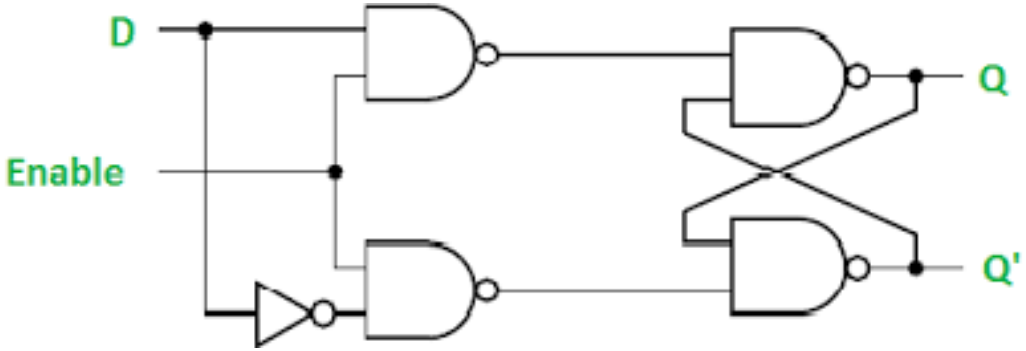
D 래치(Latch)

순차 논리회로 : 입력과 현재의 내부상태값도
출력값을 결정하는 데 사용하는 회로

D Latch : Enable이 1일 때만 D 값을
Q에 저장할 수 있음



D latch 기호



D latch 회로

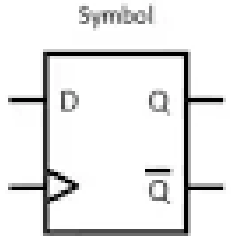
Enable	D	Q+	Q'+
1	1	1	0
1	0	0	1
0	1	Q	Q'
0	0	Q	Q'

D latch 동작

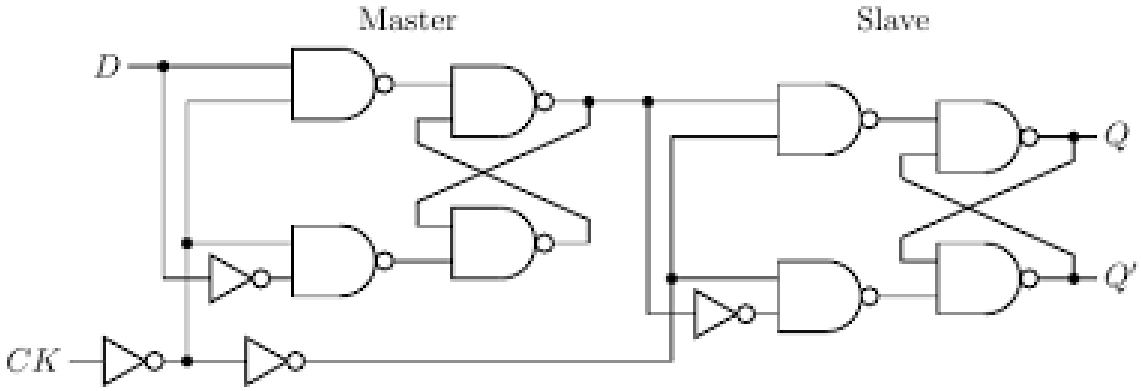
D 플립플롭(Flip Flop)

D flip flop ;

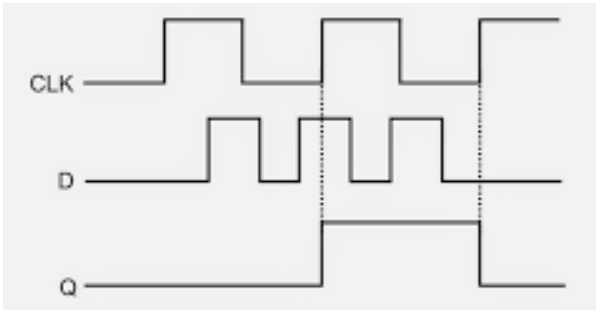
- 입력과 출력 분리
- 클록의 edge 에서만 출력이 변하여 저장됨
- 프로세서 내부의 레지스터 구성에 사용



D flip flop 기호



D flip flop 회로

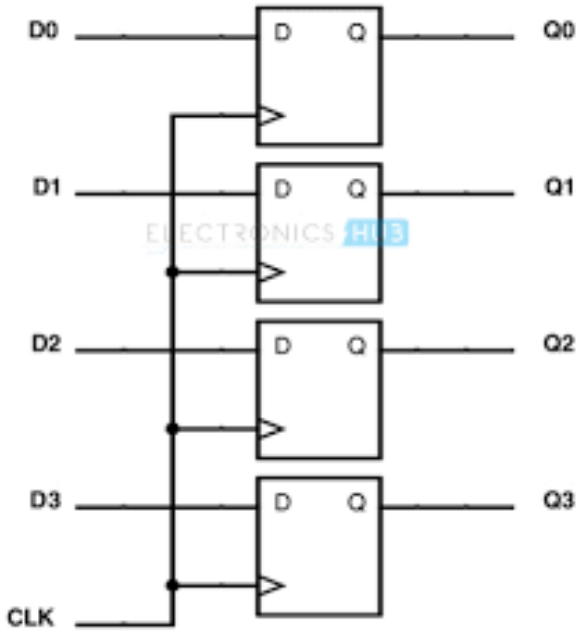


D flip flop 동작
(Rising edge triggered)

레지스터(Register)

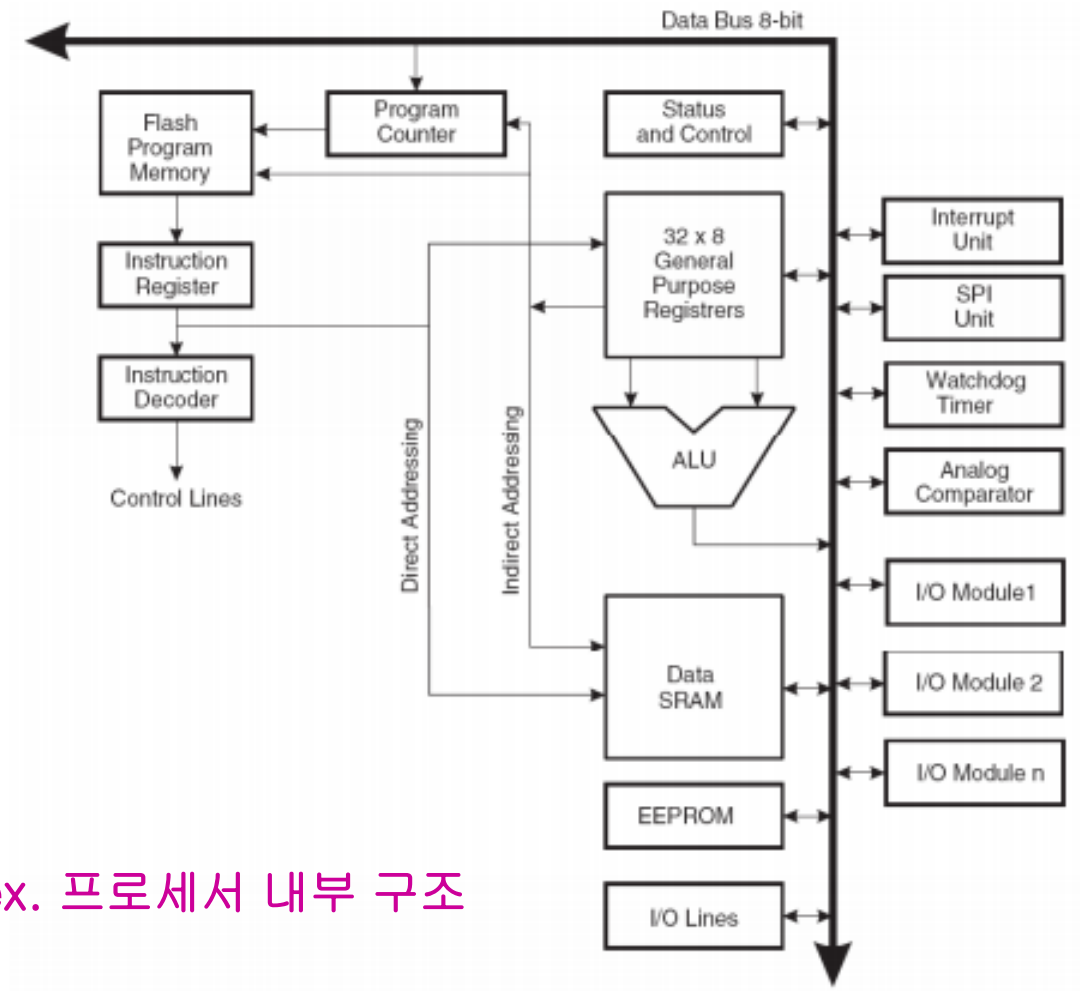
레지스터 :

- 프로세서 내부의 레지스터는 D 플립플롭의 병렬 배치
- 레지스터는 일종의 메모리
- 명령어, 데이터, 어드레스 등



ex. 4bit register

프로세서 내부 구조(예)



ex. 프로세서 내부 구조