



DeepL

Subscribe to DeepL Pro to translate larger documents.  
Visit [www.DeepL.com/pro](https://www.DeepL.com/pro) for more information.

# 디지털 디자인 및 컴퓨터 아키텍처

사라 해리스 & 데이비드 해리스

## 1장: 0에서 1로

**수정 됨 유영환, 2023**

# 1장 :: 주제

- **복잡성 관리의 기술**
- **번호 시스템**
  - 이진수
  - 16진수
  - 비트, 바이트, 니블
  - 추가
  - 서명된 번호
  - 확장
- **논리 게이트**
- **논리 레벨**
- **CMOS 트랜지스터**
- **트랜지스터 레벨 게이트 설계**
- **전력 소비량**

1장: 0에서 1로

# 복잡성 관리의 기

술

# 복잡성 관리의 기술

- 한 사람의 머릿속에 한 번에 담기에는 너무 큰 것을 어떻게 디자인할 수 있을까요?
- 추상화
- 규율
- 세 가지
  - 계층 구조

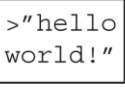


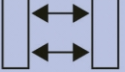
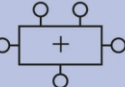
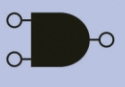
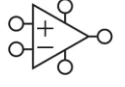


- 모듈화
- 규칙성

# 추상화

## 중요하지 않은 세부 정보 숨기기

- **디지털 회로:** 아날로그 전압을 0 또는 1로 변환하는 논리 게이트
- **로직 설계:** 복잡한 구조(예: 가산기 및 메모리)
- **마이크로 아키텍처:** *아키텍처에* 정의된 명령을 실행하기 위해 논리 요

focus of this course

|                      |   |                           |
|----------------------|---|---------------------------|
| Application Software |    | programs                  |
| Operating Systems    |    | device drivers            |
| Architecture         |    | instructions<br>registers |
| Micro-architecture   |    | datapaths<br>controllers  |
| Logic                |    | adders<br>memories        |
| Digital Circuits     |    | AND gates<br>NOT gates    |
| Analog Circuits      |    | amplifiers<br>filters     |
| Devices              |  | transistors<br>diodes     |
| Physics              |  | electrons                 |

소를 결합하는 것

- **아키텍처**: 프로그래머가 사용하는  
명령어 및 레지스터 세트



- 의도적으로 디자인 선택권 제한
- 예시: 디지털 규율
  - 연속 전압이 아닌 이산 전압
  - 아날로그 회로보다 설계가 간단하여 보다 정교한 시스템 구축 가능
  - 디지털 카메라, 디지털 텔레비전, 휴대폰, CD

# 등 아날로그 시스템을 대체하는 디지털 시스템

# 세 가지

- 계층 구조

- 모듈과 서브모듈로 구분된 시스템

- 모듈화

- 잘 정의된 기능 및 인터페이스 보유

- 규칙성

- 균일성을 장려하여 모듈을 쉽게 재사용할 수 있음

니다.

# 디지털 규율: 이진 값

- **두 개의 불연속 값입니다:**
  - 1과 0
  - 1, 참, 높음
  - 0, 거짓, 낮음
- **1과 0:** 전압 레벨, 회전 기어, 유체 레벨 등
- 디지털 회로는 전압 레벨을 사용합니다.
  - 0: 저전압(GND)

- 1: 고전압( $V_{DD}$ )
- **비트**: 2진수 숫자

1장: 0에서 1로

# 숫자 체계: 이진수

# 번호 시스템

- 십진수

1000년대 칼럼  
100의 열  
10의 열

$$537410 = 5 \times 10^3 + 3 \times 10^2 + 7 \times 10^1 + 4 \times 10^0$$

오천      3백      일곱      네  
십      가지

디지털 시스템에서 십진수는 소수점이 있는 숫자뿐만 아니라 10을 기본으로 하는 모든 숫자를 의미합니다.

- 이진수

8의 열  
4의 열  
2의 열

$$11012 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 1310$$

하나      여      둘      하나 넷



두 번째

하  
나

# 이진수로 계산

## 바이너리

0

1

10

11

100

101

...

## 십진수

0

1

2

3

4

5

# 두 가지의 힘

- $2^0 = 1$
- $2^1 = 2$
- $2^2 = 4$
- $2^3 = 8$
- $2^4 = 16$
- $2^5 = 32$
- $2^6 = 64$
- $2^7 = 128$
- $2^8 = 256$
- $2^9 = 512$
- $2^{10} = 1024$
- $2^{11} = 2048$
- $2^{12} = 4096$
- $2^{13} = 8192$
- $2^{14} = 16384$
- $2^{15} = 32768$

편리한 암  
기

# 숫자 변환

- 바이너리를 십진수로 변환합니다:

- $10011_2$  을 소수로 변환

- $16 \times 1 + 8 \times 0 + 4 \times 0 + 2 \times 1 + 1 \times 1 = 19_{10}$

- 십진수를 바이너리로 변환합니다:

- 4710을 바이너리로 변환

- $32 \times 1 + 16 \times 0 + 8 \times 1 + 4 \times 1 + 2 \times 1 + 1 \times 1 = 101111_2$

# 10진수를 2진수로 변환

- 두 가지 방법이 있습니다:
  - **방법 1:** 맞는 2의 최대 거듭제곱 찾기, 빼기, 반복하기
  - **방법 2:** 2로 반복해서 나누고 나머지는 다음으로 큰 비트에 넣습니다.

# 10진수를 2진수로 변환

53<sub>10</sub>

**방법 1:** 53<sub>10</sub>에 맞는 2의 최대 거듭제곱을 찾아서 빼고 반복합니다.

|            |      |                       |
|------------|------|-----------------------|
|            | 32×1 |                       |
| 53-32 = 21 | 16×1 |                       |
| 21-16 = 5  | 4×1  |                       |
| 5-4 = 1    | 1×1  | = 110101 <sub>2</sub> |

**방법 2:** 반복적으로 2로 나누고, 나머지는 다음으로 큰 비트 53<sub>10</sub> = 53/2 = 26 R1  
에 넣습니다.

|           |    |                       |
|-----------|----|-----------------------|
| 26/2 = 13 | R0 |                       |
| 13/2 = 6  | R1 |                       |
| 6/2 = 3   | R0 |                       |
| 3/2 = 1   | R1 |                       |
| 1/2 = 0   | R1 | = 110101 <sub>2</sub> |

# 이진 값 및 범위

- **N자리 소수점 숫자**
  - 몇 개의 값이 있나요
  - 범위?
  - 예시: 3자리 십진수:

-  
-

- **N비트 이진수**
  - 몇 개의 값이 있나도 \_ \_
  - 범위:
  - 예시: 3자리 이진수:

-  
-

1장: 0에서 1로

**숫자 체계: 16진수**



# 16진수

- 베이스 16
- 바이너리  
의 약어

| 16진수 | 십진수 등가 | 이진 등가물 |
|------|--------|--------|
| 0    | 0      | 0000   |
| 1    | 1      | 0001   |
| 2    | 2      | 0010   |
| 3    | 3      | 0011   |
| 4    | 4      | 0100   |
| 5    | 5      | 0101   |
| 6    | 6      | 0110   |
| 7    | 7      | 0111   |
| 8    | 8      | 1000   |
| 9    | 9      | 1001   |
| A    | 10     | 1010   |
| B    | 11     | 1011   |
| C    | 12     | 1100   |
| D    | 13     | 1101   |
| E    | 14     | 1110   |
| F    | 15     | 1111   |

# 16진수에서 이진수로 변환

- 16진수를 2진수로 변환합니다:
  - $4AF_{16}$  (0x4AF라고도 함)을 바이너리로 변환합니다.
- 16진수를 10진수로 변환합니다:
  - $4AF_{16}$ 을 십진수로 변환

# 16진수 및 이진 접두사

- 텍스트 파일에서 구독을 작성하기 어려움
- 일부 프로그래밍 언어에서는 접두사를 사용합니다.
  - 16진수: 0x
    - $0x23AB = 23AB_{16}$
  - 바이너리: 0b

- $0b1101 = 1101_2$

1장: 0에서 1로

**숫자 시스템: 바이  
트, 니블 및 올 댓**

재즈

# 비트, 바이트, 니블...

- 바이트: 8비트
  - 다음 중 하나를 나타냅니다. \_\_\_\_  
값
  - [\_\_, \_\_]
- 니블: 4비트
  - 다음 중 하나를 나타냅니다. \_\_\_\_  
값
  - [\_\_, \_\_]

이진수 한 자리는 \_\_\_\_ bit  
16진수 중 하나는 \_\_\_\_ 비트 또는 \_\_\_\_ 니블  
두 개의 16진수는 \_\_\_\_ 바이트

10010110

가장 중요      최하위 비  
한 비트      트

바이트

10010110

니블

CEBF9AD7

왼쪽이 가장 중요함 오  
른쪽이 가장 중요하지  
않음

가장 중요  
한 바이트

최하위 바  
이트  
└─┘



## 2의 큰 거듭제곱

- $2^{10} = 1 \text{ 킬로}$   $\approx 10^3 (1024)$
- $2^{20} = 1 \text{ 메가}$   $\approx 10^6 (1,048,576)$
- $2^{30}$   $\approx 10^9 (1,073,741,824)$
- $2^{40} \text{ 1기가} = 1 \text{ 기}$
- $2^{50} \text{ 가}$
- $2^{60} = 1 \text{ 테라}$   $\approx 10^{12}$
- $\quad \quad \quad = 1 \text{ 페타}$   $\approx 10^{15}$
- $\quad \quad \quad = 1 \text{ 엑사}$   $\approx 10^{18}$

## 2의 거듭제곱 추정

- $2^{24}$  의 값은 무엇입니까?

$$2^4 \times 2^{20} \approx 1 \text{ 600만}$$

- 32비트 정수 변수는 얼마나 큰 값을 나타

낼 수 있나요?

$$2^7 \times 2^{25} \approx 40 \text{ 억}$$

1장: 0에서 1로

# 번호 시스템: 더하 기

# 추가

- 십진수

$$\begin{array}{r} 11 \leftarrow \text{캐리} \\ 3734 \\ + 5168 \\ \hline 8902 \end{array}$$

- 바이너리

$$\begin{array}{r} 11 \leftarrow \text{캐리} \\ 1011 \\ + 0011 \\ \hline 1110 \end{array}$$

# 이진 덧셈 예제

- 다음을 추가합니다.

4비트 이진

수

수를 추가합니다.

- 다음 4비트 이진

$$\begin{array}{r} 1001 \\ + 0101 \\ \hline \end{array}$$

$$\begin{array}{r} 1011 \\ + 0110 \\ \hline \end{array}$$

1

---

# 오버플로

- 디지털 시스템은 **고정된 비트 수**로 작동합니다.
- 오버플로: 결과물이 너무 커서 사용 가능한 비트 수에 맞지 않는 경우
- 이전 예제  $11 + 6$  참조

1장: 0에서 1로

# 번호 시스템: 서명 된 숫자



# 서명된 이진수

- 부호/크기 숫자

| Signed magnitude binary |           |   |   | Decimal |
|-------------------------|-----------|---|---|---------|
| Sign                    | Magnitude |   |   |         |
| 0                       | 1         | 0 | 1 | +5      |
| 1                       | 1         | 0 | 1 | -5      |

- 2의 정수

0 0 0 1 0 1 0 0 → Binary number (+20)

1 1 1 0 1 0 1 1 → One's complement

|                 |                 |
|-----------------|-----------------|
| 1 1 1 0 1 0 1 1 |                 |
| + 1             |                 |
| 1 1 1 0 1 1 0 0 | → 2s complement |

(-20)

# 부호/크기 숫자

- 1 부호 비트, N-1 크기 비트
- 부호 비트는 가장 중요한(가장 왼쪽) 비트입니다.

– 양수: 부호 비트 = 0

$$A: \{a_{N-1}, a_{N-2}, \dots, a_2, a_1, a_0\}$$

– 음수: 부호 비트 = 1

$$A = (-1)^{a_{N-1}} \sum_{i=0}^{N-2} a_i 2^i$$

- 예:  $\pm 6$ 의 4비트 부호/마그 표현:

+6 = **0110**

- 6 =

- N비트 부호/크기 숫자의 범위입니다:

# 부호/크기 숫자

## 문제:

- 덧셈이 작동하지 않습니다(예:  $-6 + 6$ ):

$$\begin{array}{r} 1110 \\ + 0110 \\ \hline \end{array}$$

10100 (잘못되었습니다!)

- $0(\pm 0)$ 의 두 가지 표현:

1000

0000

# 2의 정수

- 부호/크기 숫자와 같은 문제가 없습니다:

– 추가 작업

– 0에 대한 단일 표현

0 0 0 1 0 1 0 0 → Binary number (+20)

1 1 1 0 1 0 1 1 → One's complement

|                 |
|-----------------|
| 1 1 1 0 1 0 1 1 |
| + 1             |
| 1 1 1 0 1 1 0 0 |

→ 2s complement (-20)

# 2의 정수

- msb의 가중치는  $-2^{N-1}$ 입니다.

$$A = a_{N-1}(-2^{N-1}) + \sum_{i=0}^{N-2} a_i 2^i$$

- 가장 양수인 4비트 숫자입니다: 0111
- 대부분 음수 4비트 숫자입니다: 1000
- 가장 중요한 비트는 여전히 부호를 나타냅니다(1 = 음수, 0 = 양수).



- N비트 2의 <sup>11</sup>정수 범위입니다:

# 기호 반전하기

- 2의 정수 부호를 반대로 하는 방법

1. 비트 반전
2. 1 추가

- **예시:**  $3_{10} = 0011_2$ 의 부호 반전

1. **1100**
- 2.

역사적으로 이러한 부호 반전 방법을 "두 개의 보수를 취하기"라고 불렀습니다: "두 가지의 보수

를 취하기"라고 불렀습니다. 하지만 이 용어는 혼란스러울 수 있으므로 대신 "부호 반전"이라고 부릅니다.

## 2의 보완 예시

- $610_{10} = 01102_{10}$ 의 부호를 반전시킵니다.
  - 1.
  - 2.
- 두 개의 정수  $1001_2$  의 소수값은 무엇입니까?
  - 1.
  - 2.

## 2의 보완 추가

- 2의 보수수를 사용하여  $6 + (-6)$ 을 더합니다.

$$\begin{array}{r} 0110 \\ + 1010 \\ \hline \end{array}$$

- 2의 보수수를 사용하여  $-2 + 3$  더하기

$$\begin{array}{r} 1110 \\ + 0011 \\ \hline \end{array}$$

# 빼기

- 부호를 뒤집고 더하여 2의 보수를 뺍니다.
- 2의 보수를 취하여 역 부호화하기
- 예:  $3 - 5 = 3 + (-5)$

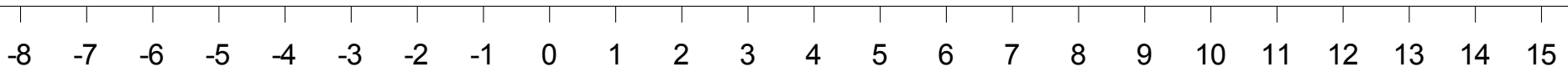
$$\begin{array}{r} 0011 \quad 3 \\ + \quad -5 \\ \hline 101 \end{array}$$

$$\begin{array}{r} 1 \\ \hline 1110 \end{array} - 2$$

# 번호 체계 비교

| 번호 체계   | 범위                      |
|---------|-------------------------|
| 서명되지 않음 | $[0, 2N-1]$             |
| 부호/크기   | $[-(2n-1-1), 2n-1-1]$   |
| 2의 보완   | $[-2^{n-1}, 2^{n-1}-1]$ |

예를 들어 4비트 표현이 있습니다:



서명되지

0000 0001 0010 0011 0100 0101 0110 0111

1000 1001 1010 1011 1100 1101 1110 1111

않음

1000 1001



1010 1011    1100 1101 1110 1111 0000 0001 0010 0011 0100 0101 0110 0111

1111   1110   1101   1100   1011   1010   1001    $\begin{smallmatrix} 0000 \\ 1000 \end{smallmatrix}$  0001 0010 0011 0100 0101 0110 0111

부호/크기

1장: 0에서 1로

**번호 시스템: 내선**

**번호**

# 비트 폭 증가

숫자를  $N$ 비트에서  $M$ 비트로 확장( $M > N$ ) :

- 2의 보수 숫자에 대한 **부호 확장**
- 부호 없는 숫자의 **영 확장자**

# 부호 확장

- MSB에 복사된 서명 비트
- 숫자 값은 동일합니다.
- 예 1:
  - 3의 4비트 표현 = 0011
  - 8비트 부호 확장 값: 00000011
- 예 2:

- 5의 4비트 표현 = 1011
- 8비트 부호 확장 값: 11111011

# 제로 익스텐션

- 0이 msb에 복사된 경우
- 음수에 대한 값 변경
- 예 1:

-4비트 값  $= 0011 = {}_{310}$

– 8비트 영 확장 값:  $00000011 = {}_{310}$

- 예: 2:

- 4비트 값  $= 1011 = -5_{10}$
- 8비트 영 확장 값:  $00001011 = 11_{10}$

1장: 0에서 1로

# 로직 게이트



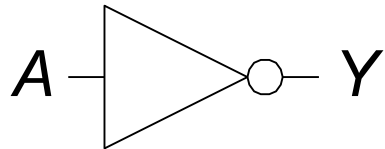
# 논리 게이트

- 논리 함수를 수행합니다:
  - 반전(NOT), AND, OR, NAND, NOR 등
- 단일 입력:
  - NOT 게이트, 버퍼
- 두 개의 입력:
  - 밋, 또는, 또는, 낸드, 노르, 노르

- **다중 입력**

# 단일 입력 로직 게이트

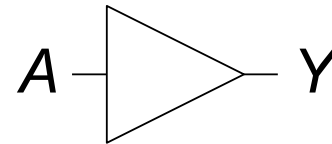
**NOT**



$$Y = \overline{A}$$

| A | Y |
|---|---|
| 0 |   |
| 1 |   |

**BUF**

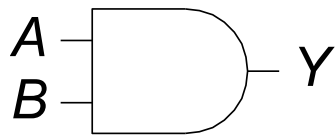


$$Y = A$$

| A | Y |
|---|---|
| 0 |   |
| 1 |   |

# 2입력 로직 게이트

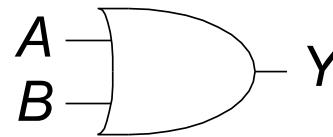
**AND**



$$Y = AB$$

| A | B | Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

**또는**

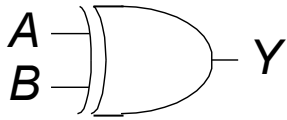


$$Y = A + B$$

| A | B | Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

# 더 많은 2입력 로직 게이트

## XOR



$$Y = A \oplus B$$

| A | B | Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

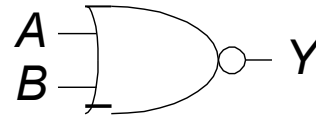
## NAND



$$Y = \overline{A \cdot B}$$

| A | B | Y |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

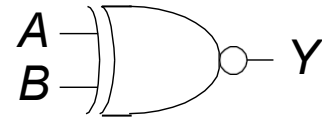
## NOR



$$Y = \overline{A + B}$$

| A | B | Y |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

## XNOR



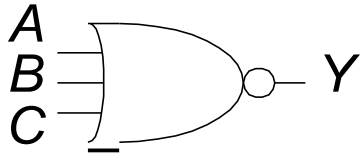
$$Y = \overline{A \oplus B}$$

| A | B | Y |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

... 입력이 같을 때  
TRUE이므로 이퀄리티  
게이트라고 합니다.

# 다중 입력 로직 게이트

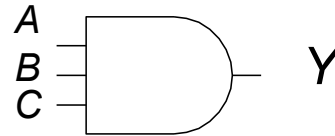
## NOR3



$$Y = \overline{A+B+C}$$

| A | B | C | Y |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 |   |
| 0 | 1 | 0 |   |
| 0 | 1 | 1 |   |
| 1 | 0 | 0 |   |
| 1 | 0 | 1 |   |
| 1 | 1 | 0 |   |
| 1 | 1 | 1 |   |

## AND3



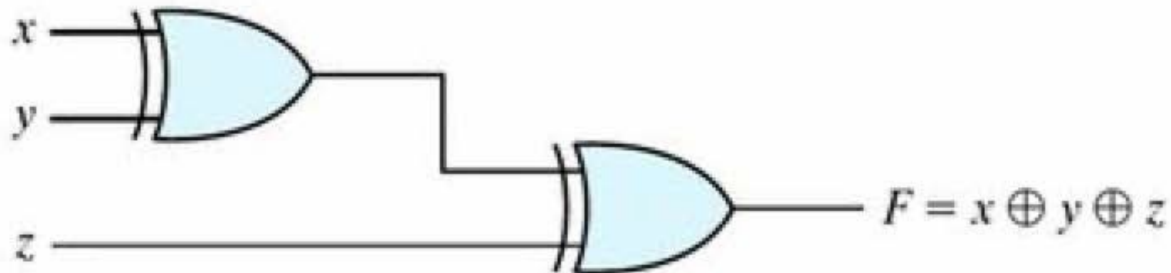
$$Y = ABC$$

| A | B | C | Y |
|---|---|---|---|
| 0 | 0 | 0 |   |
| 0 | 0 | 1 |   |
| 0 | 1 | 0 |   |
| 0 | 1 | 1 |   |
| 1 | 0 | 0 |   |
| 1 | 0 | 1 |   |
| 1 | 1 | 0 |   |
| 1 | 1 | 1 |   |

진리 테이블 행  
은 이진 순서로  
나열됩니다.

# 다중 입력 XOR

- 홀수 패리티



(a) Using 2-input gates



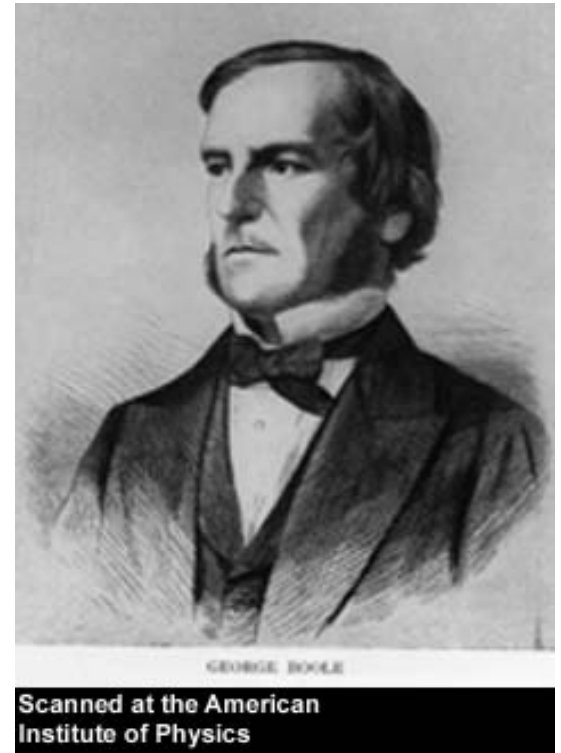
(b) 3-input gate

| $x$ | $y$ | $z$ | $F$ |
|-----|-----|-----|-----|
| 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   |

(c) Truth table

# 조지 불, 1815-1864

- 노동 계급 부모에게서 태어난
- 수학을 독학으로 배워 아일랜드 퀸즈 칼리지 교수진에 합류했습니다.
- *사상의 법칙에 대한 탐구*(1854) 저술 (1854)
- 이진 변수 도입
- 세 가지 기본 논리 연산에 대해 소





# 개합니다: AND, OR, NOT

1장: 0에서 1로

# 논리 레벨

# 논리 레벨

- 이산 전압은 1과 0을 나타냅니다.
- 예를 들어
  - 0 = 접지(GND) 또는 0볼트
  - 1 =  $V_{DD}$  또는 5V
- 4.99볼트는 어때요? 0인가요, 1인가요?
- 3.2볼트는 어떨까요?

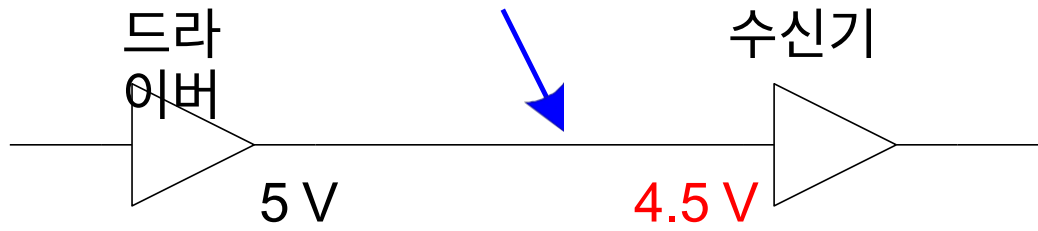
# 논리 레벨

- 1과 0의 전압 *범위*
- *노이즈*를 허용하는 다양한 입력 및 출력 범위

# 노이즈란 무엇인가요?

- **신호를 저하시키는 모든 것**
  - 예를 들어 저항, 전원 공급 장치 노이즈, 주변 전선과의 결합 등입니다.
- **예: 게이트(드라이버)는 5V를 출력하지만**  
긴 와이어의 저항으로 인해 수신기는 4.5V를 얻습니다.

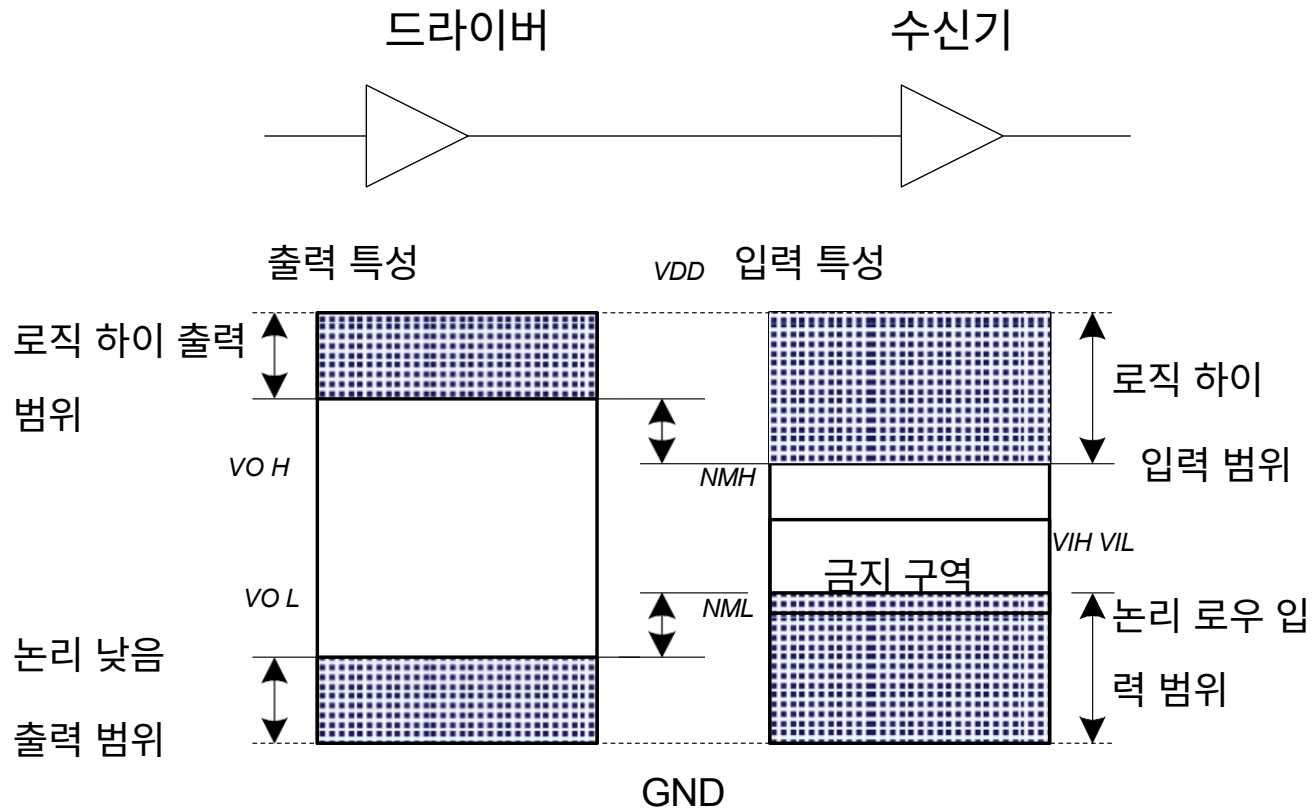
소음



# 정적 규율

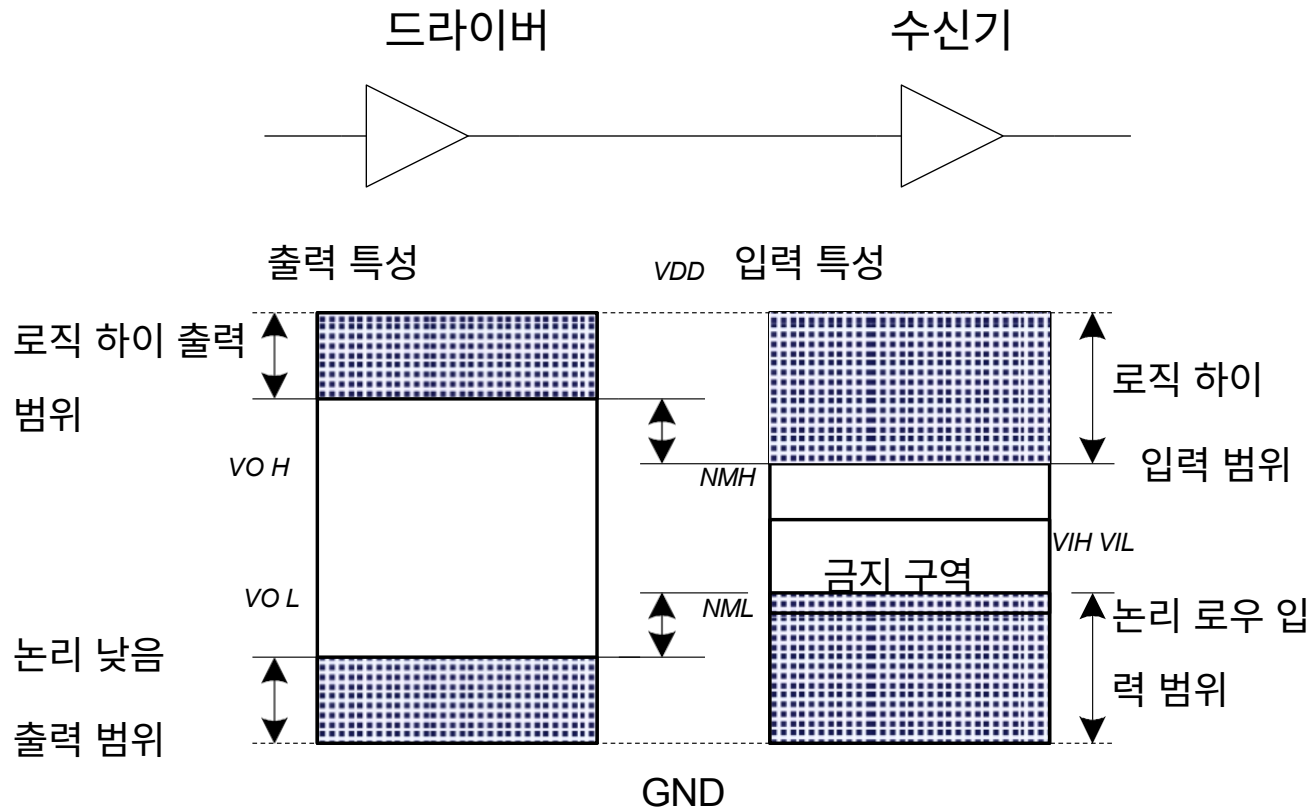
- 논리적으로 유효한 입력이 있는 경우 모든 회로 소자는 논리적으로 유효한 출력을 생성해야 합니다.
- 제한된 범위의 전압을 사용하여 불연속 값을 표현합니다.

# 노이즈 마진





# 노이즈 마진

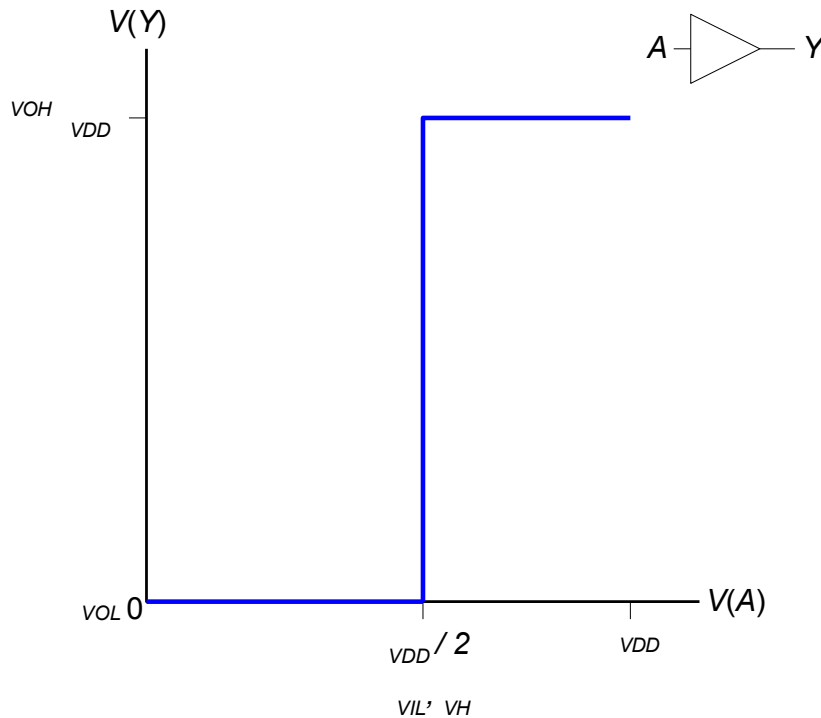


높은 노이즈 마진:  $NMH =$

낮은 노이즈 마진:  $NML =$

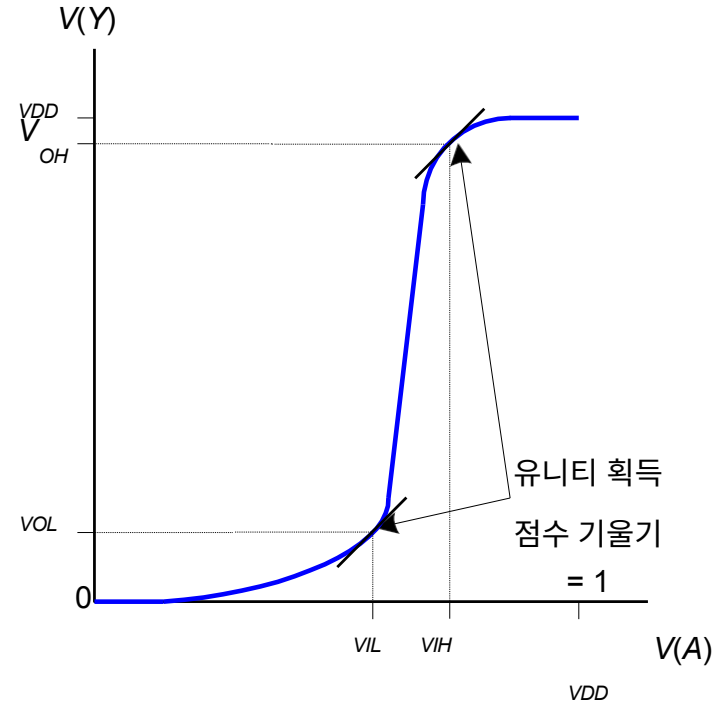
# DC 전송 특성

이상적인 버퍼:



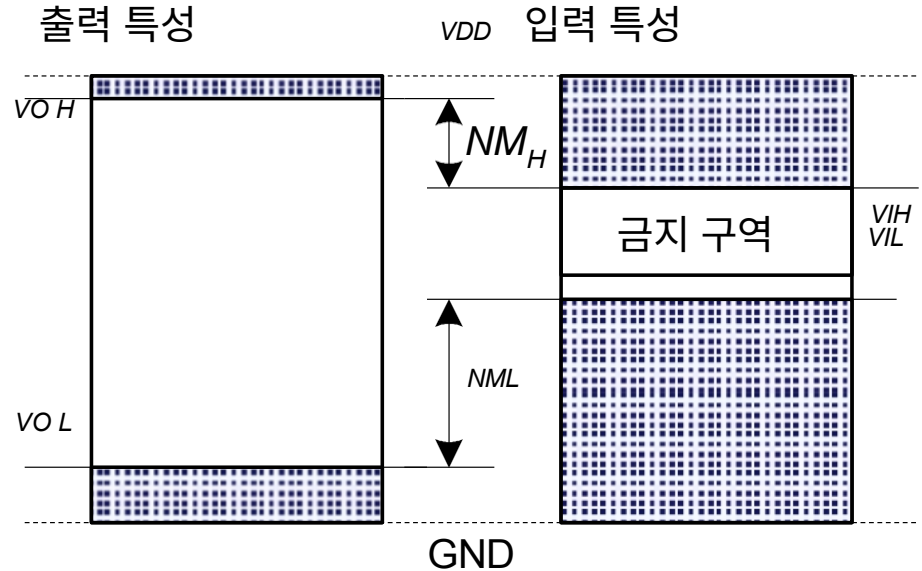
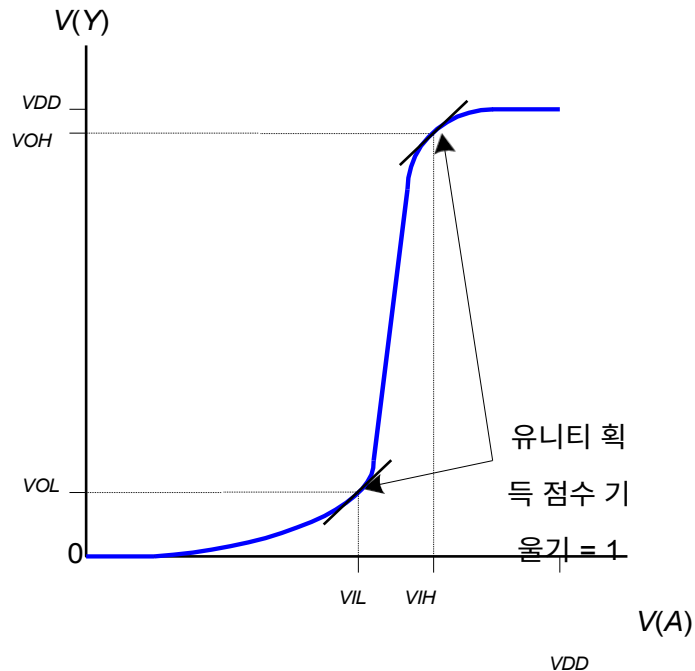
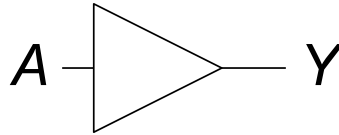
$$NMH = NML = V_{DD}/2$$

실제 버퍼:



$$NMH, NML < V_{DD}/2$$

# DC 전송 특성



# VDD 스케일링

- 1970년대와 1980년대,  $VDD = 5V$
- $VDD$ 가 떨어졌습니다.
  - 작은 트랜지스터를 튀기지 마십시오.
  - 전력 절약
- 3.3V, 2.5V, 1.8V, 1.5V, 1.2V, 1.0V, ...
  - 공급 전압이 다른 칩을 연결할 때는 주의하세요.

# VDD 스케일링

- 1970년대와 1980년대,  $VDD = 5V$
- $VDD$ 가 떨어졌습니다.
  - 작은 트랜지스터를 튀기지 마십시오.
  - 전력 절약
- 3.3V, 2.5V, 1.8V, 1.5V, 1.2V, 1.0V, ...
  - 공급 전압이 다른 칩을 연결할 때는 주의하세요.

# 로직 제품군 예제

| 로직 제품군  | VDD             | VIL  | VIH  | VOL  | VOH  |
|---------|-----------------|------|------|------|------|
| TTL     | 5 (4.75 - 5.25) | 0.8  | 2.0  | 0.4  | 2.4  |
| CMOS    | 5 (4.5 - 6)     | 1.35 | 3.15 | 0.33 | 3.84 |
| LVTTL   | 3.3 (3 - 3.6)   | 0.8  | 2.0  | 0.4  | 2.4  |
| LVC MOS | 3.3 (3 - 3.6)   | 0.9  | 1.8  | 0.36 | 2.7  |

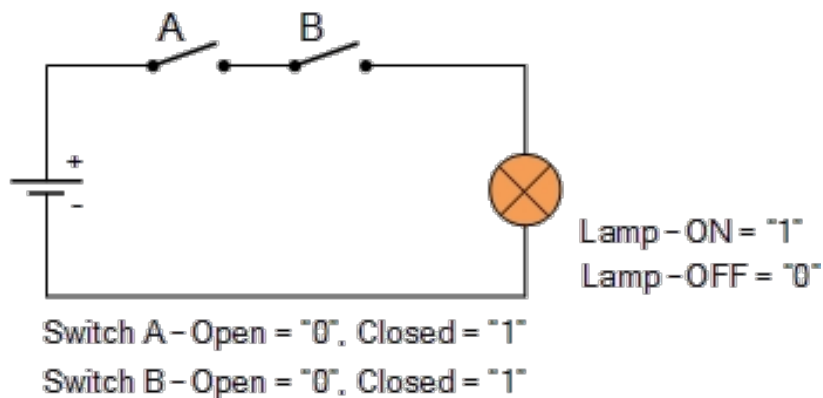
- 트랜지스터-트랜지스터 로직(TTL)
- 상보형 금속 산화막 반도체(CMOS)

1장: 0에서 1로

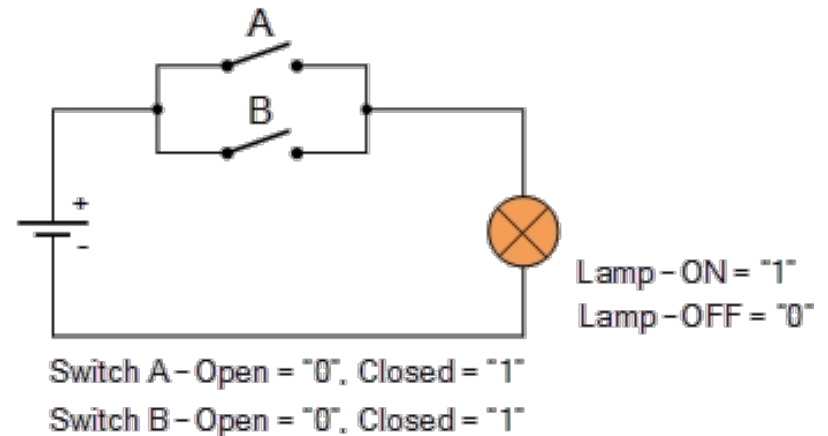
# CMOS 트랜지스터

# 스위치

- AND 연산:  $1 \times 1 = 1$



- OR 연산:  $0 + 0 = 0$

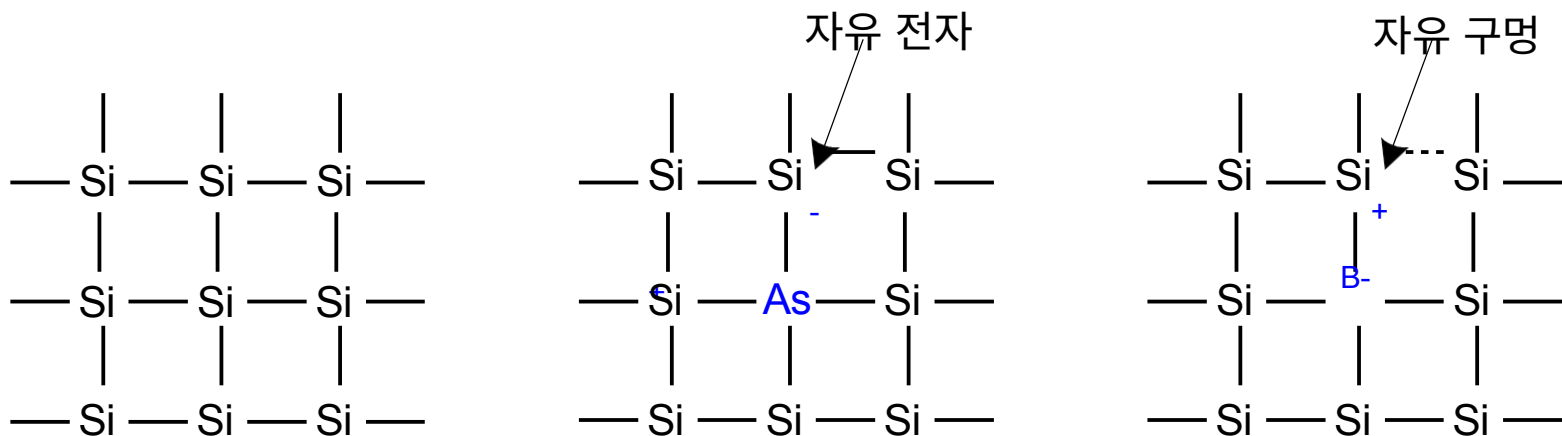


*How can we make electronically controlled switches?*



# 실리콘

- 순수 실리콘은 전도성이 좋지 않습니다(무로 요금 없음).
- 도핑된 실리콘은 좋은 반도체입니다(무로).
  - n형(자유 음전하, 전자)
  - p형(자유 양전하, 정공)



| | |  
실리콘 격자

| | |  
n-타입 (\* 비소: 비소)

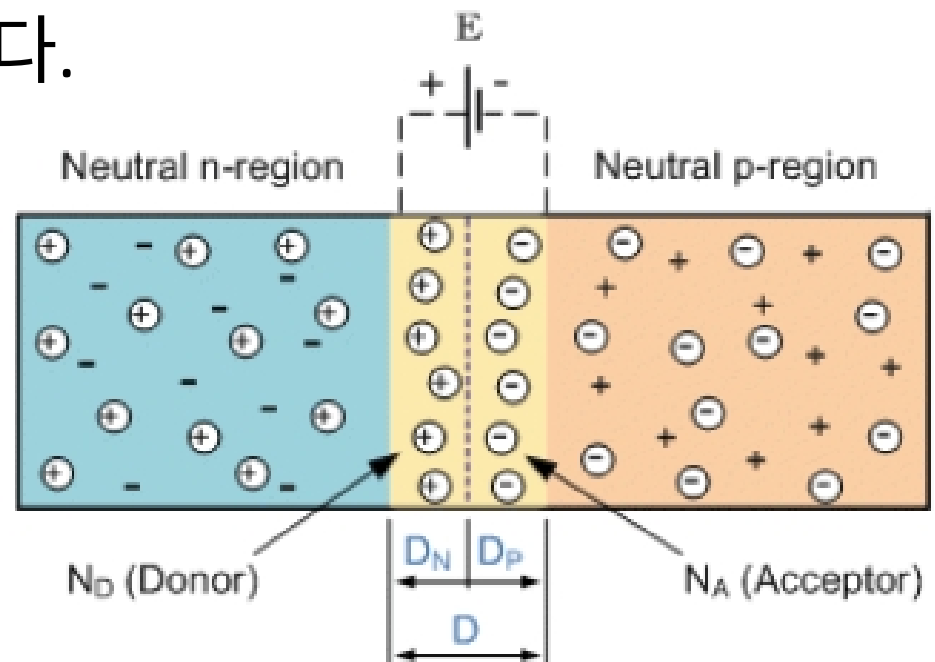
| | |  
p-타입 (\* B: 붕소)

# 반도체

- 기능

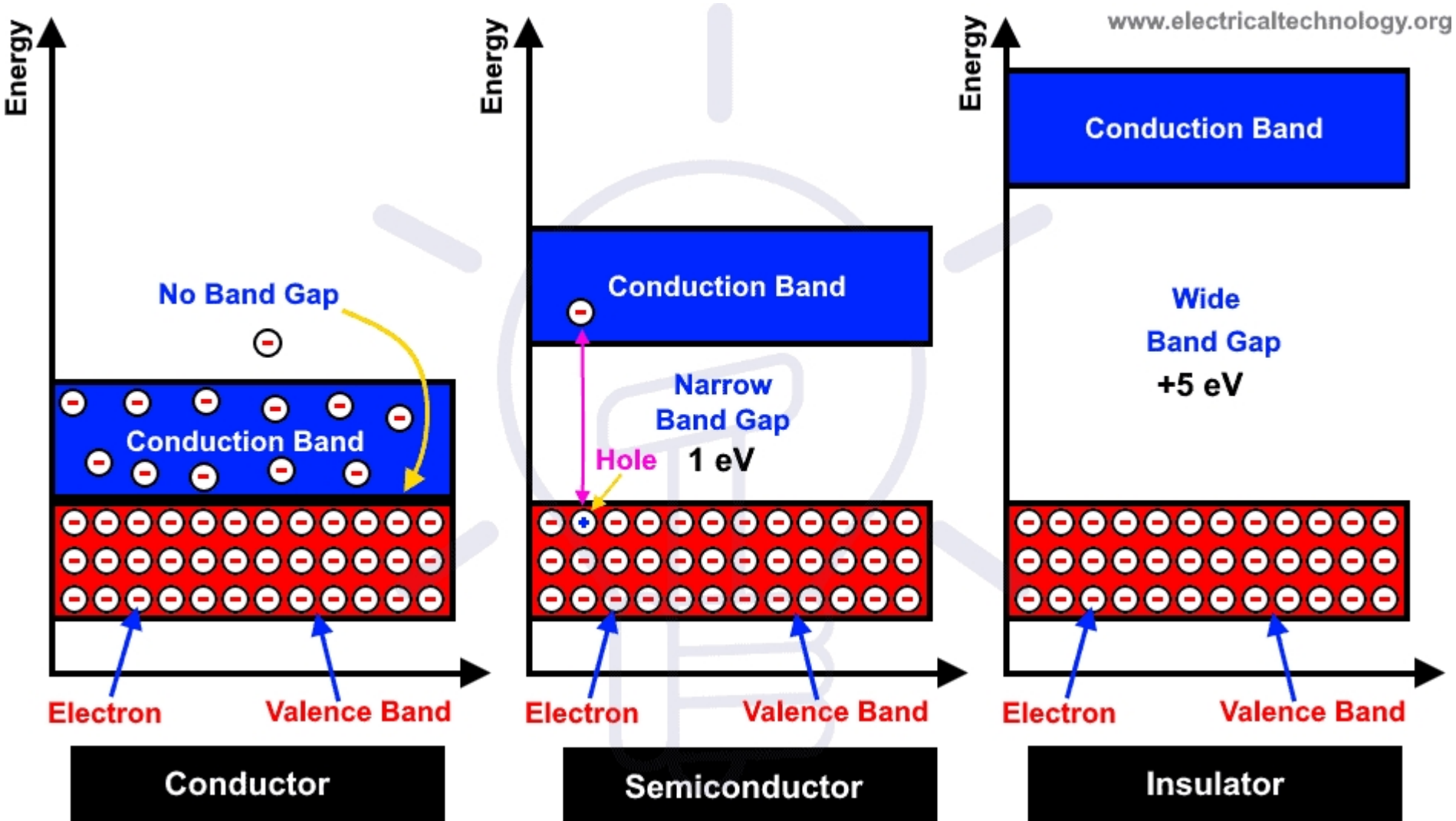
- 저온 절연체
- 전자가 원자가 밴드에서 점프하도록 에너지를 주면 전도체가 됩니다.

전도 대역으로



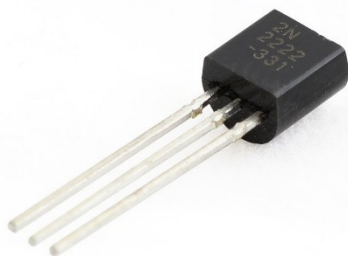
72     디지털 디자인 및 컴퓨터 아키텍처제로에서 원으로

# 반도체



# 트랜지스터

- 반도체인 실리콘으로 제작된 트랜지스터
- 트랜지스터로 구축된 로직 게이트
- 3포트 전압 제어 스위치
  - 3 포트의 전압에 따라 2개 포트 연결<sup>rd</sup> 포트
  - $g$ 가 1일 때  $d$ 와  $s$ 가 연결(ON)됩니다.



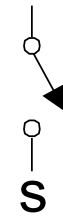
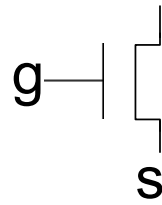
$d$

$g = 0$

$d$

$g = 1$

$d$



OFF

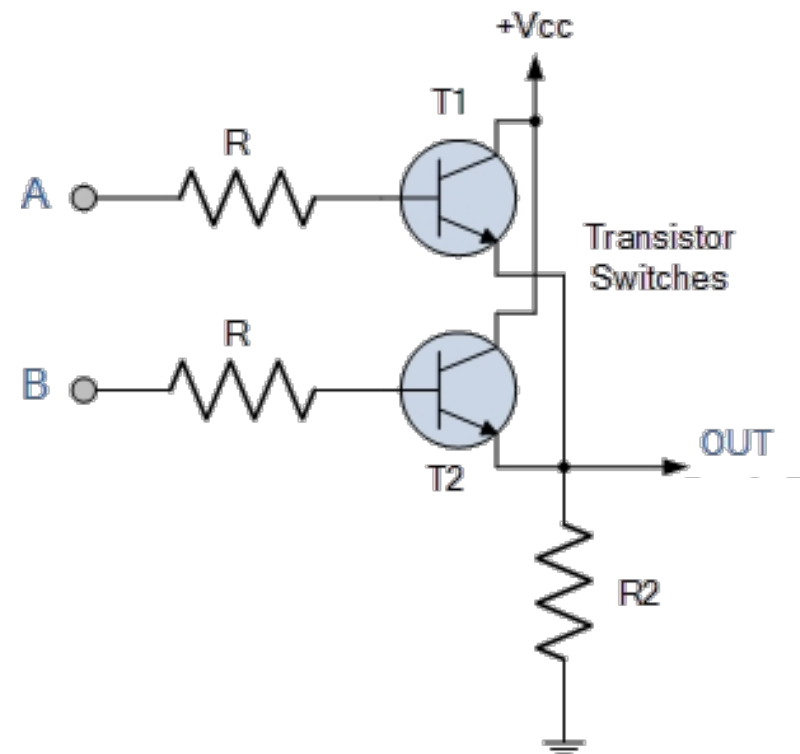
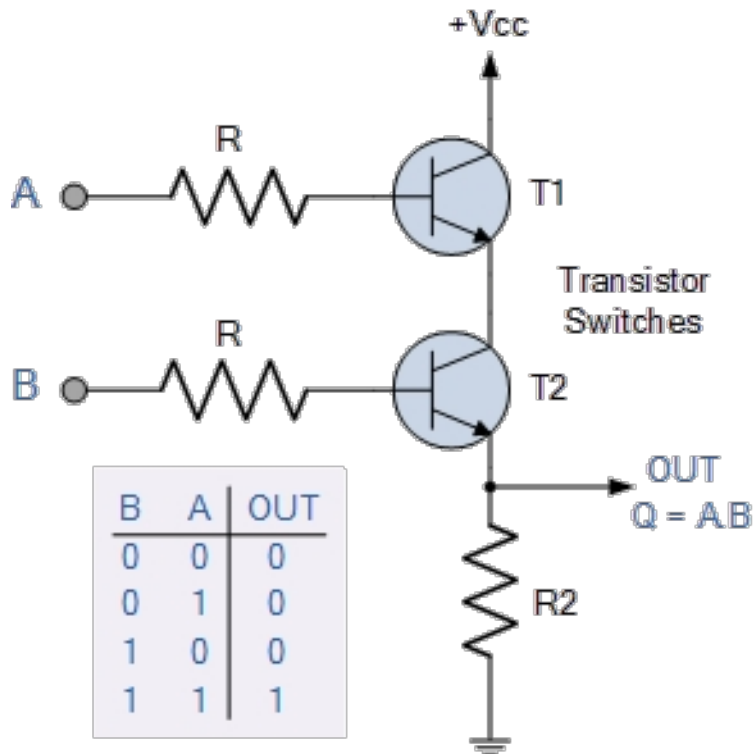


ON

# 트랜지스터가 있는 로직 게이트

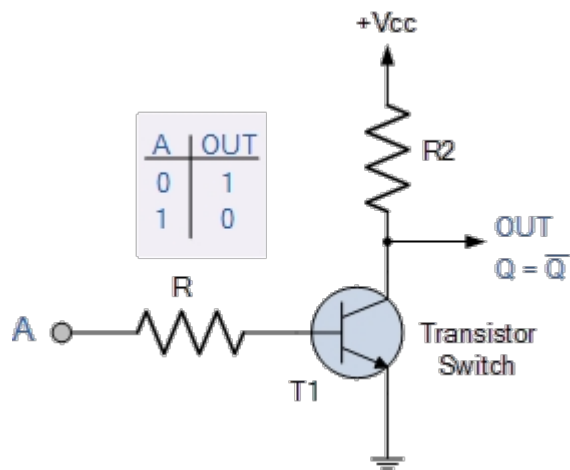
- AND 게이트

- OR 게이트

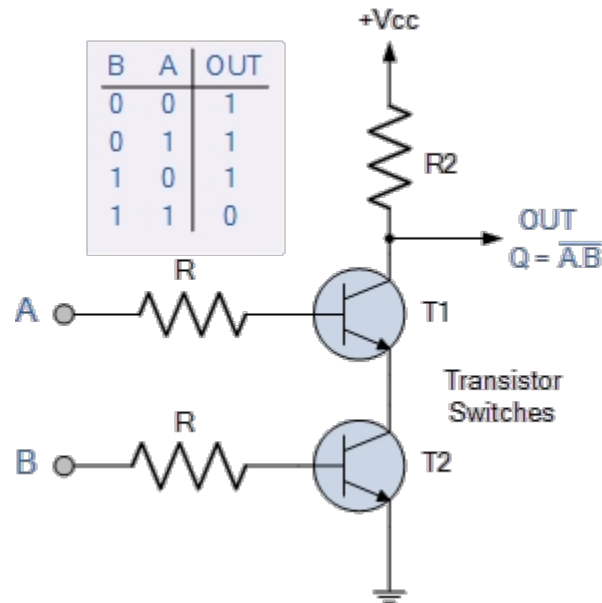




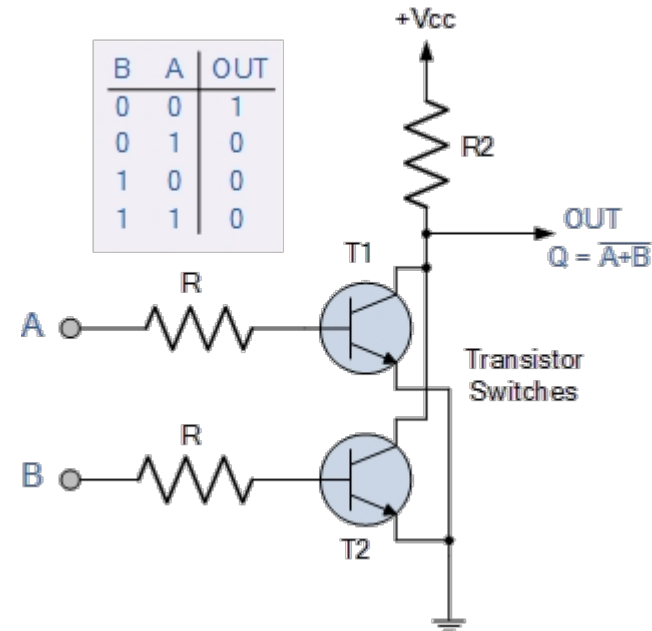
# 트랜지스터가 있는 로직 게이트



NOT



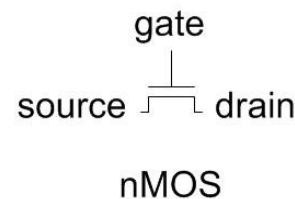
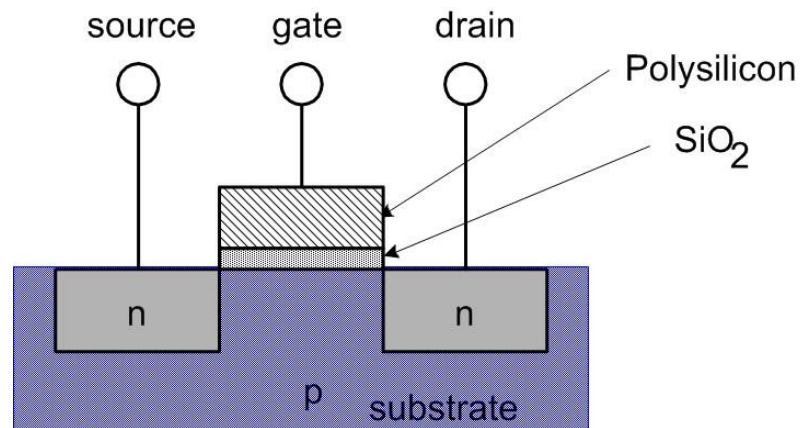
NAND



NOR

# MOS 트랜지스터

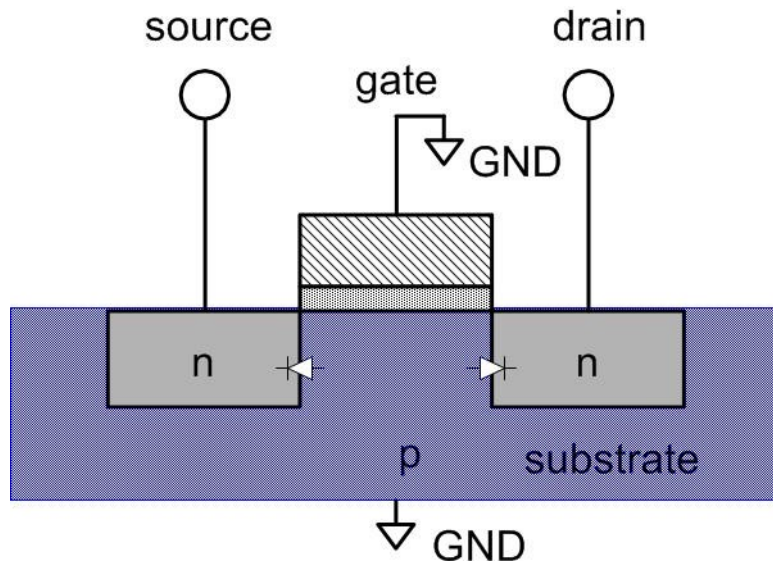
- 금속 산화물 실리콘(MOS) 트랜지스터:
  - 폴리실리콘(기존 금속) 게이트
  - 산화물(이산화규소) 절연체
  - 도핑된 실리콘



# 트랜지스터: nMOS

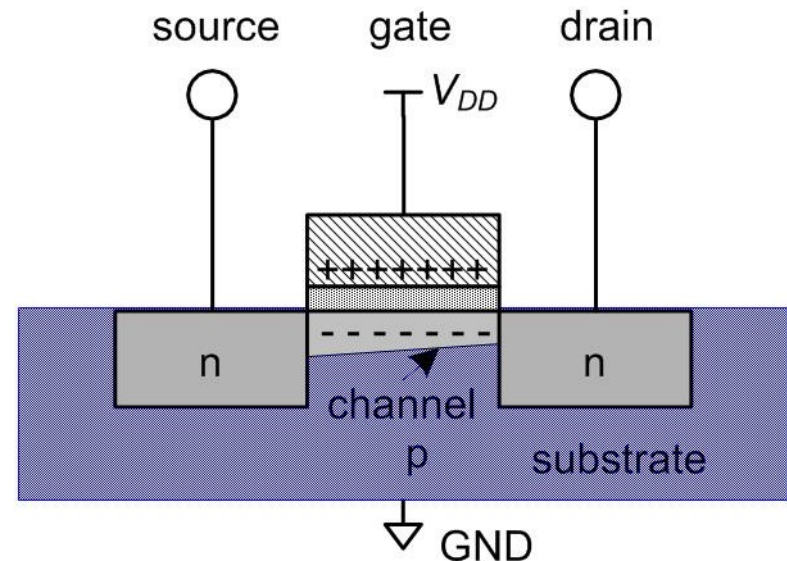
게이트 = 0

꺼짐(소스와 드레인  
간 연결 없음)



게이트 = 1

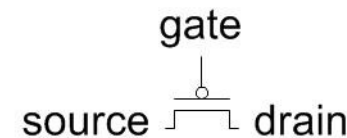
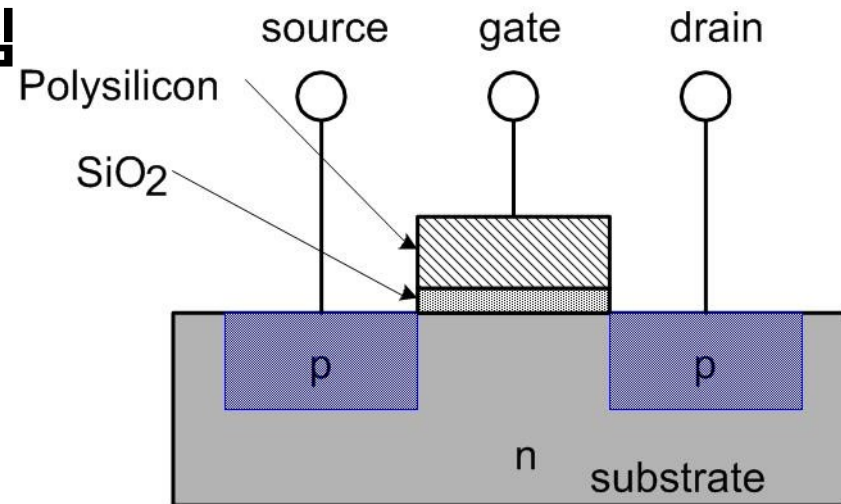
켜짐(소스-드레인 간 채널)



# 트랜지스터: pMOS

**pMOS 트랜지스터는 반대편에 있습니다.**

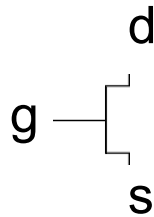
- 게이트 = 0일 때 켜짐
- 게이트 = 1일 때 꺼짐



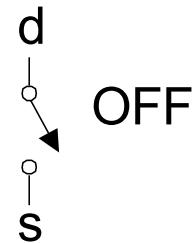
pMOS

# 트랜지스터 기능

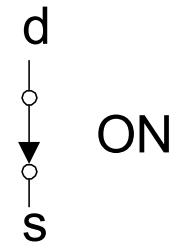
nMOS



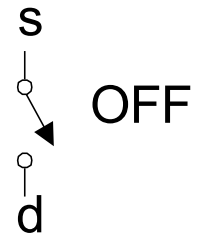
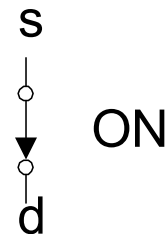
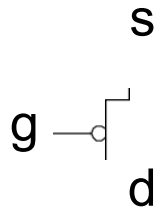
$g = 0$



$g = 1$



pMOS

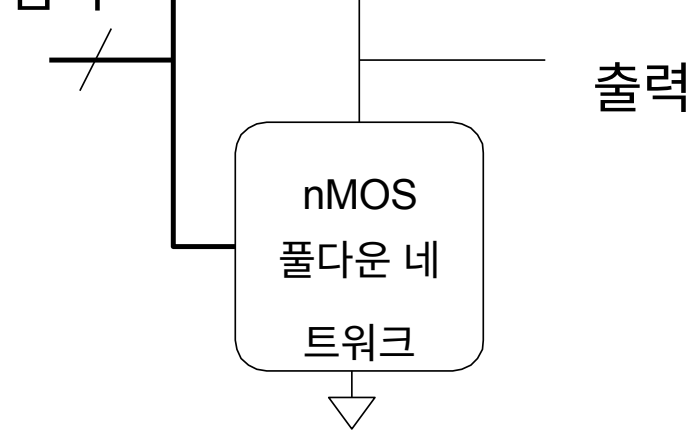


1장: 0에서 1로

# 트랜지스터의 게이트

# 트랜지스터 기능

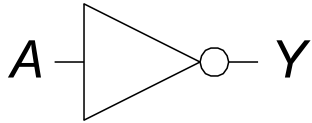
- **nMOS**: 양호한 0을 전달하므로 소스를 GND에 연결합니다.
- **pMOS**: 양호한 1을 전달하므로 소스를  $V_{DD}$ 에 연결합니다.





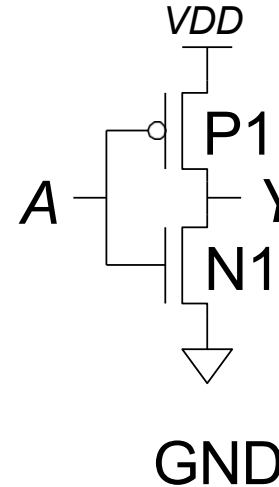
# CMOS 게이트: NOT 게이트

**NOT**



$$Y = \overline{A}$$

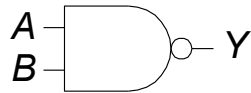
| A | Y |
|---|---|
| 0 | 1 |
| 1 | 0 |



| A | P1 | N1 | Y |
|---|----|----|---|
| 0 |    |    |   |
| 1 | 켜  |    |   |
|   |    | 켜  |   |

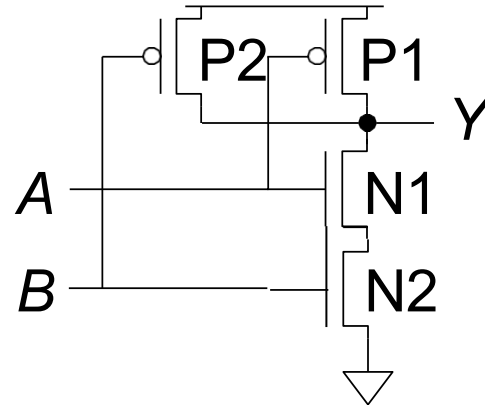
# CMOS 게이트: NAND 게이트

## NAND



$$Y = \overline{AB}$$

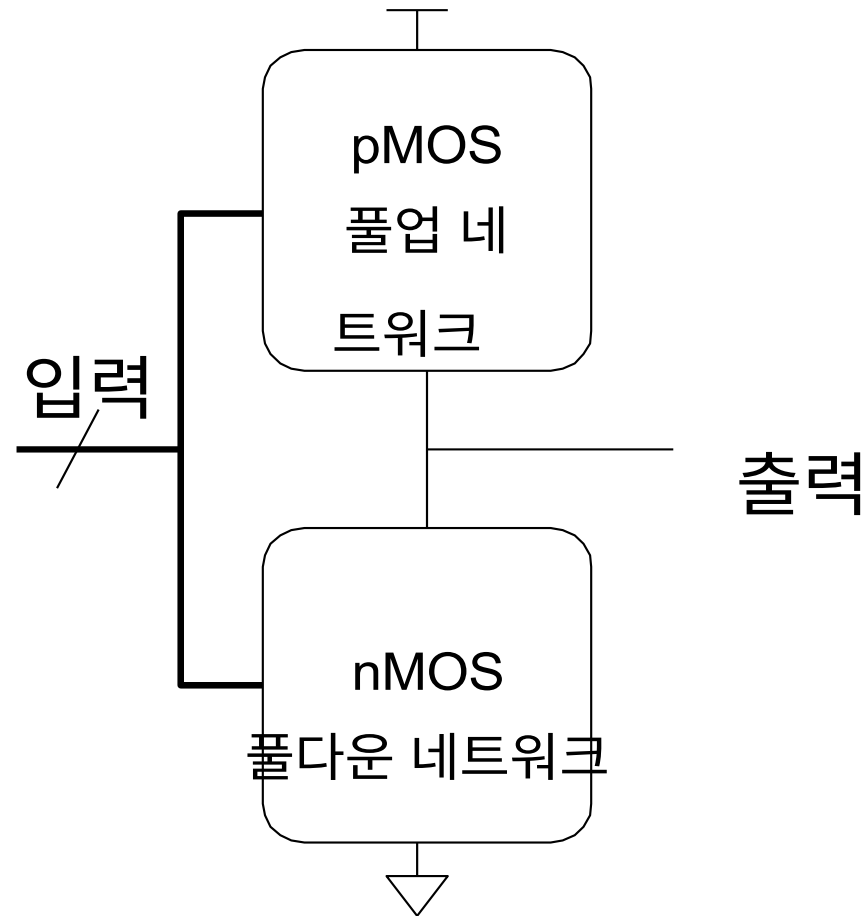
| A | B | Y |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |



| A | B | P1 | P2 | N1 | N2 | Y |
|---|---|----|----|----|----|---|
| 0 | 0 |    |    |    |    |   |
| 0 | 1 | —  | —  |    |    |   |
| 1 | 0 | —  |    |    | —  |   |
| 1 | 1 |    | —  | —  |    |   |

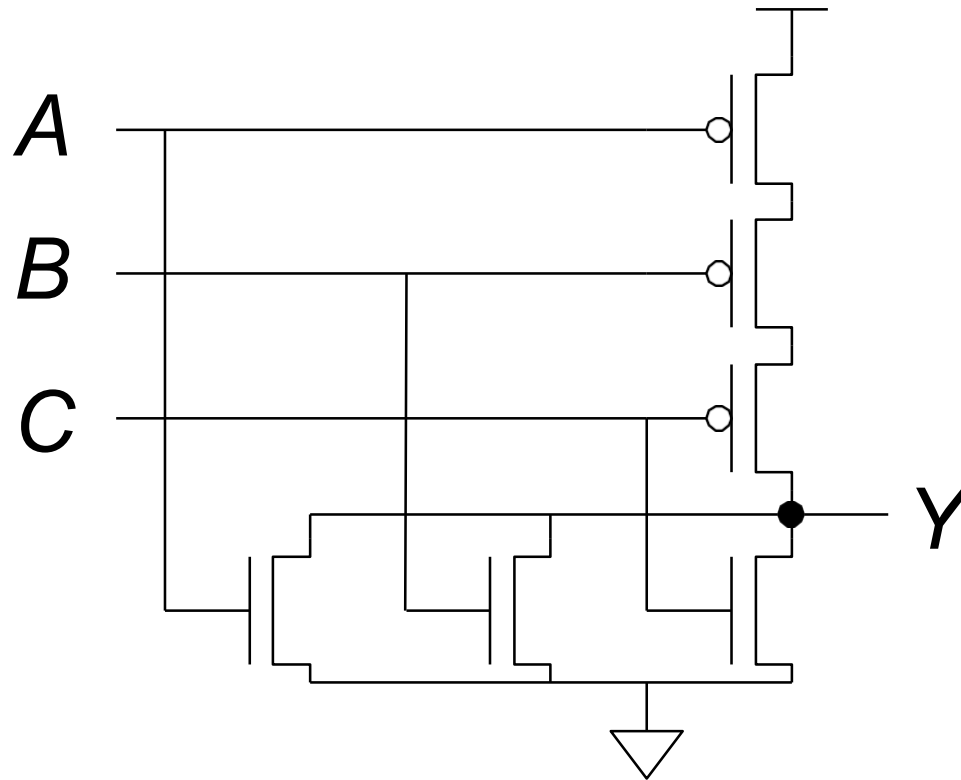
기 기

# CMOS 게이트 구조



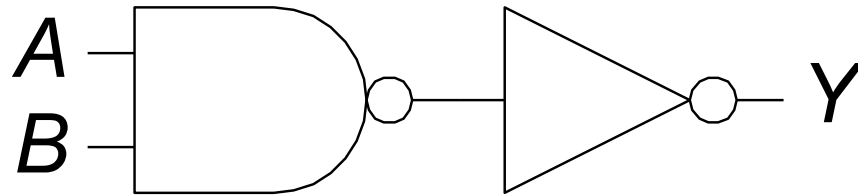
# NOR3 게이트

3입력 NOR 게이트는 어떻게 구축하나요?





# AND2 게이트

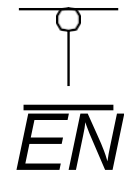
2입력 AND 게이트는 어떻게 구축하나요?



# 전송 게이트

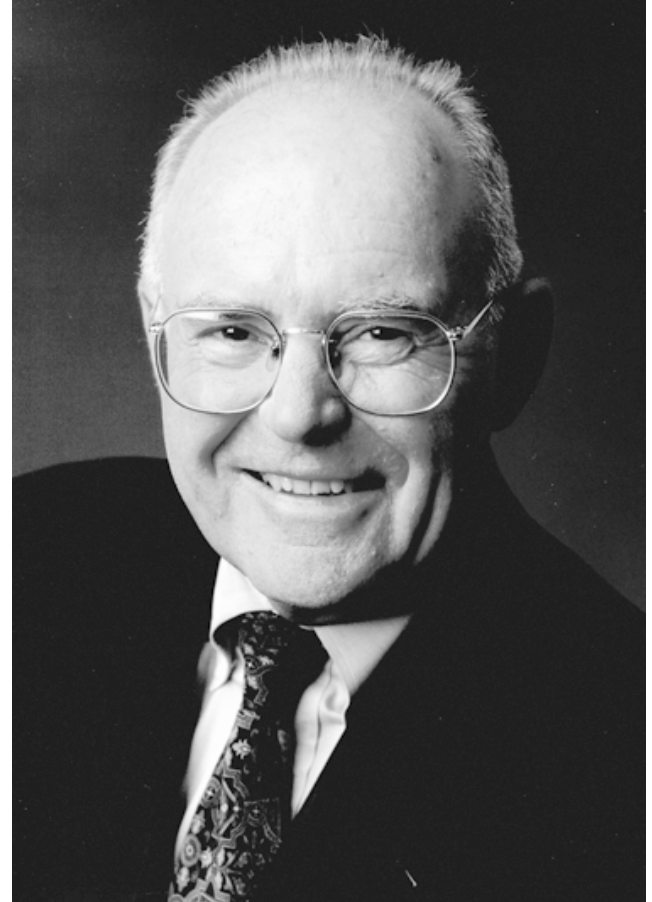
- nMOS 패스 1의 성능 저하
- pMOS 통과 0의 불량
- 두 패스의 병렬 조합 또는 *전송 게이트*가 더 나은 스위치입니다.
  - 0과 1을 모두 잘 통과합니다.
- $\overline{EN} = 1$ 이면 스위치가 켜져 있습니다: 
- $\overline{EN} = 0$ 이고 A가 B에 연결되어 있습니다.  B

- When  $\overline{EN} = 0$ , the switch is OFF:
  - A가 B에 연결되어 있지 않습니다.



# 고든 무어, 1929-

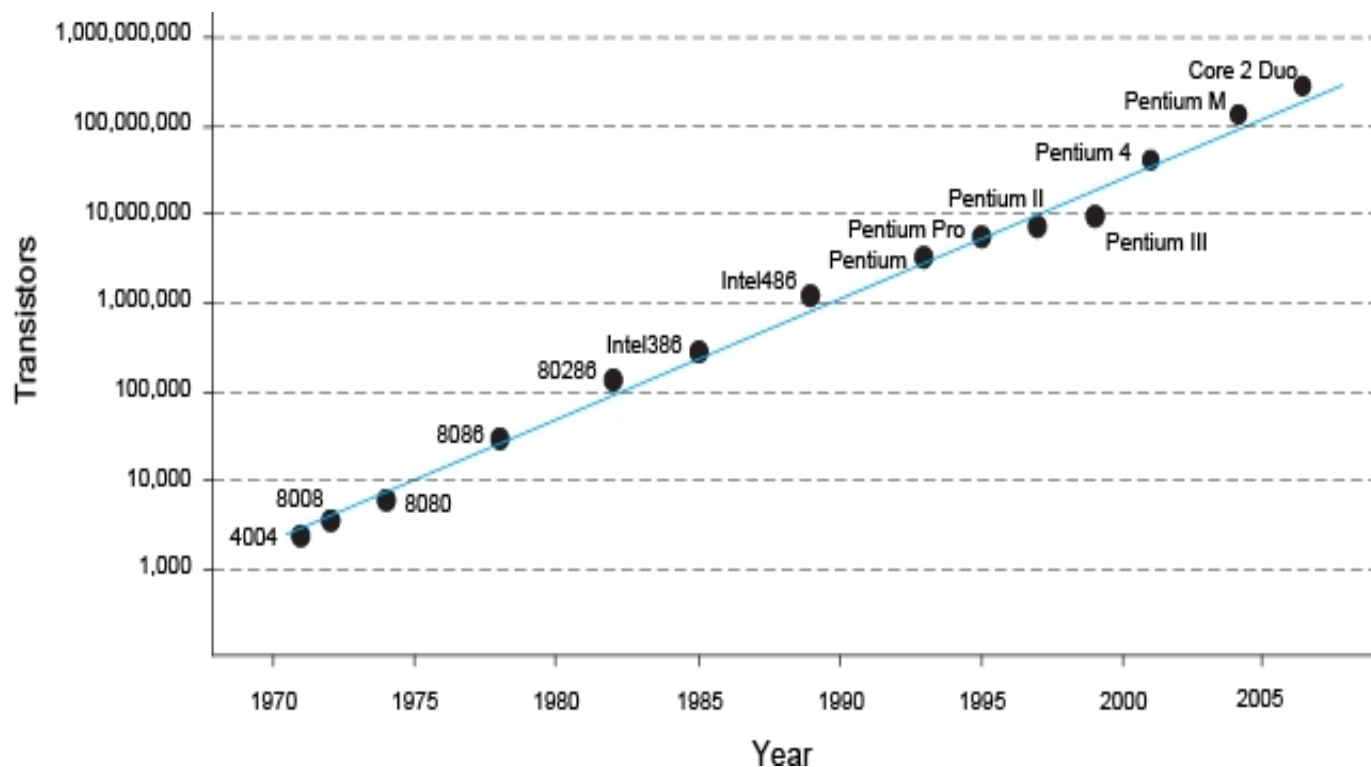
- 1968년 로버트 노이스와 함께 인텔을 공동 설립했습니다.
- **무어의 법칙:** 컴퓨터 칩의 트랜지스터 수가 매년 두 배로 증가 (1965년 관측)
- 1975년 이후 트랜지스터 수는 2년마다 두 배씩 증가했습니다





- .
- 코러러리: 트랜지스터는 더 빨라지고 전력은 낮아집니다.

# 무어의 법칙



"자동차가 컴퓨터와 같은 개발 주기를 따랐다면 오늘날 롤스로이스의 가격은 100달러에 달하고 연비는 100만 마일에 달하며 1년에 한 번 폭발적

으로 증가할 것입니다." (로버트 크링글리, 인포월드)

- 로버트 크링글리

1장: 0에서 1로

# 전력 소비량

# 전력 소비량

**전력 = 단위 시간당 소비되는 에너지**

- 동적 전력 소비
- 정적 전력 소비

# 동적 전력 소비

- 트랜지스터 게이트 커패시턴스를 충전하는 전력
  - 커패시턴스  $C$ 를  $V_{DD}$ 로 충전하는 데 필요한 에너지는  $CV_{DD}^2$
  - 주파수  $f$ (초당  $f$  사이클)로 실행되는 회로
  - 커패시터는 사이클당  $\alpha$ 회 충전됩니다(1에서 0까지 방전은 무료).
- 동적 전력 소비:

$$P_{dynamic} = \alpha C V_{DD}^2 f$$

# 정적 전력 소비

- 게이트가 전환되지 않을 때 소비되는 전력
- 대기상태의 *공급 전류*로 인해 발생하는  $I_{DD}$  (*누설 전류*라고도 함)
- 정적 전력 소비:

$$P_{static} = I_{DD}V_{DD}$$



# 전력 소비 예시

- 앵그리 버드를 실행하는 휴대폰의 전력 소비  
량 추정하기

- $V_{DD} = 0.8V$

- $C = 5nF(5 \times 10^{-9} \text{ 패럿})$

- $f = 2GHz(2 \times 10^9 \text{ 헤르츠})$

- $\alpha = 0.1$

- $I_{DD} = 100mA$

$$P = \alpha C V_{DD}^2 f + I_{DD} V_{DD}$$

$$\begin{aligned} &= (0.1)(5 \text{ nF})(0.8 \text{ V})^2 (2 \text{ GHz}) + (100 \text{ mA})(0.8 \text{ V}) \\ &= \mathbf{(0.64 + 0.08) \text{ w} \approx 0.72\text{w}} \end{aligned}$$

# 전력 소비 예시

- 휴대폰의 배터리가 8W인 경우, 주머니에 넣어둔 상태에서 배터리 수명을 예상해 보세요.

- $V_{DD} = 0.8V$

- $I_{DD} = 100mA$

$$P_{static} = I_{DD}V_{DD} = 0.08W$$

$$\text{배터리 수명} = \text{용량} / \text{소비량}$$

$$= (8W\text{-hr}) / (0.08W) = 100\text{시간}(4\text{일})$$