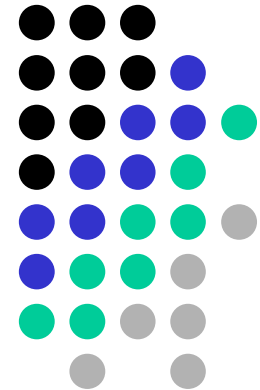


# Chapter 2. 데이터처리 (저장 및 처리)

- 컴퓨터개론 -

금오공과대학교  
컴퓨터소프트웨어공학과



# Contents

---

## 1. 비트 저장

1.1 비트, 연산, 게이트

## 2. 2진 체계

2.1 비트 패턴을 이용한 정보 표현

2.2 2진법

## 3. 수(양수, 음수, 정수)의 표현

3.1 양수의 저장

3.2 정수의 저장

## 4. 텍스트 표현

4.1 ASCII

4.2 한글 코드

## 5. 비트의 처리/활용

5.1 데이터 압축

5.2 통신 오류

## 1.1 비트와 비트 패턴

1. **비트(bit)**: 2진 숫자 (binary digit) - 0 또는 1

2. 비트 패턴은 정보 표현에 사용된다.

- A. 숫자
- B. 텍스트 문자
- C. 이미지
- D. 사운드
- E. 기타 등등

**바이트(Byte) =  
8개의 비트(Bit) 모임**

**Nibble은?**

예) 비트는 언제 사용?

32비트 컴퓨터, 64비트 컴퓨터,

32비트 운영체제,

인터넷 속도 100메가: 100M**bps**

**bit per second**

**RPM?**

## 1.2 부울(Boolean) 연산

### 1. 부울 연산:

한 개 이상의 참(True) / 거짓(False) 값을 다루는 연산

### 2. 부울 연산자

A. AND: &

B. OR: |

C. XOR (eXclusive OR): ^

D. NOT: ~ (Negation)

AND와 OR의  
논리연산 - &&, ||  
비트연산 - &, |

## 1.2 부울 연산 AND, OR, XOR, NOT

AND 연산

$$\begin{array}{r} 0 \\ \text{AND } 0 \\ \hline 0 \end{array}$$

$$\begin{array}{r} 0 \\ \text{AND } 1 \\ \hline 0 \end{array}$$

$$\begin{array}{r} 1 \\ \text{AND } 0 \\ \hline 0 \end{array}$$

$$\begin{array}{r} 1 \\ \text{AND } 1 \\ \hline 1 \end{array}$$

OR 연산

$$\begin{array}{r} 0 \\ \text{OR } 0 \\ \hline 0 \end{array}$$

$$\begin{array}{r} 0 \\ \text{OR } 1 \\ \hline 1 \end{array}$$

$$\begin{array}{r} 1 \\ \text{OR } 0 \\ \hline 1 \end{array}$$

$$\begin{array}{r} 1 \\ \text{OR } 1 \\ \hline 1 \end{array}$$

XOR 연산

$$\begin{array}{r} 0 \\ \text{XOR } 0 \\ \hline 0 \end{array}$$

$$\begin{array}{r} 0 \\ \text{XOR } 1 \\ \hline 1 \end{array}$$

$$\begin{array}{r} 1 \\ \text{XOR } 0 \\ \hline 1 \end{array}$$

$$\begin{array}{r} 1 \\ \text{XOR } 1 \\ \hline 0 \end{array}$$

NOT 연산

$$\begin{array}{r} \text{NOT } 0 \\ \hline 1 \end{array}$$

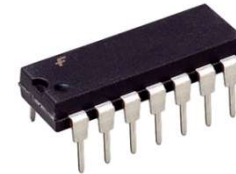
$$\begin{array}{r} \text{NOT } 1 \\ \hline 0 \end{array}$$

## 1.3 게이트(gate)

### 1. 게이트: 부울 연산을 계산하는 장치

A. 흔히 (소형) 전자 회로로 구현된다

B. 컴퓨터 제작에 사용되는 구성 요소



예)

AND Gate

OR Gate

XOR Gate

Not Gate

NAND Gate

NAND 게이트



표준 논리 기호

A	B	C
0	0	1
0	1	1
1	0	1
1	1	0


진리표

$$C = \sim(A.B) = \overline{(A.B)}$$




# 1.4 AND, OR, XOR, NOT 게이트 기호와 입출력 값

AND Gate

입력  출력


입력1	입력2	출력
0	0	0
0	1	0
1	0	0
1	1	1

OR Gate

입력  출력

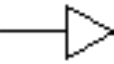
입력1	입력2	출력
0	0	0
0	1	1
1	0	1
1	1	1

XOR Gate

입력  출력

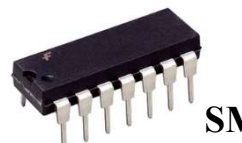
입력1	입력2	출력
0	0	0
0	1	1
1	0	1
1	1	0

NOT Gate

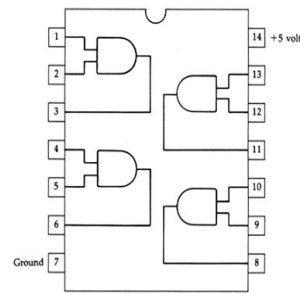
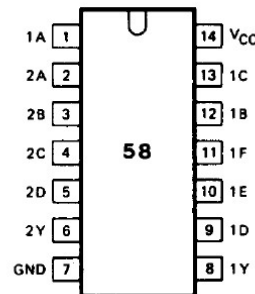
입력  출력

입력	출력
0	1
1	0

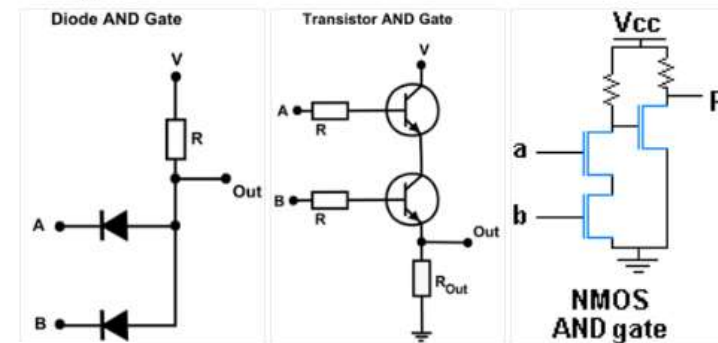
DIP



SMD



## Implementations

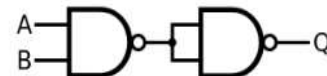


## Alternatives

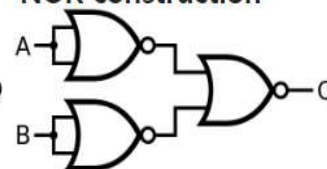
Desired gate



NAND construction



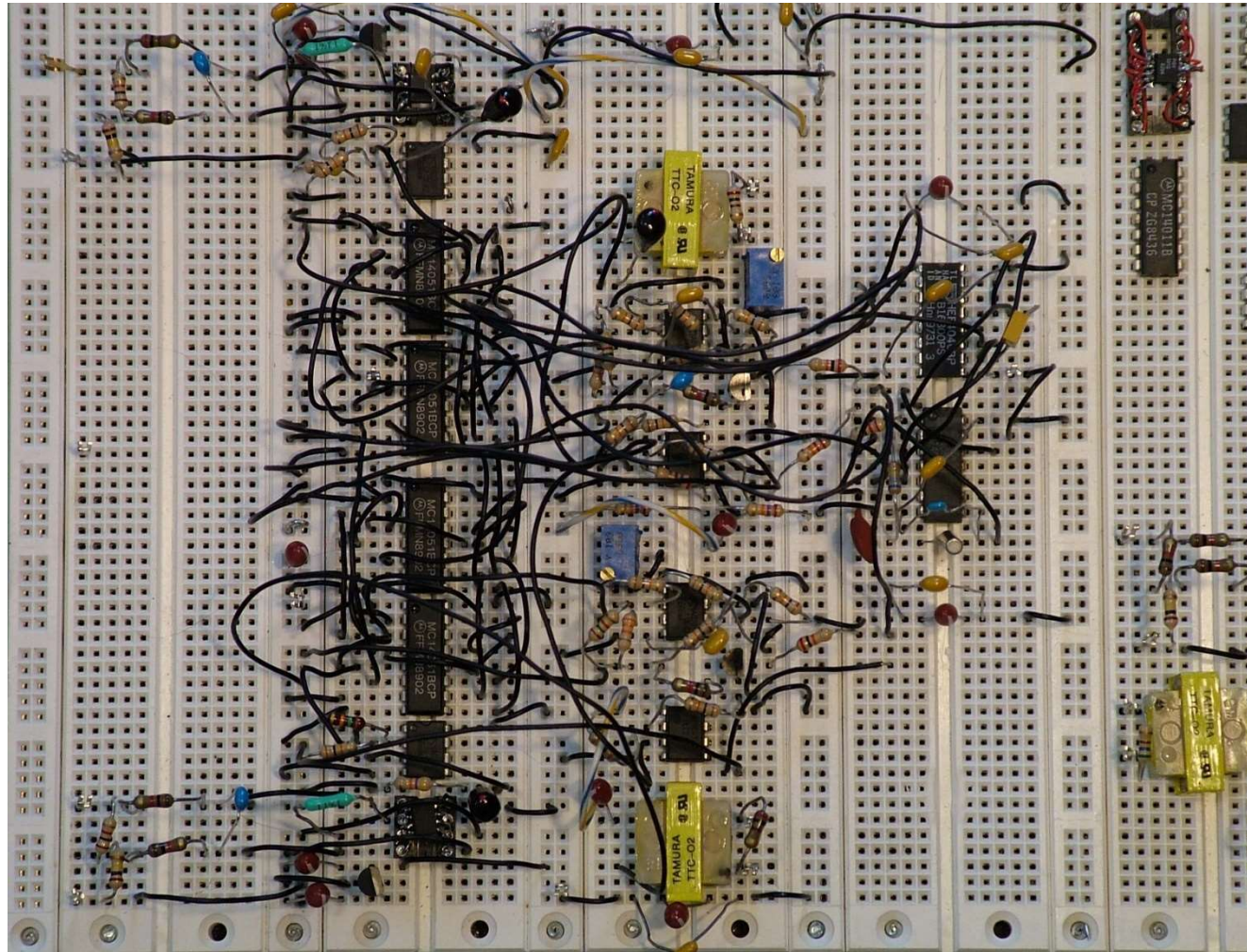
NOR construction





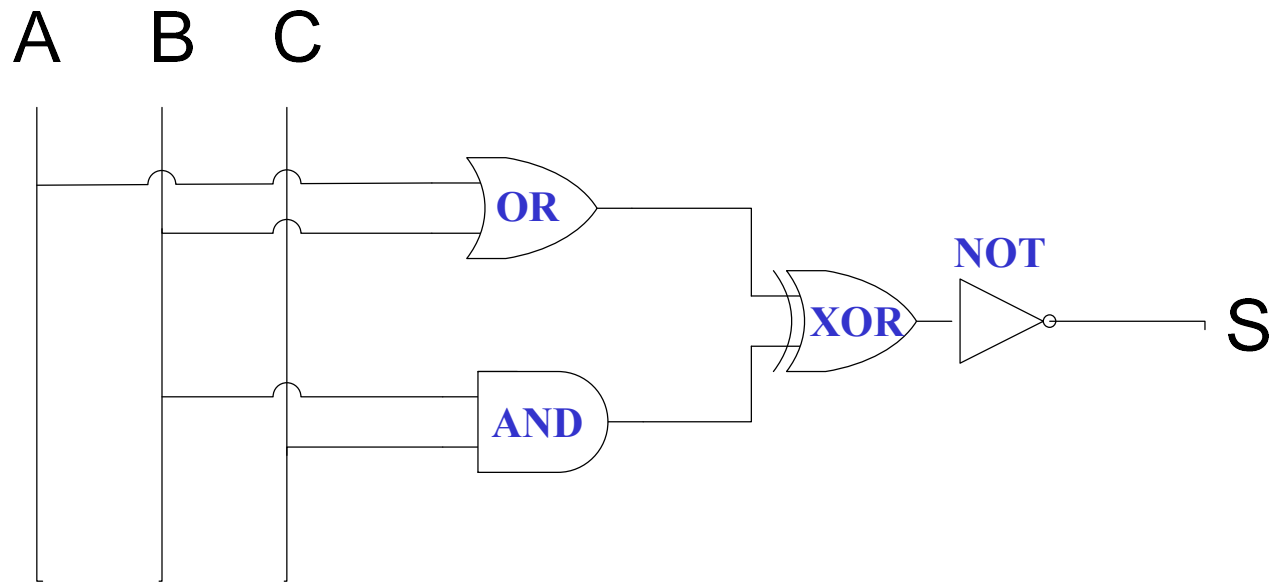
# 1.5 AND, OR, XOR, NOT 게이트 사용

## 1. Bread Board : 논리회로





## 1.6 AND, OR, XOR, NOT 게이트 회로 예



XOR – 다르면 1

A	B	A OR B	C	A AND B AND C	XOR	S
0	0	0	1	0	0	1
0	1	1	1	0	1	0
1	0	1	0	0	1	0
1	1	1	1	1	0	1

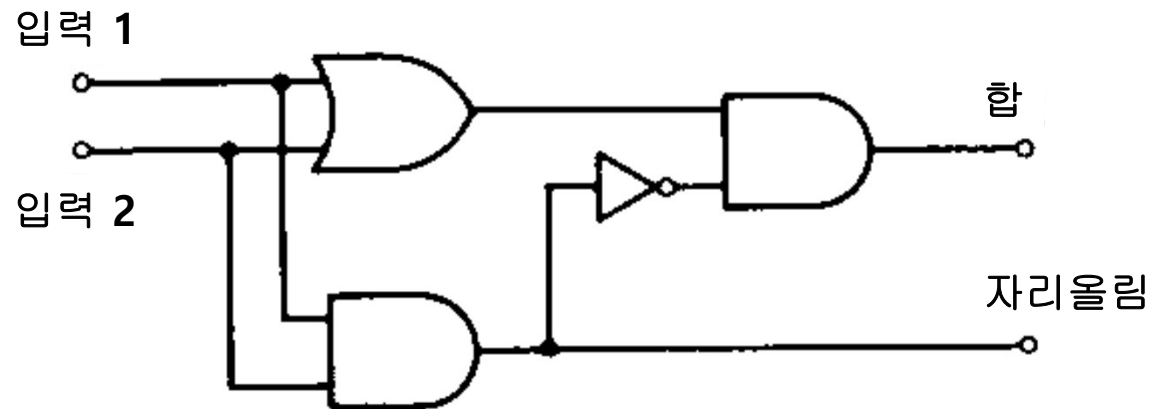
## 1.7 AND, OR, XOR, NOT 게이트 사용

---

1. CPU : 산술 논리 연산이 주요한 목적  
덧셈 회로, 뺄셈 회로(ALU: Arithmetic Logic Unit) 등을  
게이트를 사용하여 설계 가능
2. 메모리: 값의 저장이 주요한 목적  
1비트 저장 가능 플립플롭을 게이트를 사용하여 설계 가능

## 1.8 게이트를 이용한 2진 덧셈 회로

0	1	0	1
+ 0	+ 0	+ 1	+ 1
0	1	1	10



## 1.9 플립플롭(Flip-Flop)

### 1. 플립플롭: 게이트로 만든 회로, 1 비트를 저장할 수 있음

A. 입력신호에 의해서 상태를 바꾸도록 지시할 때까지 현재의 2진상태 유지

B. 저장 값을 1로 설정하는 입력 라인을 가짐

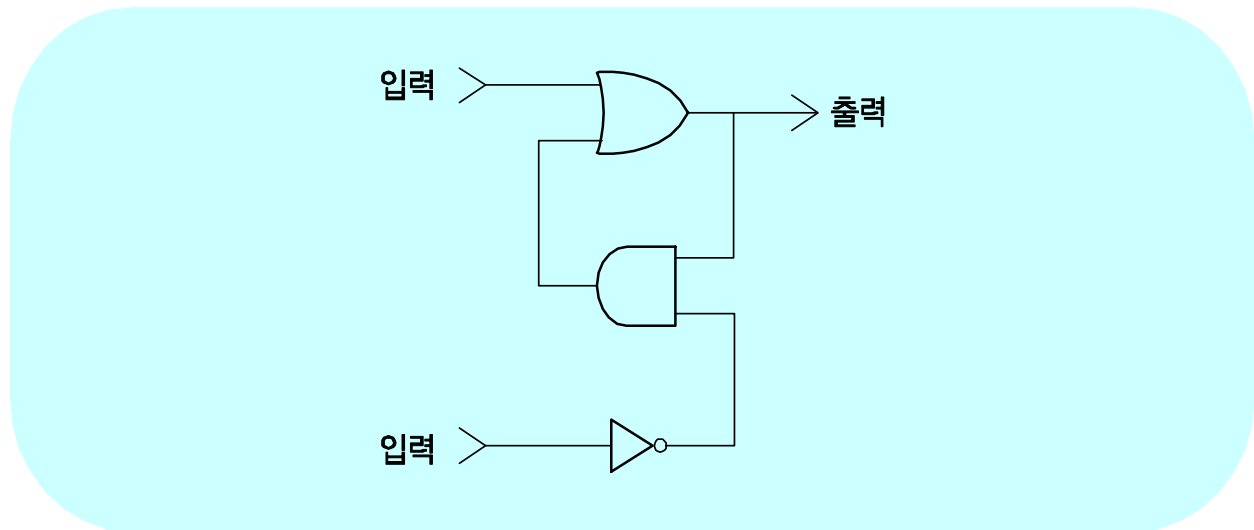
Flip : 톡치다, 뒤집다

C. 저장 값을 0로 설정하는 입력 라인을 가짐

Flop : 톡 던지다, 탁 때리다

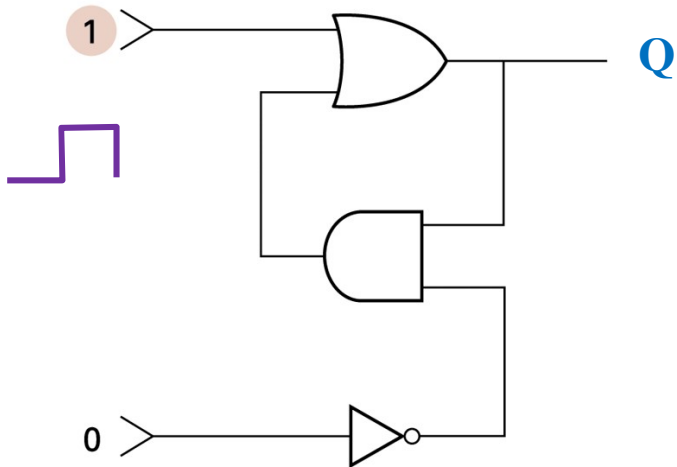
D. 두 입력 라인 모두 0일 때, 가장 최근의 저장 값을 유지함  
(플립플롭 디자인에 따라서 다름)

### 2. 플립플롭 예:



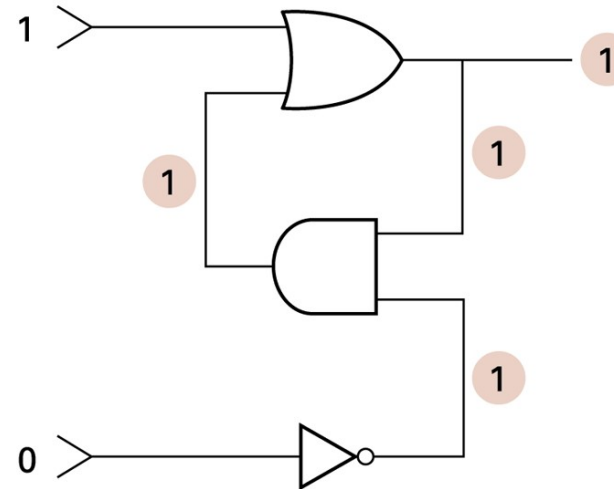
## 1.10 플립플롭의 출력을 1로 설정하기

a. 1 is placed on the upper input.



위쪽 입력에 1을 인가

b. This causes the output of the OR gate to be 1 and, in turn, the output of the AND gate to be 1.

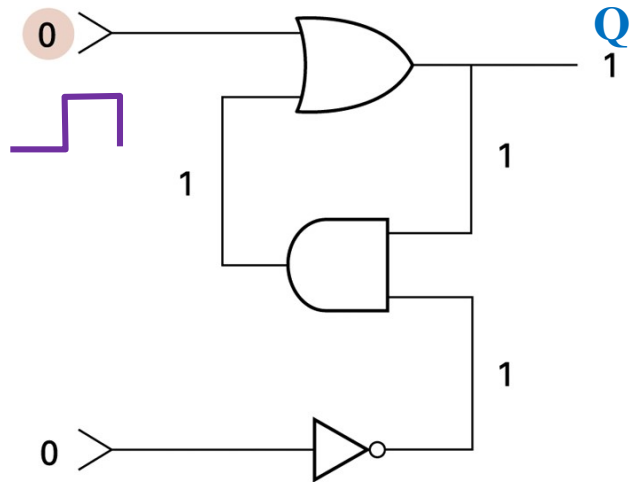


이 결과로 OR 게이트의 출력(Q)이 1이 되며,  
이는 다시 AND 게이트의 출력을 1로 만든다.



## 1.10 플립플롭의 출력을 1로 설정하기 (계속)

- c. The 1 from the AND gate keeps the OR gate from changing after the upper input returns to 0.

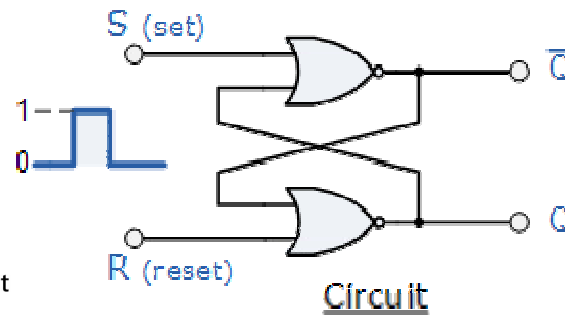
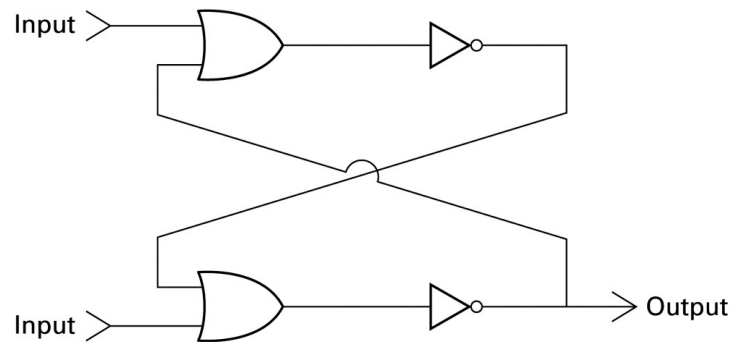


AND 게이트의 출력 1은  
위쪽 입력이 0으로 돌아간 후에도  
OR 게이트의 출력이 변하지 않게 만든다 - **Set**.

# 1.11 플립플롭을 구성하는 또 다른 방법

## 1. 플립플롭의 예

RS(Reset-Set) FlipFlop, D(Data, Delay) FlipFlop,  
JK(Jack과 Kilby가 발명) FlipFlop 등



S	R	Q	$\bar{Q}$
0	0	No change	
0	1	0	1
1	0	1	0
1	1	0	0
		(Invalid)	

## 2. 메모리, 레지스터 등을 구성하는데 활용

## 3. 자세한 이론, 활용 및 실험은 논리회로 수업에서 학습

