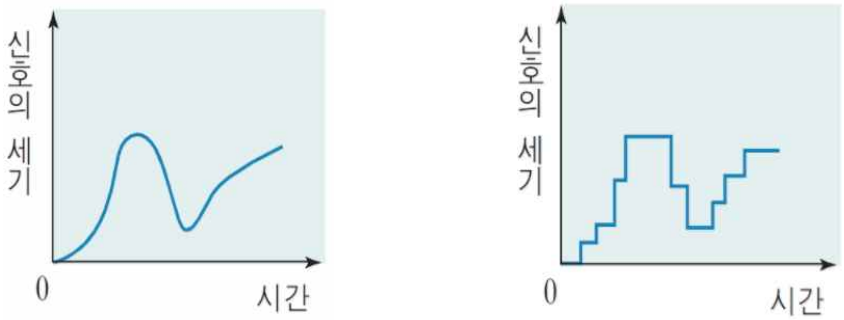
아날로그와 디지털

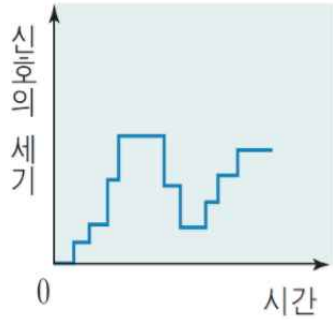
1. 아날로그 신호

- **아날로그 신호 (Analog Signal)**
  - 시간에 따라 연속적으로(continuous) 변하는 값
  - 정해진 범위내에서 어떠한 값으로도 표현가능
  - 주로 물리량 표현에 사용 : 전압, 전류, 온도, 습도 등



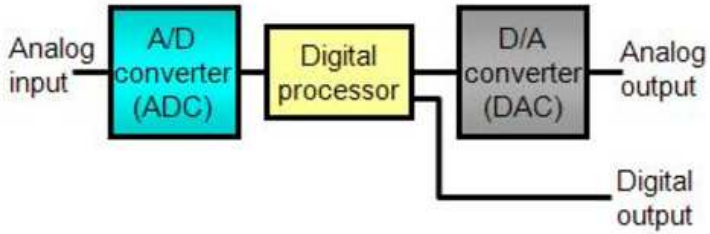
2. 디지털 신호

- **디지털 신호 (Digital Signal)**
  - 시간에 따라 불연속적인(discrete) 값
  - 정해진 범위내에서 특정 값으로만 표현가능
  - 연속적인 아날로그 값을 디지털로 변환하여 사용
  - 장점
    - 처리하기 용이하다.
    - 잡음에 강하다.
    - 시스템 구현이 쉽다.
  - 단점
    - 아날로그 신호를 완벽하게 표현할 수 없다. (변환오류가 필연적 발생)



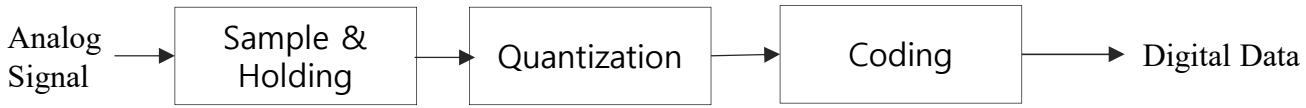
3. 아날로그 신호의 디지털 변환

- **아날로그 신호의 디지털 변환 회로**
  - ADC(Analog-to-Digital Conversion : ADC)
    - 물리량을 표시하는 아날로그 전기 신호를 디지털 신호로 변환하는 회로
  - DAC(Digital-to-Analog Converter)
    - 디지털 신호를 연속적인 아날로그 신호로 변환하는 회로



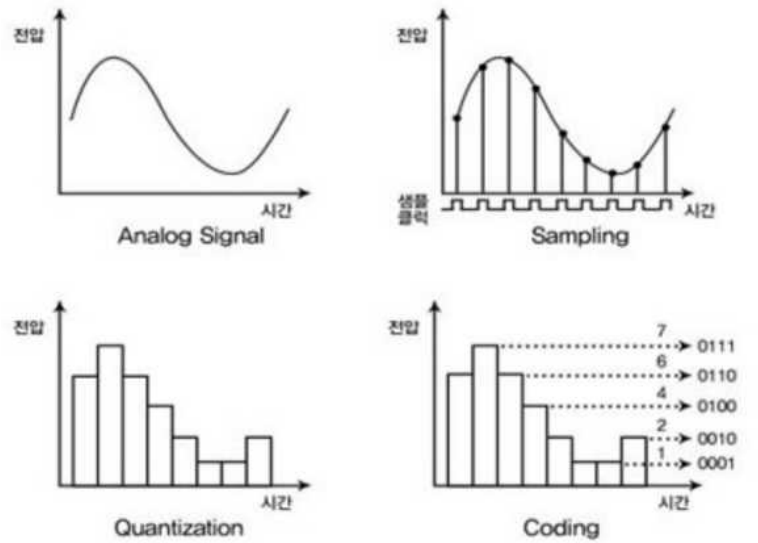
3.1 아날로그 신호의 디지털 변환과정

- **Sample & Holding**
  - 연속적으로 변화하는 아날로그 신호를 샘플링하여 일정시간동안 유지하는 것.
    - Sampling time
- **Quantization**
  - 샘플한 신호를 디지털 데이터로 표현.
    - Resolution



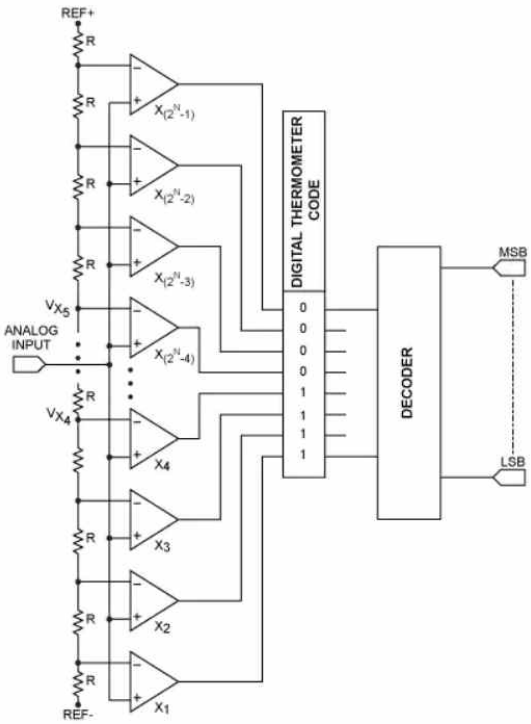
3.2 ADC 동작 예시

- **Sampling → Quantization → Coding**



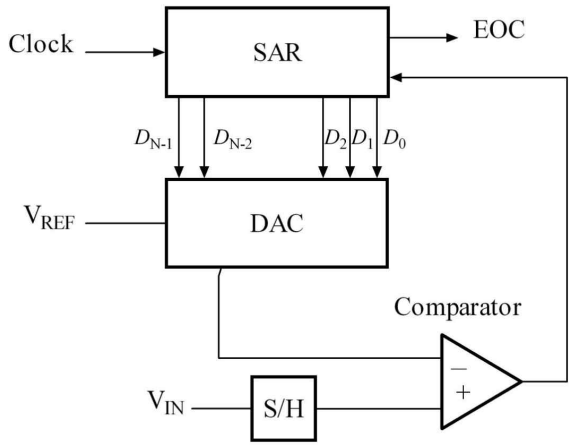
3.3 ADC 방법

- **Direct-Conversion ADC**
  - 고속의 비교기(Comparator)와 기준 전압 생성을 위한 전압 분배 회로(Resistive-divider)로 구성하는 ADC.
    - N-bit 변환기 : 2N-1 개의 비교기 + 2N 개의 저항을 사용한 전압 분배회로
  - 가장 빠른 변환 방법
  - 가장 많은 하드웨어 사용
  - 높은 전력 소모
  - Flash ADC 또는 Parallel ADC 로 알려짐.



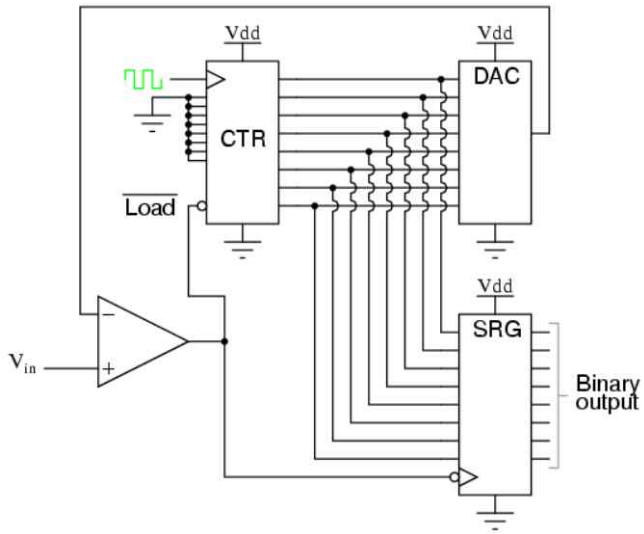
3.3 ADC 방법

- **Successive-approximation ADC**
  - DAC 와 비교기를 사용하여 MSB부터 LSB까지 추정하면서 변환.
    - EOC : End of Conversion
    - SAR : SA Register
    - S/H : Sample & Holder
  - 비교적 간단한 회로
  - 긴 변환시간



3.3 ADC 방법

- **Ramp-compare ADC**
  - Free-running 이진 카운터와 DAC, 비교기로 구성.
    - CTR : Binary Counter
    - DAC : Digital-to-Analog Converter
    - SRG : Shift Register
  - 변환 값에 따라 변환 속도 차이가 발생 → 안정적 변환 주기 확보 어려움.
    - 아날로그 입력 전압이 높으면, 변환 시간 길어지고, 낮으면, 변환시간 짧음.
  - Stairstep-Ramp 또는 Counter ADC 로 알려짐.



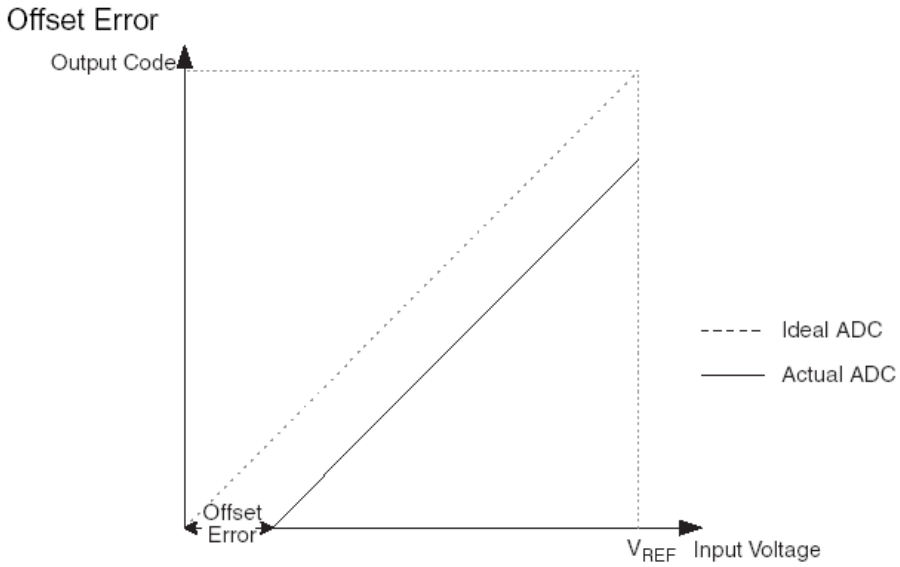
3.4 ADC 변환 에러

- **양자화 에러 (quantization error)**
  - 연속적인 아날로그 신호를 유한한 비연속적인 디지털 코드로 양자화하기 때문에 필연적으로 발생.

- **절대 에러 (absolute accuracy)**
  - 실제 값과 변환 값의 최대 차이값.
  - Offset 에러, Gain 에러, Integral/Differential Non-linearity 에러로 표시.

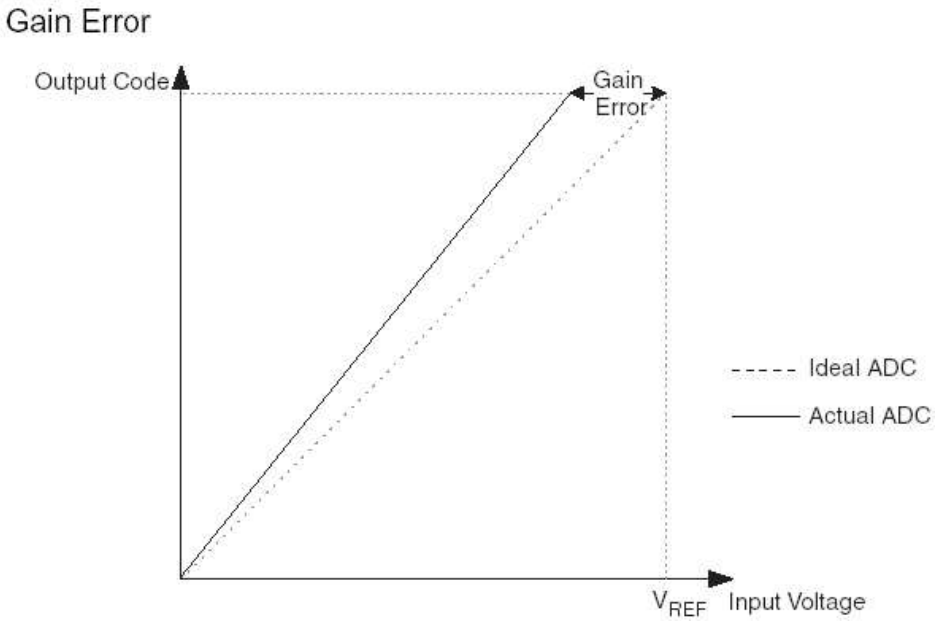
3.4.1 Offset 에러

- 첫번째 변환(0x00 에서 0x01) 시, 실제 값과 변환 값 사이의 오차.



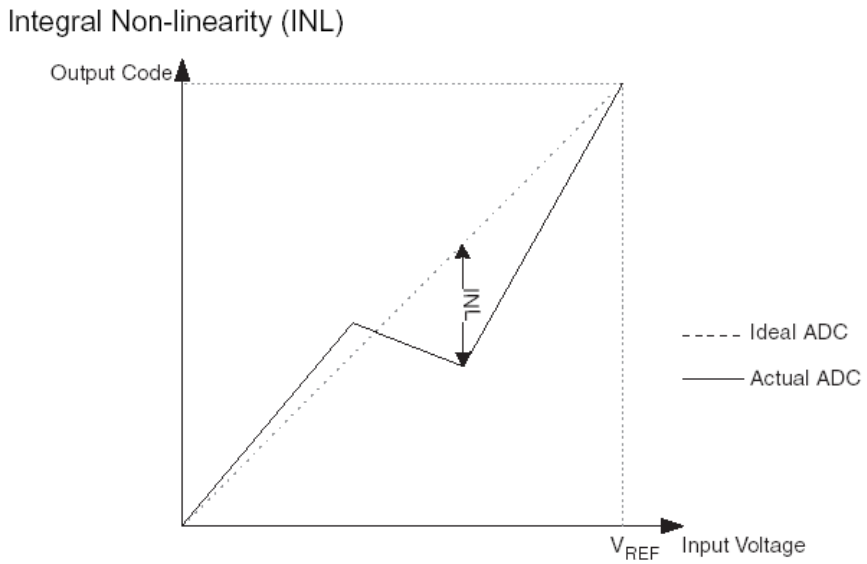
3.4.2 Gain 에러

- 마지막 변환(0xFE 에서 0xFF) 시, 실제 값과 변환 값 사이의 오차.



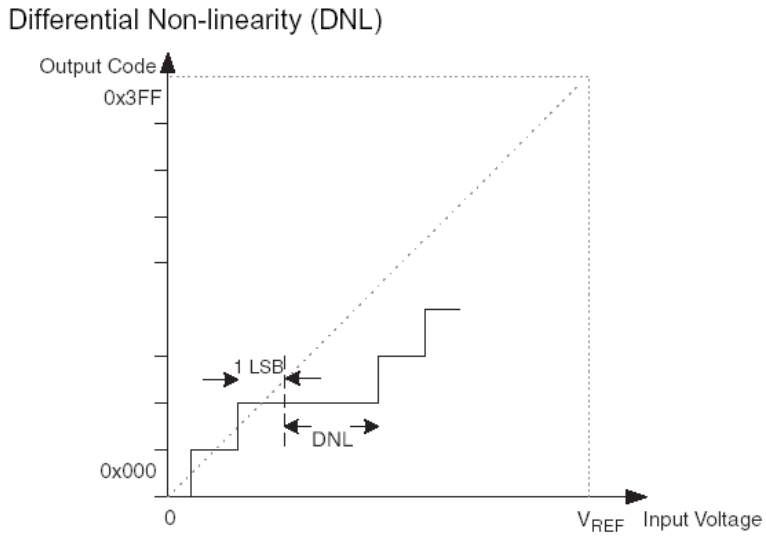
3.4.3 INL 에러

- **Integral Non-linearity (INL)**
  - 변환과정에서 발생한 실제 값과 변환 값의 최대 차이값.
  - ADC의 선형성(Linearity) 오류



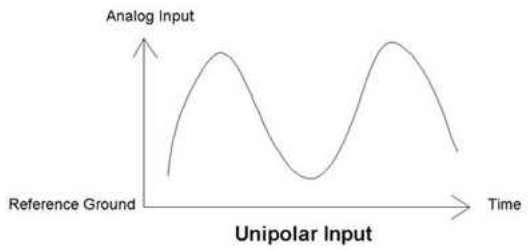
3.4.4 DNL 에러

- **Differential Non-linearity (DNL)**
  - 실제 코드 변환폭과 변환시 발생한 변환폭의 최대 차이값.
    - 변환폭 : 다음 변환까지의 입력신호 변동폭

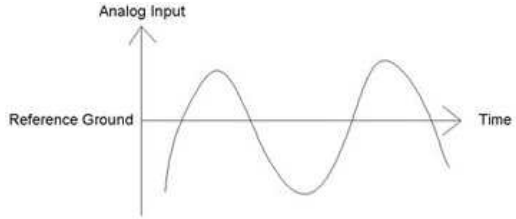


3.5 ADC 선택 기준

- 변환 속도 (Conversion Speed) : <100us (고가)
- 해상도(Resolution) : 8-, 10-, 12-bit 가 일반적
- 출력레벨 : TTL, CMOS
- 안정도 : 온도 변화에 따른 변환 안정성
- 아날로그 입력 범위
- 아날로그 입력 극성(Polarity) : unipolar, bipolar



Unipolar Input

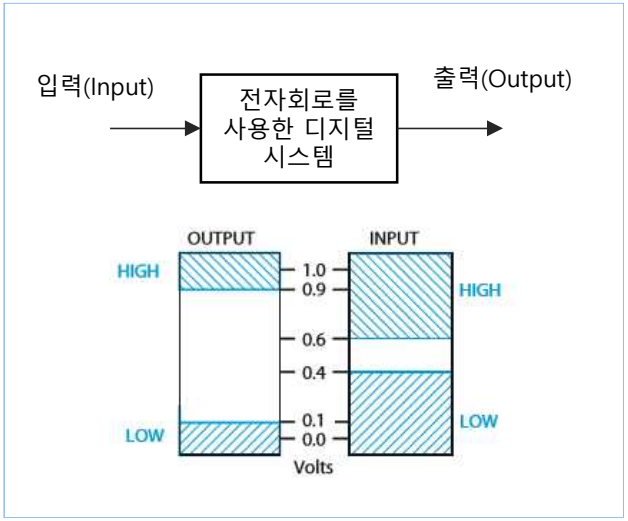


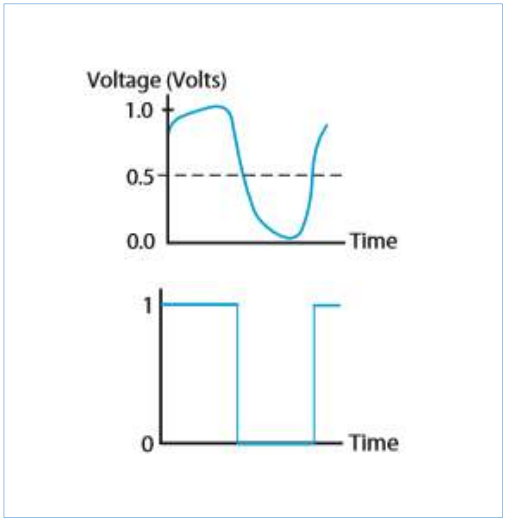
Bipolar Input

이진법

1. 바이너리

- **바이너리(Binary) : 2개의 상태(0과 1)로 표현되는 숫자, 이진(二進)**
- **2진수(Binary Digit) : 바이너리로 표현되는 숫자.**
- **전자회로를 사용한 바이너리 숫자의 표현**
  - 전자회로의 전압 값으로 2개의 상태(HIGH, LOW)를 구분





2. 2진법

- **2진법**
  - 두 개의 숫자(0, 1)만으로 표현되는 수 체계, 즉 2진수로 표현되는 수 체계
    - 밑수(Base, Radix)가 2인 수체계
    - Base 또는 Radix를 밑, 기수, 기저라 함.

$$A_{n-1}A_{n-2}\ldots A_1A_0.A_{-1}A_{-2}\ldots A_{-m+1}A_{-m}$$

$$A_{n-1}r^{n-1} + A_{n-2}r^{n-2} + \ldots + A_1r^1 + A_0r^0 + A_{-1}r^{-1} + A_{-2}r^{-2} + \ldots + A_{-m+1}r^{-m+1} + A_{-m}r^{-m}$$

10진수(Decimal) 의 표현

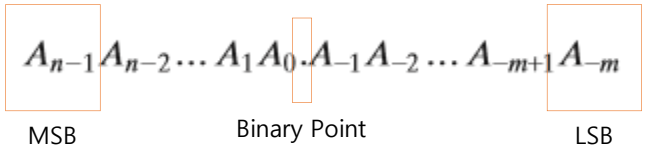
$$724.5 = 7 \times 10^2 + 2 \times 10^1 + 4 \times 10^0 + 5 \times 10^{-1}$$

2진수(Binary) 의 표현

$$(110101.11)_2 = 32 + 16 + 4 + 1 + 0.5 + 0.25 = (53.75)_{10}$$

2.1 2진 소수점

- **2진 소수점(Binary Point)**
  - 2진 분수를 표현하는 기준점
  - 10진 소수점 (Decimal point), 2진 소수점(binary point)
- **단위를 표시하는 콤마(,) 를 사용하지 않음 : 1,000 (x)**
- **MSB(Most Significant Bit)**
  - 2진 숫자로 표현된 값의 최상위 자리의 수(Most Significant Digit: MSD)
- **LSB (Least Significant Bit)**
  - 2진 숫자로 표현된 값의 최하위 자리의 수



2.2 2진 단위 표시

- **2진 단위 표시**
  - 국제단위계(SI)를 사용
    - 10진법에서 사용하는 단위체계를 2진법에서 동일하게 사용.
  - 10진법과 2진법의 결과값이 정확하게 일치하지 않음
    - 1K는 1,000 이지만, 2진법에서는 1,024에 해당.

$$4K = 2^2 \times 2^{10} = 2^{12} = 4096$$

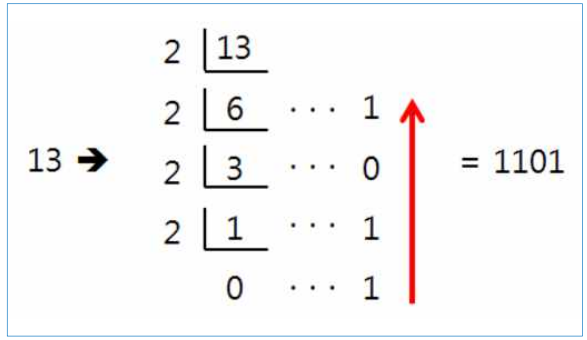
$$16M = 2^4 \times 2^{20} = 2^{24} = 16,777,216$$

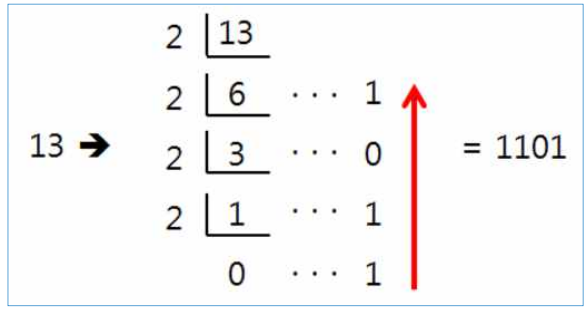
10^n	접두어	기호	배수	십진수
10^{24}	요타 (yotta)	Y	자	1 000 000 000 000 000 000 000 000
10^{21}	제타 (zetta)	Z	십해	1 000 000 000 000 000 000 000
10^{18}	엑사 (exa)	E	백경	1 000 000 000 000 000 000
10^{15}	페타 (peta)	P	천조	1 000 000 000 000 000
10^{12}	테라 (tera)	T	조	1 000 000 000 000
10^{9}	기가 (giga)	G	십억	1 000 000 000
10^{6}	메가 (mega)	M	백만	1 000 000
10^{3}	킬로 (kilo)	k	천	1 000
10^{2}	헥토 (hecto)	h	백	100
10^{1}	데카 (deca)	da	십	10
10^{0}			일	1

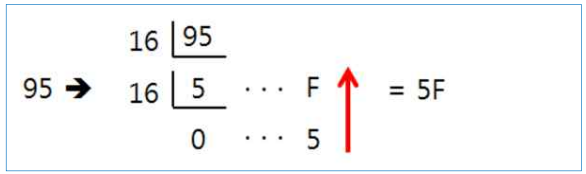
2.3 2진법 변환

- **10진수를 2진수로 변환**
  - 10진수를 '2'로 연속적으로 나누어, 각 단계의 나머지를 취함.
  - 2진수로 표시할 경우 상위 비트는 '0' 으로 채움

- **8진, 16진법**
  - 2진을 구한 후, 3 비트, 또는 4 비트 단위로 묶어서 표현







2.4 2진법 연산

- 2진수의 각 자리 수를 더한 후 결과에서 발생하는 나머지는 적고, 올림(carry)은 상위 자리수와 함께 더한다.

Carries:	00000	101100
Augend:	01100	10110
Addend:	+10001	+10111
Sum:	11101	101101

Multiplicand:	1011
Multiplier:	× 101
	1011
	0000
	1011
Product:	110111

Borrows:	00000	00110	00110
Minuend:	10110	10110	11110
Subtrahend:	−10010	−10011	−10011
Difference:	00100	00011	−01011

```
        1 0 1 1
     -------------
1 0 1 ) 1 1 0 1 1 1
      - 1 0 1
        -------------
            1 1
            1 1 1   ....모자람
            1 0 1
            --------
              1 0 1
            - 1 0 1
            ==========
                  0
```

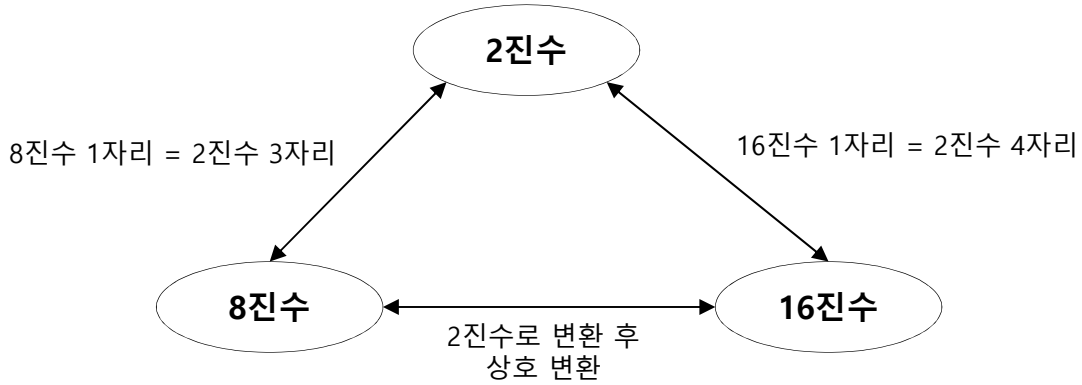
3. 8진/16진법

- **전자 계산기에서 2진수를 읽기 쉽게 8진(Octal) 또는 16진수(Hexadecimal) 로 표현**
- **프로그램에서의 진법표시**
  - 접두어 또는 접미어 사용
    - 2진 : b, b00000000
    - 8진 : 0, 004567
    - 16진 : 0x, 0x89ABCF

Decimal (base 10)	Binary (base 2)	Octal (base 8)	Hexadecimal (base 16)
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

3.1 진법 변환

- 2진수를 8진수로 변환 : 2진 3-비트를 8진-1자리 수(0~7)로 변환.
- 2진수를 16진수로 변환 : 2진 4-비트를 16진-1자리 수(0~F)로 변환.
- 8/16진수를 2진수로 변환 : 8/1 6진수 1자리를 3-/4-비트로 변환
- 8진수를 16진수로 변환 : 8진수를 2진수로 변환후, 16진수로 변환



논리회로 표현

1. 부울 대수

- **부울 대수 (Boolean Algebra)**
  - 변수 값이 참 또는 거짓, 0 또는 1로 정의되는 대수
  - 1847 영국 수학자 George Boole이 정리한 참(0)과 거짓(1)에 대한 수학적 체계.
  - 1938 벨 연구소 Shannon에 의해 전기적 스위칭 회로가 불 대수를 사용하여 표현 가능함에 따라 스위칭 대수(Switching Algebra)로 알려짐.

1.1 부울 연산

- **부울 연산(Boolean Operation)**
  - 부울 대수에서 정의한 연산
  - 기본(Basic) 연산과 2차(Secondary) 연산으로 구성
- **기본 연산**
  - AND (Conjunction) : x, y 가 모두 '1' 일 때, 연산 결과(x ^ y)는 '1'
  - OR (Disjunction) : x, y 가 모두 '0' 일 때, 연산 결과(x v y)는 '0'
  - NOT (Negation) : x 가 '0' 이면, 연산 결과(¬x) 는 '1', '1' 이면 '0'

x	y	$x \wedge y$	$x \vee y$
0	**0**	0	0
1	**0**	0	1
0	**1**	0	1
1	**1**	1	1

x	$\neg x$
0	1
1	0

$$x \wedge y = xy = \min(x, y)$$
$$x \vee y = x + y - xy = \max(x, y)$$
$$\neg x = 1 - x$$

$$x \wedge y = \neg(\neg x \vee \neg y)$$
$$x \vee y = \neg(\neg x \wedge \neg y)$$

1.1 부울 연산

- **2차 연산**
  - Material Implication : x가 '1' 이면, 결과는 'y'를 따르고, x가 '1' 이면 결과는 '1'.
  - Exclusive OR (XOR) : x, y 가 서로 다른 값을 가지면, '1'
  - Equivalence : x 와 y 가 같은 값을 가질 때, '1', Exclusive의 보수(XNOR).

$$x \to y = \neg x \vee y$$
$$x \oplus y = (x \vee y) \wedge \neg(x \wedge y)$$
$$x \equiv y = \neg(x \oplus y)$$

x	y	$x \to y$	$x \oplus y$	$x \equiv y$
0	**0**	1	0	1
1	**0**	0	1	0
0	**1**	1	1	0
1	**1**	1	0	1

1.2 부울 법칙

OR	AND
$X + 0 = X$	$X \cdot 1 = X$
$X + 1 = 1$	$X \cdot 0 = 0$
$X + X = X$	$X \cdot X = X$
$X + \overline{X} = 1$	$X \cdot \overline{X} = 0$
$\overline{\overline{X}} = X$	

X AND Y

$X + Y = Y + X$	$XY = YX$
$X + (Y + Z) = (X + Y) + Z$	$X(YZ) = (XY)Z$
$X(Y + Z) = XY + XZ$	$X + YZ = (X + Y)(X + Z)$
$\overline{X + Y} = \overline{X} \cdot \overline{Y}$	$\overline{X \cdot Y} = \overline{X} + \overline{Y}$

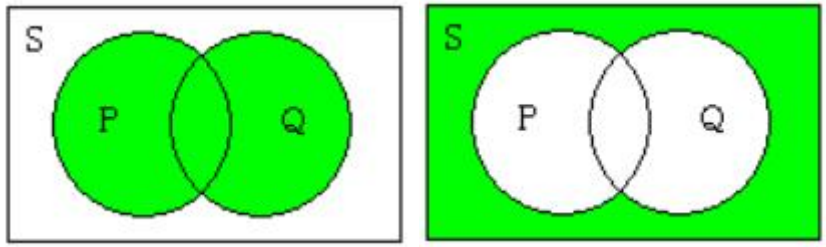
1.2.1 드 모르강의 법칙

- **De Morgan's Law**
  - 논리 곱, 논리 합, 부정 연산간의 관계를 정리한 법칙

논리기호를 사용한 표현

$$\neg(P \vee Q) = \neg P \wedge \neg Q$$
$$\neg(P \wedge Q) = \neg P \vee \neg Q$$

벤 다이어그램을 사용한 표현



집합기호를 사용한 표현

$$\overline{(A \cap B)} = \overline{A} \cup \overline{B}$$
$$\overline{(A \cup B)} = \overline{A} \cap \overline{B}$$

논리회로를 사용한 표현

$$\overline{(A + B)} = \overline{A} * \overline{B}$$
$$\overline{(A * B)} = \overline{A} + \overline{B}$$

1.3 부울 방정식

- **부울 방정식(Boolean Equation)**
  - 입력 변수와 출력 변수의 관계를 부울 연산으로 표현한 방정식
    - 바이너리 변수와 논리 연산자를 사용한 표현
  - 부울 표현(Boolean Expression), 부울 함수(Boolean Function)와 혼용.

입력 변수 : X, Y, Z
출력 변수 : F

$$F = \overline{X}YZ + \overline{X}Y\overline{Z} + XZ$$

2. 진리표

- **진리표(Truth Table)**
  - 논리 변수 값들의 조합과 각 변수 값 조합에 대한 논리연산 결과 값을 정의한 표

AND

X	Y	$Z = X \cdot Y$
0	0	0
0	1	0
1	0	0
1	1	1

OR

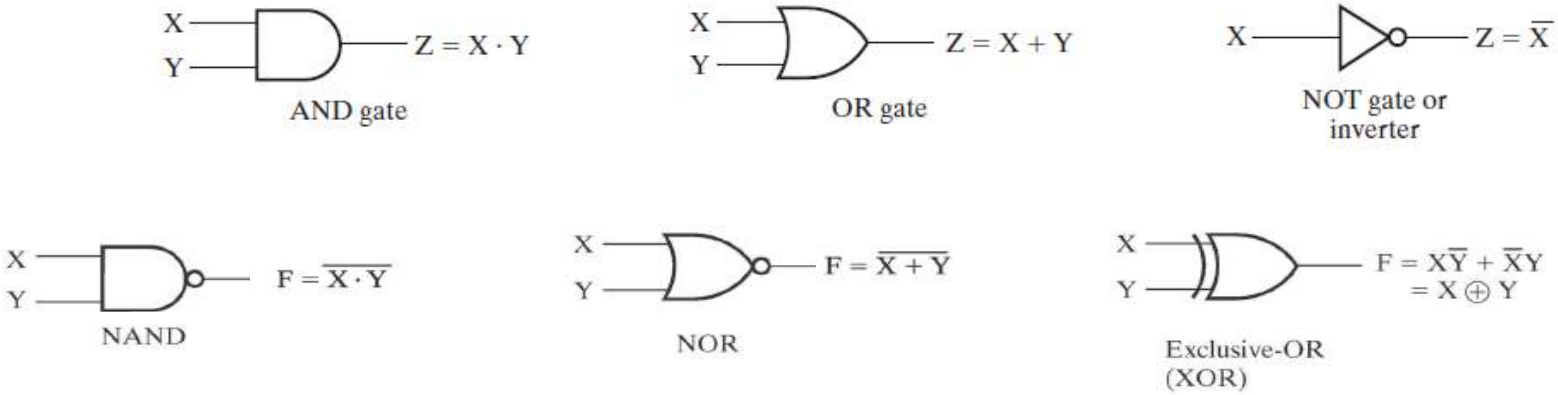
X	Y	$Z = X + Y$
0	0	0
0	1	1
1	0	1
1	1	1

NOT

X	$Z = \overline{X}$
0	1
1	0

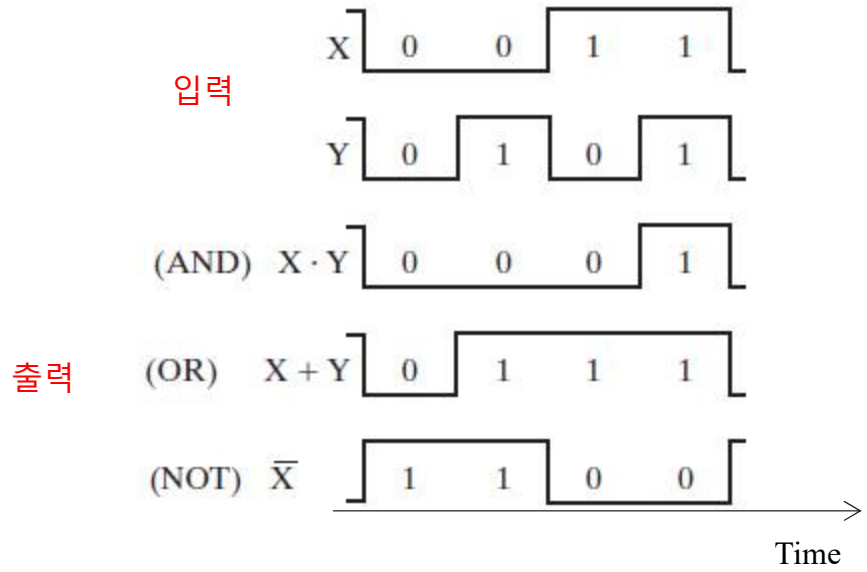
3. 논리 게이트

- **논리 게이트(Logic Gate)**
  - 기본 부울 연산을 구현한 전자회로
    - 트랜지스터를 사용한 집적회로 형태로 구현
    - 그래픽 심볼로 표시



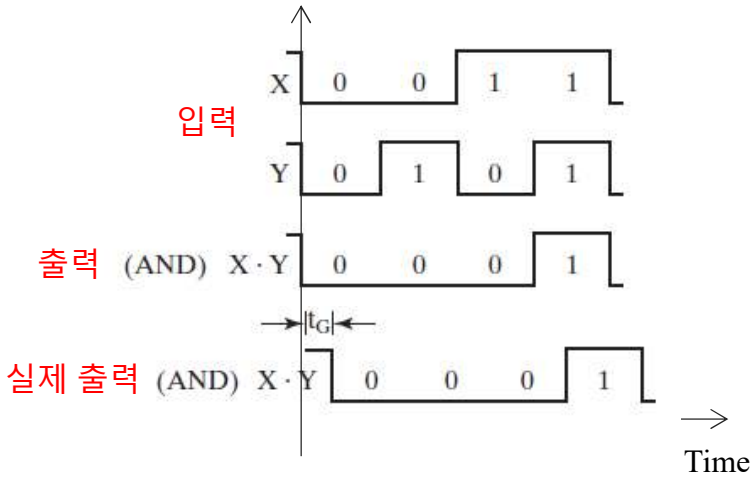
3.1 타이밍 다이어그램

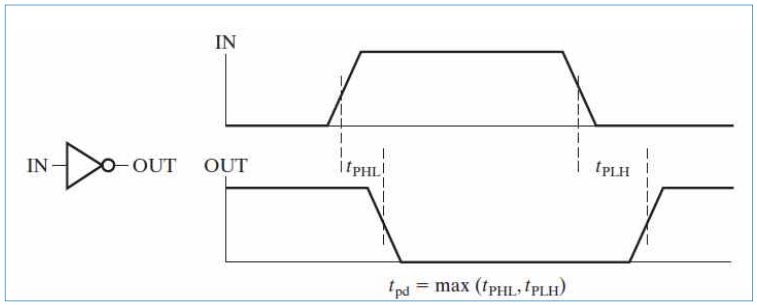
- **시간에 따른 입력 신호와 출력신호의 변화를 보여주는 그래프**
  - 가로축 : 시간
  - 세로축 : 입력 출력 신호의 전압/전류 파형(Waveform)



3.2 전파 지연

- **입력신호 변화에 따른 출력신호의 변화가 발생하는 시간차이**
  - 게이트 지연(Gate Delay) 또는 전파 지연(Propagation Delay)라고 함.
  - 논리회로의 처리 시간지연





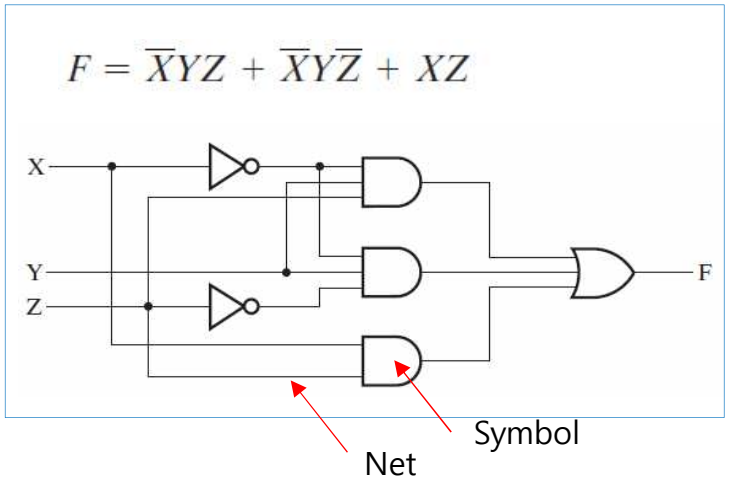
4. 논리 회로 표현

- **논리 회로**
  - 논리 회로 구성요소들과 그들간의 상호 연결 관계를 표현한 회로
    - 논리회로 구성요소 : 조합회로, 순차회로

- **논리 회로 표현방법**
  - Boolean Equation : 논리회로를 부울 방정식으로 표현
  - Schematic : 논리 회로 구성요소들을 그래픽 심볼을 사용하여 표현한 논리회로
  - HDL Description : 논리 회로 구성요소들을 하드웨어 기술언어(HDL : Hardware Description Language)를 사용하여 표현한 논리회로.

4.1 논리회로의 Schematic 표현

- **논리 회로를 논리 게이트 심볼들과 상호 연결 관계로 표현**
  - 상호 연결 관계 : Netlist



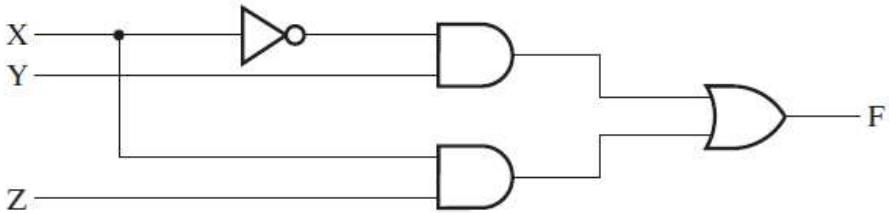
$F = \overline{X}YZ + \overline{X}Y\overline{Z} + XZ$

$= \overline{X}Y(Z + \overline{Z}) + XZ$

$= \overline{X}Y \cdot 1 + XZ$

$= \overline{X}Y + XZ$

간략화를 통한 성능향상
→ 게이트 수 감소
→ 전파지연 감소



4.2 논리회로의 HDL 표현

- **HDL(Hardware Description Language)**
  - 논리회로를 텍스트를 사용하여 표현
  - 대표적 HDL : VHDL, Verilog HDL

- **HDL을 사용한 논리회로 표현**
  - 구조적 기술(Structural Description) : 논리 회로 동작을 하드웨어 컴포넌트와 그 연결방법을 기술.
  - 동작 기술(Behavioral Description) : 논리 회로 동작을 동작중심으로 추상적으로 기술.
  - 데이터 흐름 기술(Dataflow Description) : 데이터 흐름 중심으로 기술

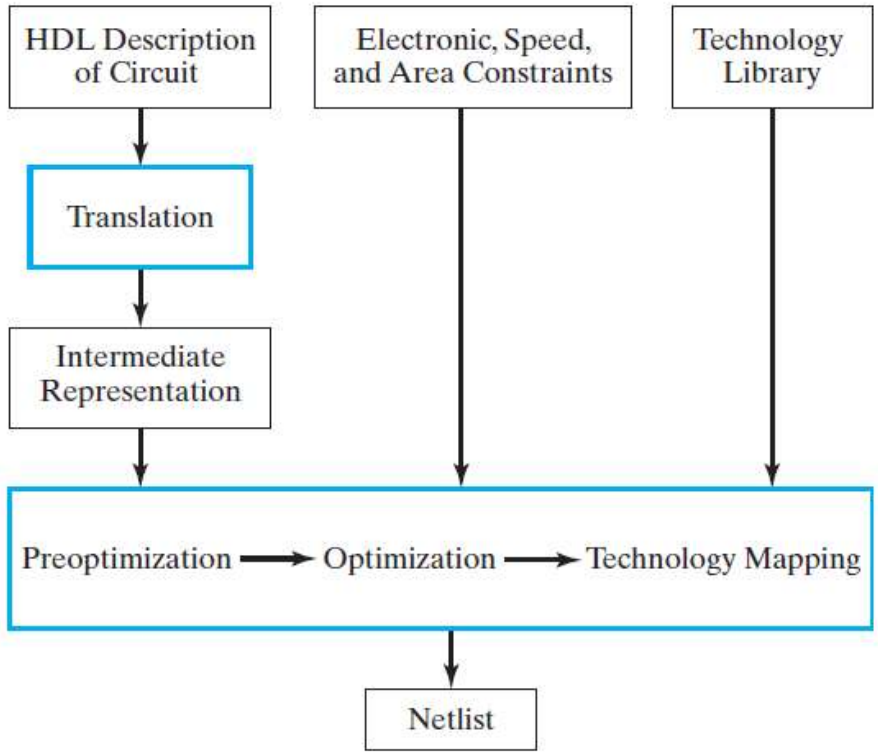
```
entity andgate is
        Port(  A : in   std_logic;
               B : in   std_logic;
               Y : out std_logic
               );
end andgate;

architecture Behavioral of andgate is
begin
        Y<= A and B ;
end Behavioral;
```

HDL Description

4.3 HDL-to-Logic 절차

- HDL에서 로직회로로 변환하는 절차 : 논리회로 합성(Synthesis)



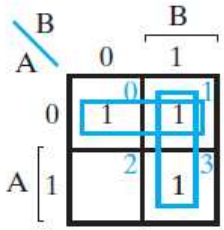
5. 논리회로 간략화

- **논리회로 간략화**
  - 로직 게이트 수 감소
  - 전파지연 감소
- **간략화 방법**
  - 부울대수를 사용한 간략화
  - Karnaugh Map을 사용한 간략화
    - 입력 변수가 4개 이하인 경우, 적용 용이
  - Quine-MaCluskey 알고리즘
    - 입력/출력변수의 개수에 제한
  - ECAD Tool을 사용한 간략화
    - ESPRESSO 알고리즘

5.1 Karnaugh Map

- **카르노 맵 (Karnaugh Maps)**
  - 진리표의 2차원적 표현방법.
  - 논리 표현식을 맵 형태로 표현한 후, 상호 연관관계를 패턴으로 분석하여 논리를 단순화해 나가는 방법

2-input K-map



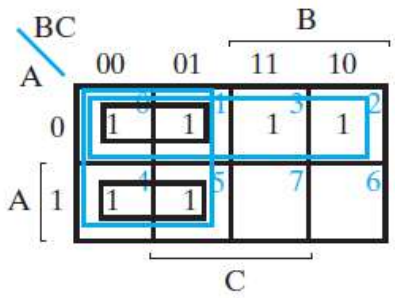
$F=\overline{A}\overline{B} + \overline{A}B + AB$

$F = \overline{A} + B$

5.2 Karnaugh Map을 사용한 최적화

- **3-input/4-input 카르노 맵 적용 사례**

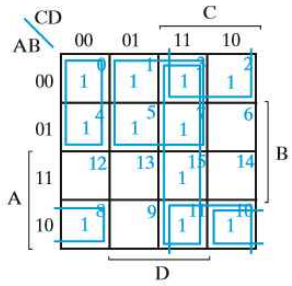
3-input K-map



$F=\overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$

$F = \overline{A} + B$

4-input K-map



$G(A,B,C,D) = \overline{A}\,\overline{C}\,\overline{D} + \overline{A}D + \overline{B}C + CD + A\overline{B}\,\overline{D}$

$G = \overline{B}\,\overline{D} + \overline{A}\,\overline{C} + CD$

기본 조합 회로

1. 조합 회로 개요

- **조합 회로(Combinational Circuit)**
  - 논리 게이트의 조합으로 이루어진 회로

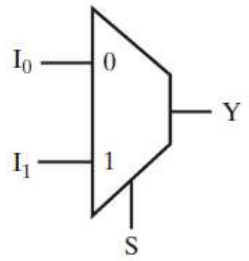
- **조합회로 특성**
  - 입력신호의 변화가 일정한 시간지연 후에 출력에 그대로 반영된다.
    - Transparent Circuit
  - 기억소자가 없다.

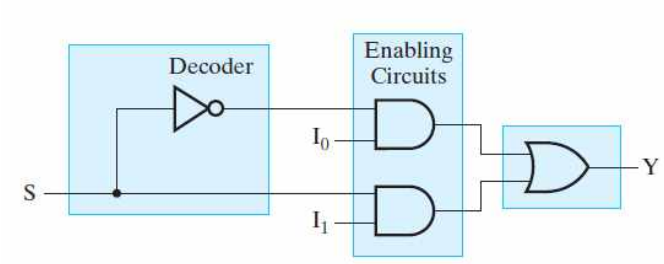
- **대표적 조합회로**
  - 멀티 플렉서(Multiplexer)
  - 인코더/디코더(Encoder/Decoder)
  - 반-가산기/전-가산기(Half/Full Adder)
  - 버퍼(Buffer)

2. 멀티플렉서

- **멀티 플렉서(Multiplexer)**
  - N 개의 입력신호 중 1개만을 선택해서 출력.

S	I_0	I_1	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1





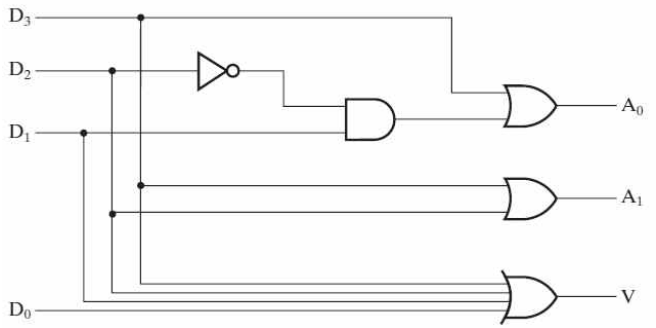
3. 인코더

- **인코더(Encoder)**
  - 입력 신호를 2진 값으로 변환하는 회로.
    - 2n 개의 입력 신호중 활성화된 1개의 신호의 위치를 n-bit로 표현
  - 4-to-2 인코더 : 4개의 입력을 2-비트로 출력
- **Priority Encoder**
  - n-개의 입력신호중 최상위 비트의 위치를 2진 값으로 표현

Validity Bit

Inputs				Outputs		
D_3	D_2	D_1	D_0	A_1	A_0	V
0	0	0	0	X	X	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1

4-Input Priority Encoder의 진리표

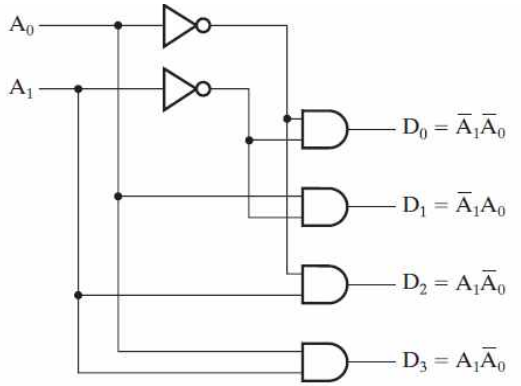


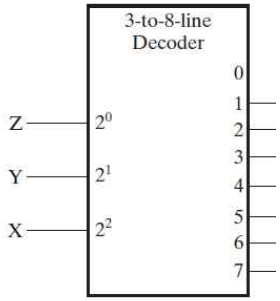
4. 디코더

- **디코더 (Decoder)**
  - 2진 값을 해석해서, 하나의 출력신호만을 활성화시키는 회로
  - 입력이 n-bit 라고 할 때, 출력은 2n 개가 된다.
    - 2-to-4, 3-to-8 디코더

A_1	A_0	D_0	D_1	D_2	D_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

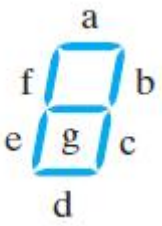
2-to-4 Decoder





4.1 7-Segment 디코더

- 이진값을 7개의 LED가 배치된 장치에 십진수로 표현할 수 있도록 이진값을 디코딩하는 회로



7-Segment

4.1 7-Segment 디코더

BCD Input				Seven-Segment Decoder						
A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
All other inputs				0	0	0	0	0	0	0

$$a = \overline{A}C + \overline{A}BD + \overline{B}\overline{C}\overline{D} + A\overline{B}\overline{C}$$

$$b = \overline{A}\overline{B} + \overline{A}\overline{C}\overline{D} + \overline{A}CD + A\overline{B}\overline{C}$$

$$c = \overline{A}B + \overline{A}D + \overline{B}\overline{C}\overline{D} + A\overline{B}\overline{C}$$

$$d = \overline{A}C\overline{D} + \overline{A}\overline{B}C + \overline{B}\overline{C}\overline{D} + A\overline{B}\overline{C} + \overline{A}B\overline{C}D$$

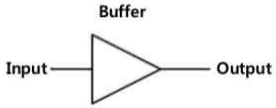
$$e = \overline{A}C\overline{D} + \overline{B}\overline{C}\overline{D}$$

$$f = \overline{A}B\overline{C} + \overline{A}\overline{C}\overline{D} + \overline{A}B\overline{D} + A\overline{B}\overline{C}$$

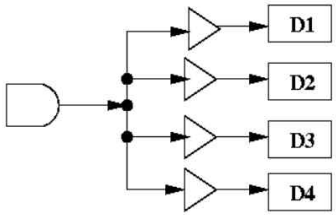
$$g = \overline{A}C\overline{D} + \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C}$$

5. 버퍼

- **버퍼**
  - 입출력 로직 값의 변화가 없음.
- **버퍼 용도**
  - 구동전류의 증폭
    - 게이트의 출력전류가 다른 게이트를 구동할 전류보다 부족할 경우 사용.
    - 논리회로의 Fan-out 보다 많은 논리회로를 구동하고자 할 경우 사용.
  - 전압레벨의 변경 : 3V에서 5V, 5V에서 3V로 변환.



Input	Output
0	0
1	1

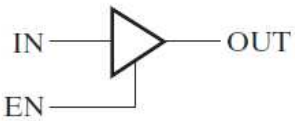


5.1 3-State 버퍼

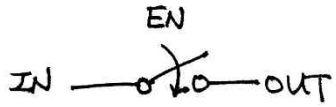
- **3-state 버퍼**
  - 0, 1, High-impedance 등 3가지 상태를 허용하는 버퍼.
  - 버스와 같이 신호선을 공유할 때 신호의 충돌을 피하는 수단으로 사용.
- **3-state 버퍼 상태**
  - 연결 상태(Close) : 로직 값 '0' 또는 '1'을 전달할 수 있는 상태
  - 단선 상태(Open) : High-Impedance (High-Z) 상태
- **High-impedance**
  - Z 라고 표시하며, 전기적으로 절연된 상태를 의미.
  - 신호선이 연결되지 않아 신호가 없는 Floating 상태.

EN	IN	OUT
0	X	Hi-Z
1	0	0
1	1	1

Truth Table

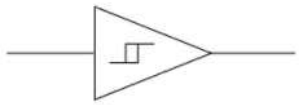


Logic Symbol

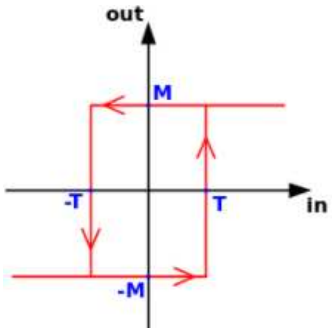


5.2 Schmitt-Trigger 버퍼

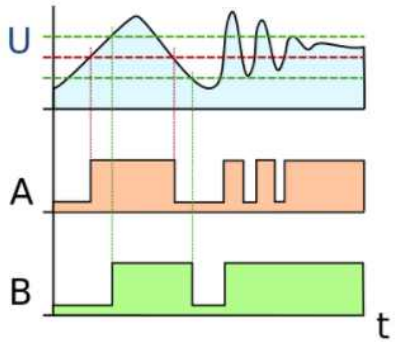
- **Schmitt-trigger 버퍼**
  - 히스테리시스(Hysteresis) 함수를 사용한 버퍼
    - 로직 레벨의 문턱값(Threshold)이 2개인 버퍼
  - 주로 신호에서 잡음을 제거할 목적으로 사용
    - 스위치의 debouncing 회로에 사용.



Schmitt Trigger Buffer 심볼



히스테리시스 함수



Schmitt Trigger Buffer 적용 결과 (B)

가산기

1. 가산기 개요

- **가산기(Adder)**
  - 2 개의 피연산자(Operand)를 더하는 회로

- **가산기 유형**
  - 반-가산기 (Half-Adder) : 2개의 1-비트 피연산자를 더하는 회로
  - 전-가산기 (Full-Adder) : 2개의 1-비트 피연산자와 캐리를 더하는 회로
  - N-비트 가산기 : 2개의 N-비트 피연산자를 더하는 회로

1.1 반-가산기

- **2 개의 1-비트 피연산자만을 더하는 조합회로**
  - 입력 신호 : 2개의 1-비트 피연산자 (X, Y)
  - 출력 신호 : 2개의 1-비트 결과
    - S(sum) : 합
    - C(carry) : 올림

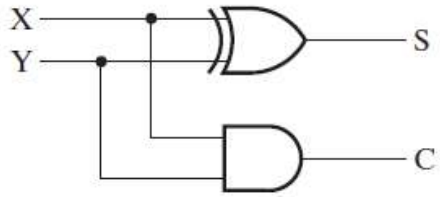
Inputs		Outputs	
X	Y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Truth Table

$$S = \overline{X}Y + X\overline{Y} = X \oplus Y$$

$$C = XY$$

Boolean Equation



Logic Diagram(Schematic)

1.2 전-가산기

- **2 개의 1-비트 피연산자와 Carry를 더하는 조합회로**
  - 입력 신호 : 2개의 1-비트 피연산자(X, Y) + 1-비트 Carry(Z)
  - 출력 신호 : 2개의 1-비트 결과
    - S(sum) : 합
    - C(carry) : 올림

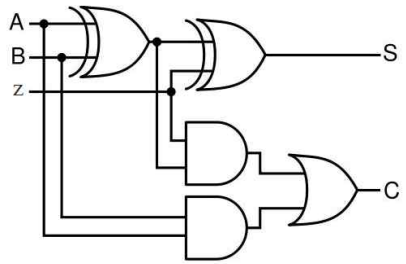
Inputs			Outputs	
X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Truth Table

$$S = \overline{X}\overline{Y}Z + \overline{X}Y\overline{Z} + X\overline{Y}\overline{Z} + XYZ$$

$$C = XY + XZ + YZ$$

Boolean Equation



Logic Diagram(Schematic)

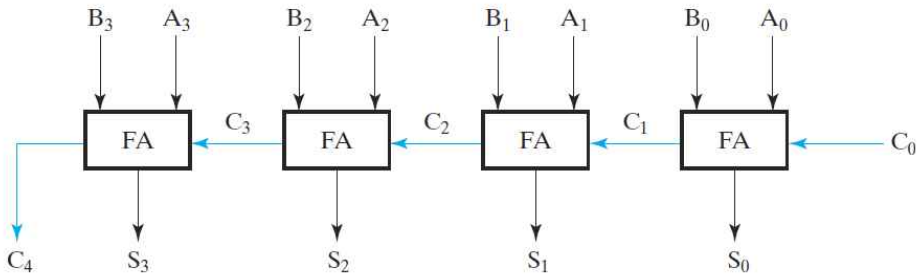
2. n-bit 가산기

- **2개의 n-비트 데이터를 더하는 회로**
  - N개의 전-가산기를 연결하여 구성

- **Carry 전파 지연을 단축시키기 위한 가산기 구조개선**
  - Ripple Carry Adder (RCA)
  - Carry Look-ahead Adder(CLA)
  - Carry Select Adder (CSL) :
  - Carry Save Adder (CSA) :
  - Carry Skip Adder (CSK) :

2.1 Ripple Carry Adder (RCA)

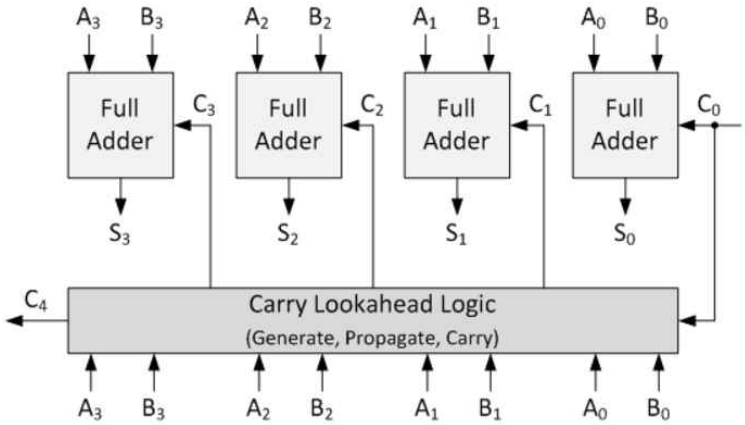
- **2개의 n-비트 데이터를 더하는 회로**
- **n-bit Ripple Carry Adder (RCA)**
  - N개의 전-가산기를 연결하여 구성
  - 가장 단순한 n-bit 가산기
  - Carry 전파지연 문제발생

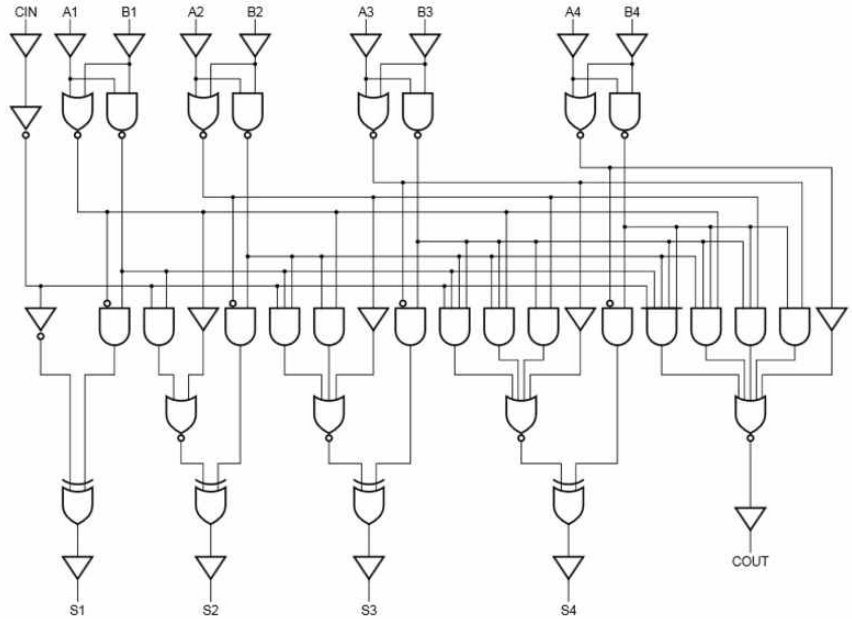


4-bit Ripple Carry Adder (S = A + B)

2.2 Carry Look-ahead Adder (CLA)

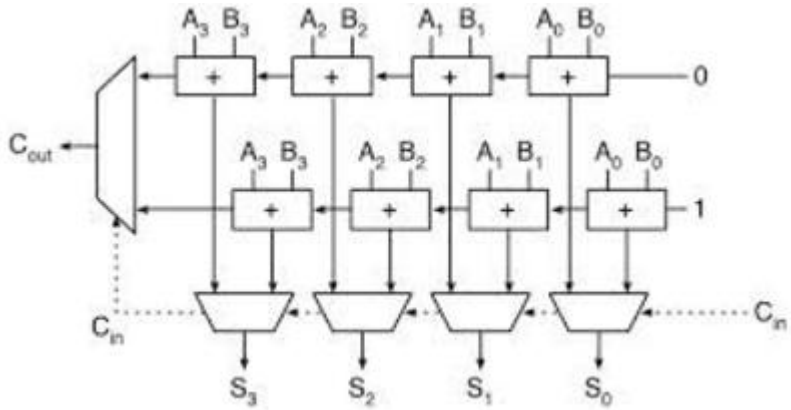
- 별도 회로를 사용하여 Carry 계산.





2.3 Carry Select Adder (CSL)

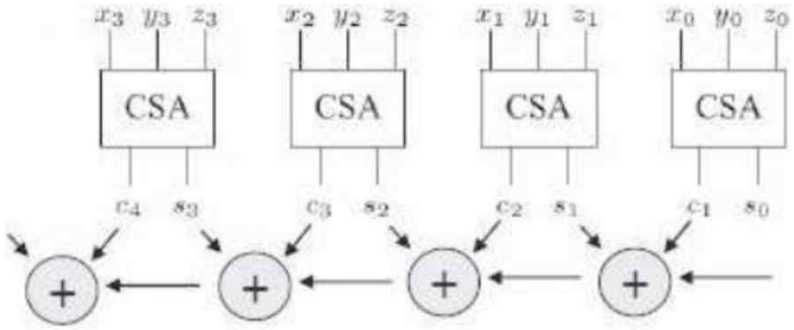
- **Carry가 '0' 또는 '1' 일 경우를 모두 고려해서 덧셈을 실시한 후에 Carry 값에 따라 최종 선택.**
  - 가산기 하드웨어 2배와 멀티플렉서 비용 필요



2.4 Carry Save Adder (CSA)

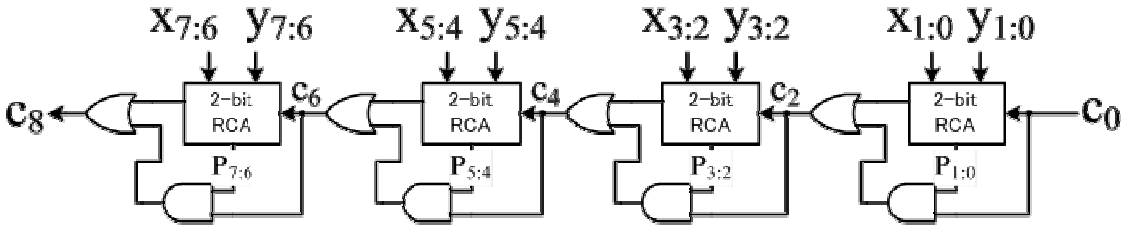
- **Carry를 상위 자릿수에 전파하지 않고, 자체 저장.**
  - 단일 덧셈보다, 누적 덧셈(Accumulative Addition)에 유리
    - Multiple-operands adder

```
  101110101
+ 110111101
= 211221202
```



2.5 Carry Skip Adder (CSK)

- 덧셈을 몇 개의 블록으로 나눈 후, 블록 단위로 carry 를 처리
  - Carry Bypass Adder

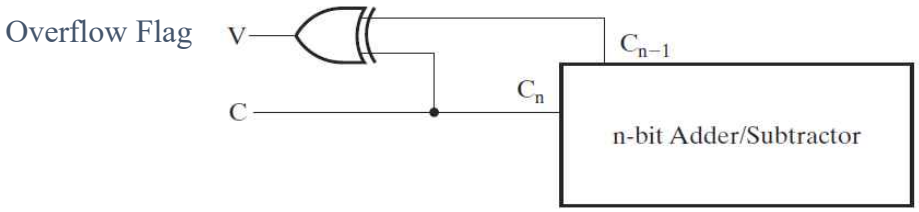


$$P_i\,(propagation) \quad P_i = A_i \oplus B_i$$

$$G_i\,(generation) \quad G_i = A_i \bullet B_i$$

2.6 오버플로우 검사

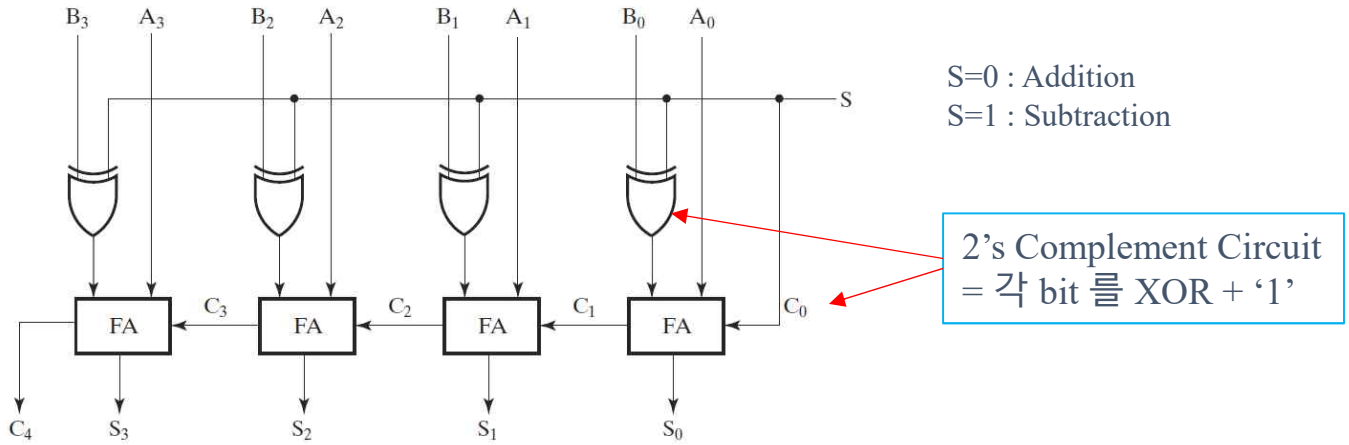
- **오버 플로우(Overflow)**
  - 덧셈 결과 값이 결과 값 저장 범위를 초과하는 경우
    - 예) 덧셈 결과 값을 4-bit로 표현하는 경우, 결과 값이 15를 초과하는 경우, 오버 플로우가 발생.
- **오버 플로우는 결과 값을 잘못 저장한 에러상태**
  - 오버 플로우 발생을 감지하여 플래그(Flag)로 표시
- **오버 플로우 감지방법**
  - 덧셈 결과 MSB에서 발생하는 Carry와 이전 자릿수에서 발생하는 Carry를 XOR



3. 덧셈-뺄셈기

- **2의 보수(2's Complement)를 사용한 덧셈-뺄셈기**
  - 뺄셈기 구현 : 음수를 2의 보수로 표현 후 더함.

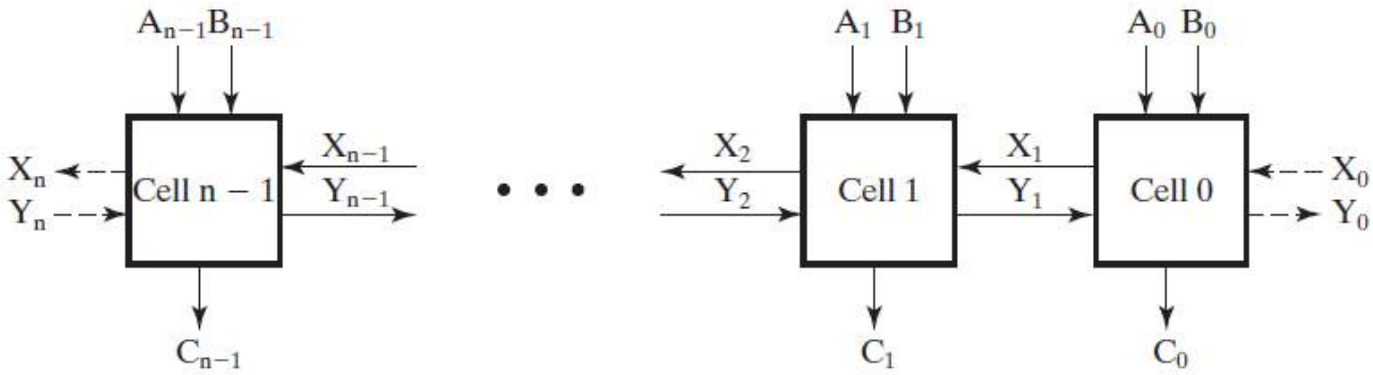
A – B = A +(B의 2's Complement)



조합회로 설계

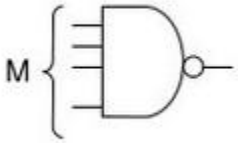
1. 조합회로의 연결

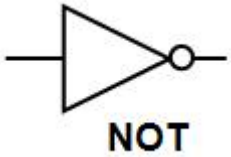
- **단위 조합회로를 연속적으로 연결하여 특정 작업을 수행할 수 있는 회로를 생성**
  - Cell 0 의 출력은 다음 조합회로 Cell 1의 입력으로 사용.



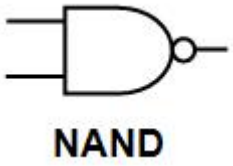
1.1 Fan-In

- **로직 게이트의 Fan-In : 로직 게이트의 입력 수**
  - Fan-in 의 수가 증가하면 전파 지연이 증가한다.

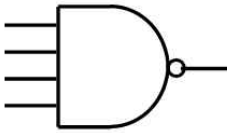




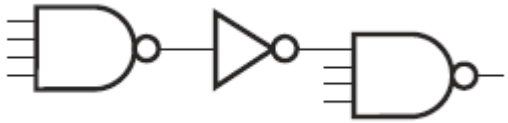
Fan-In = 1



Fan-In = 2



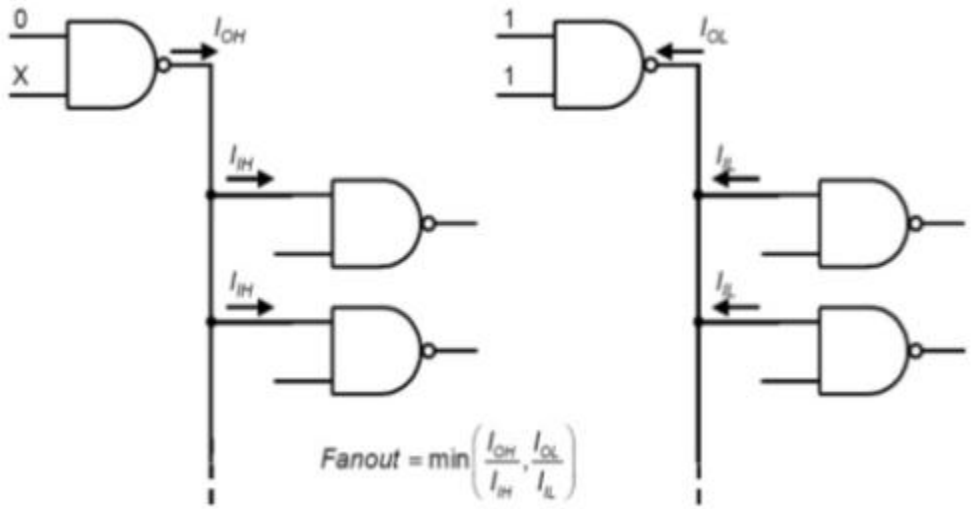
Fan-In = 4



Fan-In 이 4인 2개의 NAND 게이트로 7-inpt NAND 구현

1.2 Fan-Out

- **로직 게이트의 Fan-Out**
  - 로직 게이트의 출력이 구동할 수 있는 로직 게이트의 입력 수
  - Load-drving capability

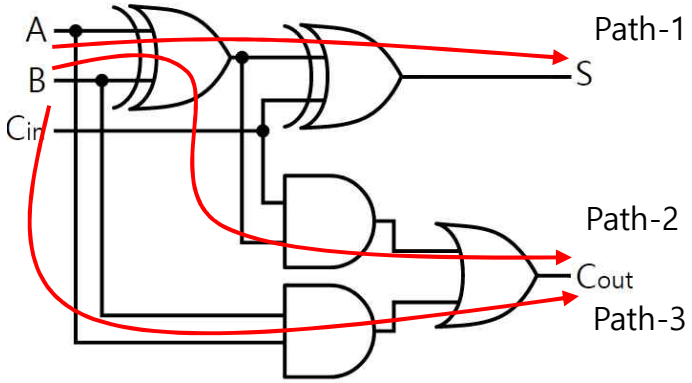


2. 조합 회로 설계 절차

- **조합회로 설계 순서**
  - 회의 동작 정의 : 사양서 작성
  - 회로의 동작을 부울 방정식이나 진리표로 표현
  - 논리 회로 최적화 기술을 적용하여 회로 간략화
  - 기능 검증
    - ECAD 도구를 사용한 논리 시뮬레이션
    - 논리회로의 타이밍 정보를 사용한 타이밍 분석
  - 사용가능한 구현기술로 매핑하여 구현
    - 표준 상용 논리 IC 를 사용한 구현
    - FPGA(Field Programmable Gate Array)
    - 주문형 반도체로 구현

2. 조합 회로 설계 절차

- **조합회로 동작 속도 계산**
  - 회로의 시간지연을 고려하여 회로의 Critical Path를 탐색
    - Critical Path : 전파지연의 합이 최대인 경로
  - 회로의 성능을 예측가능
    - 예상 동작 주파수 = 1/(최대 전파지연의 합)



XOR의 전파지연 = 20ns
AND의 전파연 = 10ns
OR의 전파지연 = 12ns 라 가정하면,

Critical Path = Path-2
Max. 전파지연의 합 = 20 + 10 + 12ns = 42ns
예상 동작 주파수 = 1/42 GHz = 0.024 GHz = 24Mhz

end