디지털 논리회로

Chapter 1. 수의 체계와 변환

tb_elec_engineer@naver.com

1.1 디지털 시스템과 스위칭회로

- 디지털 시스템 : 계산 · 데이터 처리 , 제어시스템 , 통신 , 측정
  → 높은 정확도와 신뢰도
  - 아날로그 (연속적인 양)
  - 디지털 (이산적인 양) : 이진수 체계 → 비트 단위로 신호 처리
    전자기계를 이용하여 구현, 처리하기 용이

- 컴퓨터 (디지털) 의 논리
  - 이진 논리 : 0 (false) 과 1 (true)
  - Memory 의 유무에 따라 조합논리와 순차논리로 나뉨
  - 논리 연산
    - … 이고 (AND)
    - … 이거나 (OR)
    - … 가 아닌 (NOT)
  - 모든 것이 명확히 정의됨

- 조합회로와 순차회로
  - 조합회로 (combinational circuit)
    메모리 X
    ∴ 입력값에 따라 회로의 출력이 결정됨
    출력 = f (입력)
  - 순차회로 (sequential circuit)
    메모리 O
    메모리에는 회로의 현 상태가 저장
    ∴ 입력값과 현재상태에 따라 회로의 출력과 다음상태가 결정됨
    (출력, 다음상태) = f (입력, 현재상태)

· 논리회로 설계란 ?

: 기능이 주어질 때, 그 기능을 논리소자 (하드웨어)로 구현하는 것

↳ 문장, 말, pseudo code, program
진리표
카르노맵
minterm의 합, maxterm의 곱, …

tb_elec_engineer@naver.com

구현 방법

논리 회로 설계에서는 논리 게이트를 서로 연결하여 회로망을 형성하여 구현
하나의 기능을 구현하는 논리 회로는 많이 있다. 그 중 최적을 구하는 것이 중요

1.2 수 체계와 변환

· R 진수 → 10진법

$$1011.11_2 = 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 + 1\times2^{-1} + 1\times2^{-2}$$
$$= 8+0+2+1+\frac{1}{2}+\frac{1}{4} = 11.75_{10}$$

$$147.3_8 = 1\times8^2 + 4\times8^1 + 7\times8^0 + 3\times8^{-1}$$
$$= 64+32+7+\frac{3}{8} = 103.375_{10}$$

· 10진법 → R진법

$$N(\text{정수자리}) = (a_n a_{n-1}\cdots a_2 a_1 a_0)_R = a_n R^n + a_{n-1}R^{n-1} + \cdots + a_2R^2 + a_1R^1 + a_0$$

$$\frac{N}{R} = a_nR^{n-1} + a_{n-1}R^{n-2} + \cdots + a_2R^1 + a_1 = Q_1, \text{ 나머지 (remainder) } a_0$$

$$\frac{Q_1}{R} = a_nR^{n-2} + a_{n-1}R^{n-3} + \cdots + a_3R^1 + a_2 = Q_2, \text{ 나머지 } a_1$$

$$\vdots$$

$$F(\text{소수점 자리}) = (.a_{-1}a_{-2}a_{-3}\cdots a_{-m})_R = a_{-1}R^{-1} + a_{-2}R^{-2} + a_{-3}R^{-3} + \cdots + a_{-m}R^{-m}$$

$$FR = a_{-1} + a_{-2}R^{-1} + a_{-3}R^{-2} + \cdots + a_{-m}R^{-m+1} = a_{-1} + F_1$$

$$F_1R = a_{-2} + a_{-3}R^{-1} + \cdots + a_{-m}R^{-m+2} = a_{-2} + F_2$$

$$\vdots$$

예제) $53.625_{10} \rightarrow \square_2$

N(정수)

$2|53$
$2|26 \cdots 1 = a_0$
$2|13 \cdots 0 = a_1$
$2|6 \cdots 1 = a_2$
$2|3 \cdots 0 = a_3$
$2|1 \cdots 1 = a_4$
$0 \cdots 1 = a_5$ ↑

F(소수)

.625	.250	.500
× 2	× 2	× 2
1.250	0.500	1.000

$\therefore 53.625_{10} = 110101.101_2$

예제) $231.3_4 \rightarrow$ (10진수) $\rightarrow \square_7$

tb_elec_engineer@naver.com

$231.3_4 = 2\times4^2 + 3\times4^1 + 1\times4^0 + 3\times4^{-1} = 45.75_{10}$

$7|45$
$7|6 \cdots 3$
$0 \cdots 6$

.75	.25	.75	.25
× 7	× 7	× 7	× 7
5.25	1.75	5.25	1.75

$45.75_{10} = 63.5151\cdots_7$

· 2진수 → 8진수, 16진수

$\underbrace{101}_{2^2\ 1}\ \underbrace{011}_{1\times2\ 1}\ \underbrace{010}_{2}\ \underbrace{111}_{2^2\ 2\ 1}{}_2 = 5327_8$

$\underbrace{10}_{2}\ \underbrace{111}_{2^2\ 2\ 1}\ \underbrace{011}_{2\ 1}{}_2 = 273_8$

$\underbrace{1010}_{2^3\ 2}\ \underbrace{1111}_{2^3\ 2^2\ 2\ 1}\ \underbrace{0010}_{2}\ \underbrace{0101}_{2^2\ 1}{}_2 = AF25_{16}$

$\underbrace{11}_{2\ 1}\ \underbrace{0110}_{2^2\ 2}\ \underbrace{1000}_{2^3}{}_2 = 368_{16}$

$\underbrace{100}_{2^2}\ \underbrace{1101}_{2^3\ 2^2\ 1}.\ \underbrace{0101}_{2^2\ 1}\ \underbrace{11}_{2^3\ 2^2}{}_2 = 4D.5C_{16}$

※ 2진수 (Binary)
8진수 (Octal)
16진수 (Hexa)

□□
0
5
9
A (10)
B (11)
C (12)
D (13)
E (14)
F (15)

1.3 2진법 산술연산

tb_elec_engineer@naver.com

· 덧셈 (Addition)

$0+0=0$

$0+1=1$

$1+0=1$

$1+1=0$ → carry (자리올림)

ex) $13_{10} = 1101$

$11_{10} = +1011$

$11000_2 = 24_{10}$

· 뺄셈 (Subtraction)

$0-0=0$

$0-1=1$ → borrow (자리내림)

$1-0=1$

$1-1=0$

ex)

$$\begin{array}{r} 11101 \\ -\ 10011 \\ \hline 1010_2 \end{array} \qquad \begin{array}{r} 10000 \\ -\ \ \ 11 \\ \hline 1101_2 \end{array} \qquad \begin{array}{r} 11001 \\ -\ \ 1011 \\ \hline 10111_2 \end{array}$$

· 곱셈 (multiplication)

$0\times0=0$

$0\times1=0$

$1\times0=0$

$1\times1=1$

ex) $13_{10} = 1101$

$11_{10} = \times 1011$

$$\begin{array}{r} 1101 \\ 1101 \\ \hline (100111) \\ 0000 \\ 1101 \\ \hline 10001111_2 = 143_{10} \end{array}$$

$$\begin{array}{r} 1111 \\ \times\ 1101 \\ \hline 1111 \\ 0000 \\ 1111 \\ \hline 10010111 \\ 1111 \\ \hline 11000011_2 \end{array}$$

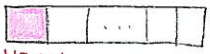
· 나눗셈 (Division)

$$\begin{array}{r|l} & 1101 \\ \hline 1011 & 10010001 \\ & 1011 \\ \hline & 1110 \\ & 1011 \\ \hline & 1101 \\ & 1011 \\ \hline & 10 \end{array}$$

1.4 음수의 표현

tb_elec_engineer@naver.com

- 무부호화수 (Unsigned number)
- 부호화수 (Signed number) :



부호자리
0 : +
1 : −

- 표현방법
  - 부호화된 크기 표현 방법 (signed magnitude)
  - 1의 보수 (1's Complement)
  - 2의 보수 (2's Complement)

⊛ 보수 : 보충해주는 수

→ 컴퓨터는 덧셈만 할 수 있어 빼기를 하기위해 보수를 사용

N 비트 수 표현 범위

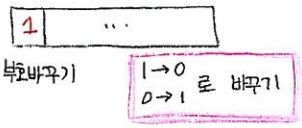
: $-(2^{(N-1)}-1) \sim +(2^{(N-1)}-1)$

- 1의 보수를 이용한 뺄셈과 덧셈 ($\overline{N}$)

n 비트에서 양의 정수 N에 대한 1의 보수 $\overline{N} = (2^n-1)-N$

1) 1의 보수로 표현하는 방법 (−X 표현하기)

11111111 − X



부호바꾸기 1→0, 0→1 로 바꾸기

1의 보수로 표현된 음수 $\overline{N}$ (−N)의 크기를 구하는 방법

→ 음수에 대해 1의 보수 취하기

⊛ 보수로 표현된 음수 −N이 주어지면, 이 음수에 대한 보수를 다시 취하면 그 음수에 대한 크기를 구할 수 있다.

2) 1의 보수의 덧셈

① 양수와 그보다 절대값이 작은 음수의 덧셈 (올림수 발생) → +1 ⇒ 끝단 돌림 (end-round) 아 자리올림

ex) 10 - 5 = 00001010 - 00000101

= 00001010 + 11111010

```
  00001010
+ 11111010      ⇒ 00000101 = 5
------------
 1 00000100 +1
```

자리올림 발생

tb_elec_engineer@naver.com

ex) -5 = 1010

+6 = 0110 ⇒ 0001 = 1

```
  1010
+ 0110
------
 10000
```

② 양수와 그보다 절대값이 큰 음수의 덧셈 (올림수 발생 x)

ex) 5 -10 = 00000101 - 00001010

= 00000101 + 11110101

```
  00000101
+ 11110101
----------
  11111010 = -5
```

→ -5임을 더 빠르 쉽게 알기 위해 1의 보수를 취해줌

10000101 0 = -5

~ 부호그대로

ex) +5 = 0101

-6 = 1001

```
  0101
+ 1001
------
  1110 = -1
```

③ over flow

두 음수의 덧셈 ≥ 2^{n-1}

-5 = 1010

+ -6 = 1001

```
  1010
+ 1001
------
 1 0011   ⇒ 0100
```

→ over flow로 인해 틀린 답

tb_elec_engineer@naver.com

⊕ 끝단돌림(자리올림)방법이 옳은 결과를 준다는 증명

① $-A+B \quad (B>A)$

$-A+B = \overline{A}+B = (2^n-1)-A+B = \underbrace{(B-A)}_{\text{결과값}} + \underbrace{2^n}_{\text{버리고}} - 1 \quad \cdots +1$

② $-A-B \quad (A+B<2^{n-1})$

$-A-B = \overline{A}+\overline{B} = (2^n-1-A)+(2^n-1-B) = \underbrace{(2^n-1-(A+B))}_{\text{결과값}} + \underbrace{2^n}_{\text{버리고}} - 1 \quad \cdots +1$

- 2의 보수를 이용한 뺄셈과 덧셈 (N^*)

n 비트에서 양의 정수 N 에 대한 2의 보수 $N^* = 2^n - N$

1) 2의 보수로 표현하는 방법

$100000000 - X$: 1의 보수 취하고 +1 ⊛ $N^* = 2^n - N = (2^n-1-N)+1 = \overline{N}+1$

2) 2의 보수의 덧셈

① 양수와 그보다 절대값이 작은 음수의 덧셈 (올림수 발생) → 무시

ex) $10-5 = 00001010 - 00000101$

$= 00001010 + (11111010+1)$

```
  00001010
+ 11111011
-----------
 1 00000101   ⇒ 00000101 = 5
(무시)
```

ex) $-5 = 1011$

$+6 = 0110 \quad \Rightarrow 0001 = 1$

```
  1011
+ 0110
------
 10001
```

② 양수와 그보다 절대값이 큰 음수의 덧셈 (올림수 발생 X)

ex) $5-10 = 00000101 - 00001010$

$= 00000101 + (11110101+1)$

$$\begin{array}{r} 00000101 \\ +\ 11110110 \\ \hline 11111011 \end{array}$$ → 읽기 쉽게하게 위해 2의 보수를 취함

$10000100+1 = 10000101$

$-0101 = -5$

ex) $+5 = 0101$

$$\begin{array}{r} +\ -6 = 1010 \\ \hline 1111 \end{array}$$

tb_elec_engineer@naver.com

③ overflow (두 음수의 덧셈(합) $> 2^{n-1}$)

$-5 = 1011$

$$\begin{array}{r} +\ -6 = 1010 \\ \hline \boxed{1}0101 \end{array}$$ → overflow로 인해 틀린 답

⊕ 부호비트에서 발생하는 자리올림을 무시하면 항상 옳은 답을 준다는 증명

① $-A+B$ $(B>A)$

$-A+B = (2^n-A)+B = 2^n+(B-A) > 2^n$

② $-A-B$ $(A+B < 2^{n-1})$

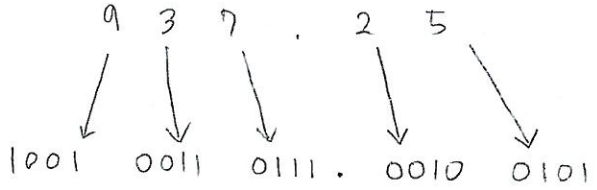
$-A-B = (2^n-A)+(2^n-B) = 2^n+\{2^n-(A+B)\}$

tb_elec_engineer@naver.com

1.5 2진 코드

10진수를 2진수로 표현하는 것 = 부호화 (coding)

→ 각 10진 숫자를 동등한 2진수로 대체시키는 것



10개의 10진수가 있기 때문에 1010 ~ 1111 까지는 존재 X

2진코드 조건 → 각 10개의 10진수에 대해 서로 다른 2진 숫자들의 조합으로 표현되어야 함

2진코드 종류
- 8-4-2-1 BCD Code
- 6-3-1-1 Code
- Excess-3 Code
- 2-out-of-5 Code

- 4비트 가중화 코드의 가중치가 W_3, W_2, W_1, W_0 일 때, 코드 a_3, a_2, a_1, a_0 는 N

$$N = W_3 a_3 + W_2 a_2 + W_1 a_1 + W_0 a_0$$

- 2진코드 예시
  - 2-out-of-5 Code : 모든 코드 조합에 대해서 5비트 중 2비트만 1을 가짐
  - Gray Code : 연속된 10진 숫자에 대한 코드 구성이 1비트 씩만 차이남

10진수	2-out-of-5 code	Gray code
0	00011	0000
1	00101	0001
2	00110	0011
3	01001	0010
4	01010	0110
5	01100	1110
⋮	⋮	⋮

Chapter 2. 부울대수

tb_elec_engineer@naver.com

• 부울대수 (Boolean algebra)

디지털 시스템의 논리설계를 위한 기본 수학

스위칭 대수 : 스위칭 회로의 활용에 한정 (0, 1)

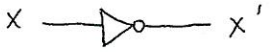
부울 변수 : X, Y 는 오직 두개의 다른 값 (0,1) 을 갖는다

→ True (1), False (0)

2.2 기본연산

① NOT (Inverter) 연산 (논리 부정)

- $0' = 1$, $1' = 0$
- $X' = 1$ if $X = 0$, $X' = 0$ if $X = 1$
- 기호 :



② AND 연산 (논리곱)

- $0 \cdot 0 = 0$ $0 \cdot 1 = 0$ $1 \cdot 0 = 0$ $1 \cdot 1 = 1$
- 진리표

A	B	C = A · B
0	0	0
0	1	0
1	0	0
1	1	1

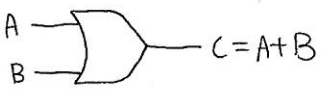
- 기호 :

③ OR 연산 (논리합)

- $0 + 0 = 0$ $0 + 1 = 1$ $1 + 0 = 1$ $1 + 1 = 1$
- 진리표

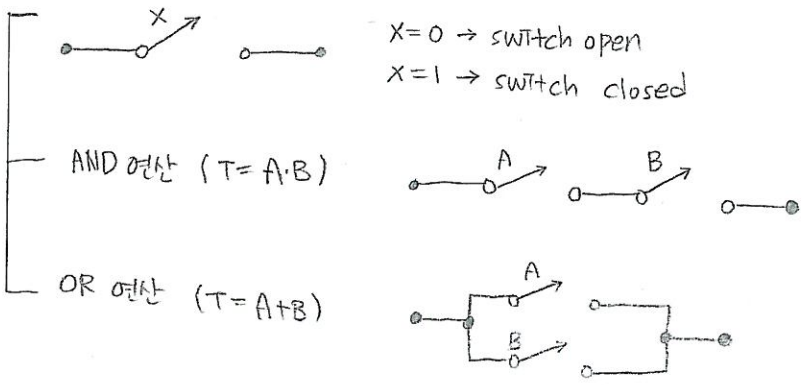
A	B	C = A + B
0	0	0
0	1	1
1	0	1
1	1	1

- 기호 :



tb_elec_engineer@naver.com

• 스위치로 표현

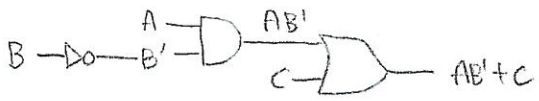


2.3 부울식과 진리표

부울식 : 하나 또는 그 이상의 부울 변수 (A, B, X, Y…) 와 상수 (0 or 1) 그리고 기본연산 (AND, OR, NOT) 으로 구성된 식

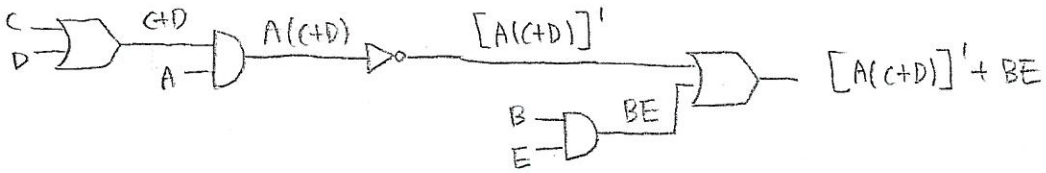
ex) 논리표현 : $AB' + C$

논리게이트 회로 (순서 : ① 괄호 ② NOT ③ AND ④ OR)



ex) 논리표현 : $[A(C+D)]' + BE$

논리 게이트 회로

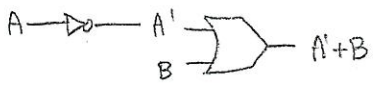


진리표 (Truth Table) or 조합표

: 부울식에 있는 부울 변수들 값의 가능한 모든 조합에 대한 부울식의 값을 나타냄

n 변수 식의 진리표는 2^n 개의 행으로 나타남

입력 회로와 진리표



A	B	A'	A'+B
0	0	1	1
0	1	1	1
1	0	0	0
1	1	0	1

tb_elec_engineer@naver.com

진리표를 이용한 증명

$AB' + C = (A+C)(B'+C)$

A	B	C	B'	AB'	AB'+C	A+C	B'+C	(A+C)(B'+C)
0	0	0	1	0	0	0	1	0
0	0	1	1	0	1	1	1	1
0	1	0	0	0	0	0	0	0
0	1	1	0	0	1	1	1	1
1	0	0	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1
1	1	0	0	0	0	1	0	0
1	1	1	0	0	1	1	1	1

⇒ 같음

2.4 기본 정리

- 0, 1 연산 : $X+0 = X$ $X+1 = 1$ $X \cdot 1 = X$ $X \cdot 0 = 0$
- 멱등 법칙 : $X+X = X$ $X \cdot X = X$
- 누승의 법칙 : $(X')' = X$
- 상보의 법칙 : $X+X' = 1$ $X \cdot X' = 0$

증명) $X=0 \rightarrow 0+0' = 0+1$

$X=1 \rightarrow 1+1' = 1+0 = 1$

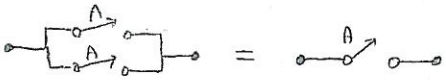
ex) $(AB'+D)E + 1 = 1$

$(AB'+D)(AB'+D)' = 0$

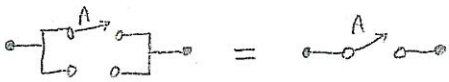
스위칭 회로를 이용하여 증명

$A \cdot A = A$

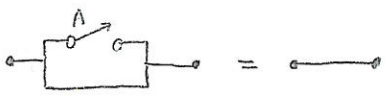
$A+A = A$



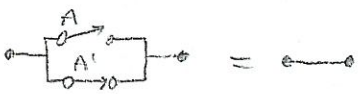
$A+0 = A$



$A+1 = 1$



$A+A' = 1$



$A \cdot A' = 0$

2.5 교환, 결합, 분배법칙

tb_elec_engineer@naver.com

- 교환법칙 : $XY = YX$ $\quad X+Y = Y+X$
- 결합법칙 : $(XY)Z = X(YZ) = XYZ$

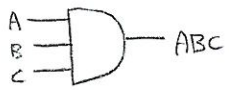
$(X+Y)+Z = X+(Y+Z) = X+Y+Z$

AND에 대한 결합법칙 증명

X	Y	Z	XY	YZ	(XY)Z	X(YZ)
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	1	0	0
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	1	0	0	0
1	1	1	1	1	1	1

AND와 OR에 대한 결합 법칙

- $(AB)C = ABC$



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

- AND 연산 : 부울변수들이 모두 AND 연산 → 모든 값이 1 = 1
  변수들 中 어떤 것이 0 = 0
  $X=Y=Z=1$ 일 때, $XYZ=1$
- OR 연산 : 부울변수들이 모두 OR 연산 → 모든 변수가 0 = 0
  변수 들 中 어느 하나 1 = 1

부울대수에서의 분배법칙 (제 1법칙) : $X(Y+Z) = XY + XZ$

부울대수에서의 분배법칙 (제 2법칙) : $X+(YZ) = (X+Y)(X+Z)$

※ 증명

$$(X+Y)(X+Z) = X(X+Z) + Y(X+Z) = XX + XZ + YX + YZ$$
$$= X + XZ + XY + YZ = X(1+Z+Y) + YZ$$
$$= X + YZ$$

tb_elec_engineer@naver.com

2.6 간략화 정리

부울대수에서의 간략화 법칙

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

※ 증명

$XY + XY' = X(Y+Y') = X \cdot 1$

$(X+Y)(X+Y') = X + YY' = X + 0 = X$

$X + XY = X(1+Y) = X \cdot 1$

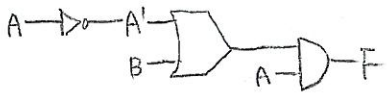
$X(X+Y) = XX + XY = X + XY = X$

$(X+Y')Y = XY + Y'Y = XY + 0 = XY$

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

등가회로 게이트

$F = A(A'+B) = AB$

초자) 

고수) $F = AA' + AB = 0 + AB = AB$

⇒ 간략화된 회로를 찾는것이 목표

ex) $Z = A'BC + A'$ 를 간략화

→ $Z = A'(BC+1) = A' \cdot 1 = A'$

or $A'(BC) + A' = A'$

tb_elec_engineer@naver.com

ex) $Z = [A+B'C + D+EF][A+B'C + (D+EF)']$ 를 간략화

→ $Z = [(A+B'C) + (D+EF)][(A+B'C) + (D+EF)']$

$= [X+Y][X+Y']$

$= X$

$= A+B'C$

ex) $Z = (AB+C)(B'D+C'E') + (AB+C)'$ 를 간략화

→ $Z = Y'X + Y$

$= X+Y$

$= (B'D+C'E') + (AB+C)'$

2.7 곱셈전개와 인수화

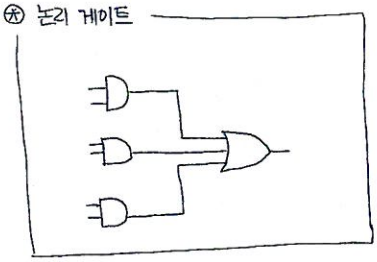
- 논리곱의 합 (Sum of Product) = '전개해라'

: 모든 곱들이 단 하나의 곱으로 이루어 질 때

ex) $AB'+CD'E+AC'E$

$A+B'+C+D'E$

※ 논리 게이트



논리 곱의 합 형식으로 표현하기 위해 곱셈전개 (Multiplying Out)을 이용

→ 제 2 분배 법칙 적용

ex) $(A+BC)(A+D+E) = [A+(BC)][A+(D+E)] = A+BC(D+E)$

$= A+BCD+BCE$

$(X+Y)(X+Z) = X+YZ$

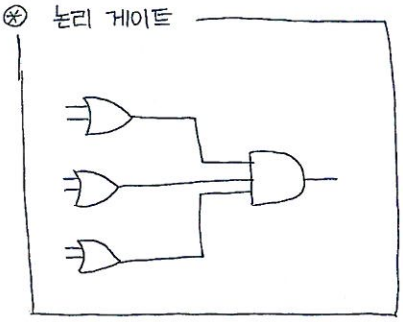
• 논리합의 곱 (Product of Sum) = '인수화 해라'

: 모든 곱들이 단일 변수들의 합으로 이루어질 때

ex) $(A+B)(C+D+E)F$

$A'B'C'(D'+E)$

※ 논리 게이트



ex) $A+B'CD$ 를 인수화

→ $A+B'CD = A+(B')CD = (A+B')(A+CD)$

$= (A+B')(A+C)(A+D)$

tb_elec_engineer@naver.com

ex) $AB'+C'D$ 를 인수화

→ $AB'+C'D = (AB'+C')(AB'+D)$

$= (A+C')(B'+C')(A+D)(B'+D)$

ex) $C'D+C'E'+G'H$ 를 인수화

→ $C'D+C'E'+G'H = C'(D+E')+G'H$

$= (C'+G'H)(D+E'+G'H)$

$= (C'+G')(C'+H)(D+E'+G')(D+E'+H)$

2.8 드모건 법칙 (DeMorgan)

+ → · · → +

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

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

진리표에 의한 증명

X	Y	X'	Y'	X+Y	(X+Y)'	X'Y'	XY	(XY)'	X'+Y'
0	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	0	0	1	1
1	0	0	1	1	0	0	0	1	1
1	1	0	0	1	0	0	1	0	0

n 변수에 대해서도 드모건 법칙이 성립한다.

ex) $(A'+B)C'$ 의 보수를 구하라 → Not 구하라 (드모건 법칙 이용)

→ $[(A'+B)C']' = (A'+B)'+C = AB'+C$

ex) $[(AB'+C)D'+E]'$ 를 구하라

→ $[(AB'+C)D']'E' = [(AB'+C)'+D]E'$

$= [(AB')'C'+D]E' = [(A'+B)C'+D]E'$

쌍대 (Dual)

주어진 부울식의 쌍대를 만든다

: AND → OR

OR → AND

1 → 0

0 → 1

tb_elec_engineer@naver.com

순서 ① 주어진 부울식에 보수를 취한다

② 개개의 변수에 보수를 취하며 구한다

ex) $AB'+C$ 의 쌍대를 구하라

→ $(AB'+C)' = (AB')'C' = (A'+B)C'$

$\therefore (AB'+C)^D = (A+B')C$