Chapter 1 :: From Zero to One
"영" 에서 "일" 까지…

Digital Design and Computer Architecture

David Money Harris and Sarah L. Harris

Lectured by Jeong-Gun Lee @ Hallym

Class Homepage: https://github.com/jeonggunlee/LogicDesign

Chapter 1 :: Topics

- **Background (배경)**
- **The Game Plan (계획)**
- **The Art of Managing Complexity (복잡도를 다루는 방법)**
- **The Digital Abstraction (디지털 추상화)**
- **Number Systems (수 표현 체계)**
- **Logic Gates (논리 회로/논리 게이트)**
- **Logic Levels (논리 수준)**
- **CMOS Transistors (CMOS 트랜지스터)**
- **Power Consumption (전력 소모)**

Background (배경)

- Big data / AI / Data Center
  - Personal Computer
  - Server
  - Mobile Processor
- Accelerator
  - 특정 기능을 빠르게 하는 초고속 프로세싱 회로
  - FPGA / ASIC / GPU …

Background (배경)

- Big dat
  - Persc
  - Serve
  - Mobi
- Accele
  - 특정 회로
  - FPG

"GPU는 거들 뿐"..AI 열풍 올라탄 FPGA의 반격

새너제이=노동균 기자

입력 2018.10.03 20:48 | 수정 2018.10.03 20:49

프로그래머블 반도체(FPGA)가 인공지능(AI) 열풍으로 승승장구했던 GPU와 한판 승부를 벌인다. FPGA 시장 선두 업체 자일링스가 데이터센터 시장을 겨냥한 가속기 카드를 내놓고 GPU를 압도한다며 실력행사에 나섰다.

빅터 펭 자일링스 CEO가 XDF 2018 기조연설 중 새로운 가속기 카드 포트폴리오 '알비오'를 선보이고 있다. /

요약정리

SK텔레콤은 AI 가속기를 어떻게 만들었을까

2019.08.16

SK텔레콤은 지난 6월 '인공지능(AI) 연산 가속기'를 발표했다. 서비스 고도화에 따라 폭증하는 방대한 데이터를 실시간으로 빠르게 처리하고 전력 효율성을 높이기 위해서다. 구글이나 페이스북 등 글로벌 IT 기업들이 자체 AI 칩을 개발하는 배경과 같은 맥락이다. SK텔레콤의 AI 가속기에는 자일링스의 FPGA(field programmable gate array)가 사용됐다.

SK텔레콤과 자일링스는 8월16일 기자간담회를 열고 자일링스 FPGA 기반 AI 가속기를 SK텔레콤 데이터센터에 적용했다고 발표했다. 국내에서 데이터센터 기반 AI 서비스에 FPGA 기반 AI 가속기를 적용한 건 이번이 처음이다. 현재 AI 가속기는 SK텔레콤의 AI 서비스 '누구'에 적용됐다. 누구 클라우드에 적용해 자동음성인식(ASR) 애플리케이션의 성능을 GPU 대비 5배 이상 끌어올렸다. 이강원 SK텔레콤 SW기술원 원장은 "AI 가속기를 사용할 경우 서버 다섯대를 카드 하나로 대체하는 효과가 있다"라며 "발열과 전기소모, 공간을 절약할 수 있어 효율적이다"라고 말했다.

Backgro

- Big d
  - Pe
  - Se
  - M
- Acce
  - 특
  - FP

로

HOME > 포커스 > 포커스

인공지능칩 'FPGA와 ASIC' 구글, 마이크로소프트까지 뛰어들다 ④

이나리 기자 | 승인 2018.02.06 16:00:04 | 수정 2018.02.06 16:10

댓글 0 | 트위터 | 페이스북

[CCTV뉴스=이나리 기자] 프로그래밍과 재설정이 가능한 비메모리 반도체의 일종인 FPGA(Field-Programmable Gate Arrays) 역시 높은 유연성 때문에 대용량 데이터 처리에 적합하다. 이런 이유로 데이터센터 프로세서에 FPGA를 같이 쓰면 전력 감소에 많은 도움을 준다. 또 FPGA가 부상하는 이유 중 하나는 범용 프로세서의 성능 향상이 한계에 달했기 때문이다. FPGA는 CPU가 감당하지 못하는 더 많은 서비스를 제공할 수 있고, CPU와 병렬로 작동하므로 전체 시스템의 혼란이나 병목현상 없이 추가적인 컴퓨팅 파워로 사용할 수 있기 때문에 최근 새로운 기술로 주목 받고 있다.

재프로그래밍이 가능한 FPGA는 칩을 번역 작업에 최적화해 사용하다가 칩 회로 구성을 다시 설정해 가상비서 서비스에 맞춰 쓸 수 있다. 따라서 FPGA는 인텔이나 AMD가 만드는 범용 프로세서와 특정 장비 전용으로 개발하는 주문형 반도체인 ASIC(Application-Specific Integrated Circuits)의 특성을 합쳤다는 평가를 받는다. 또 FPGA는 특정 함수에 맞춘 하드웨어 프로그래밍이 가능해 신경망 모델 출력 값을 빠르게 계산하는 인공지능 추론 서비스 구현에 적합하다. 이처럼 FPGA는 ASIC보다 초기 개발 비용이 저렴하고, 원하는 작업을 더 빠르게 처리할 수 있다는 것이 장점이다.

Background (배경)

- E
  -
  -
  -
- A
  -
  -

HOME > ICT > AI

인텔 FPGA, 딥 러닝을 위한 인공지능(AI) 가속화

김태만 기자 승인 2018.03.28 16:10 댓글 0

인공지능(AI)은 업계를 변혁하고, 데이터의 관리 및 해석 방식을 바꾸고, 무엇보다 가장 중요한 사람들과 기업들의 실질적 문제를 그 어느 때보다 빠르게 해결하고 있다.

28일, 발표하는 Microsoft's Bing Intelligent Search 뉴스는 인텔 FPGA(field programmable gate array) 기술이 전세계에서 가장 앞선 인공지능(AI) 플랫폼 중 하나에 어떻게 영향을 미치는지를 공개했다.

실시간 인공지능(AI)을 이용한 Bing 검색 엔진으로의 발전은 보편적 검색 결과를 제공하는 수준을 넘어 더 많은 일을 처리하고 더 많은 지식을 습득할 수 있게 도움을 준다. Bing Intelligent Search는 웹 페이지들 대신 해답을 제공하고 시스템이 단어 및 그 단어 뒤에 숨겨진 의미, 문맥 및 검색 의도를 이해할 수 있게 해준다.

Background (배경)

- B
  -
  -
  -
- A
  -
  -

클라우드, FPGA 기반 서버로 새로운 시장 열린다 - e4ds.com
www.e4ds.com/sub_view.asp?ch=31&t=2&idx=9172 ▾
2018. 5. 14. - 자일링스, FPGA 최대 장점 살린 기능 제공 지난 4월 아마존은 2018 AWS 서밋을 개최하며, 클라우드가 제공하는 다양한 기능과 혜택, 국내외 기업 ...

화웨이, FPGA 가속화 클라우드 서버로 자일링스 선택 - e4ds.com
www.e4ds.com/sub_view.asp?ch=22&t=1&idx=7346 ▾
2017. 9. 12. - 화웨이의 FPGA 가속화 클라우드 서버(FACS, FPGA Accelerated Cloud Server)는 사용자가 화웨이의 퍼블릭 클라우드에서 새로운 FPGA 기반 ...

FPGA를 지원하는 EC2 F1 인스턴스 정식 출시 | Amazon Web Services ...
https://aws.amazon.com/ko/.../ec2-f1-instances-with-fpgas-now-generally-available/ ▾
2017. 4. 20. - 지난 AWS re:Invent에서 FPGA 기능을 장착한 F1 인스턴스 개발자 ... 는 Go 프로그래밍 언어를 사용하여 FPGA를 프로그래밍 할 수 있는 클라우드 ...

FPGA 개발 시작하기 - Amazon Elastic Compute Cloud
https://docs.aws.amazon.com/ko_kr/AWSEC2/latest/.../fpga-getting-started.html ▾
FPGA 개발자 AMI 는 AFI를 개발, 테스트 및 빌드할 수 있는 도구를 제공합니다. 시스템 메모리가 최소 32GB인 모든 EC2 인스턴스(예: C5, M4, R4 인스턴스)에 FPGA ...

FPGA 및 Project Brainwave란? - Azure Machine Learning service ...
https://docs.microsoft.com/ko-kr/azure/machine.../concept-accelerate-with-fpgas ▾
2018. 10. 23. - Azure에서 FPGA를 사용하여 모델 및 심층 신경망을 가속화하는 방법을 알아봅니다. 이 문서에서는 FPGA(Field-programmable Gate Arrays)를 ...

Background (배경): 가속기

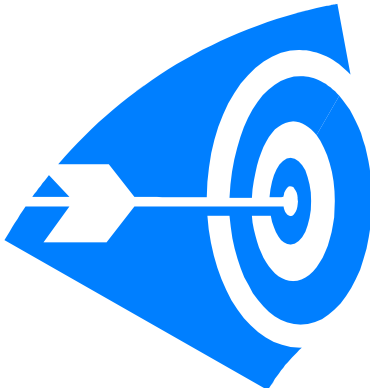
https://www.youtube.com/watch?v=kgZERPLSbmE

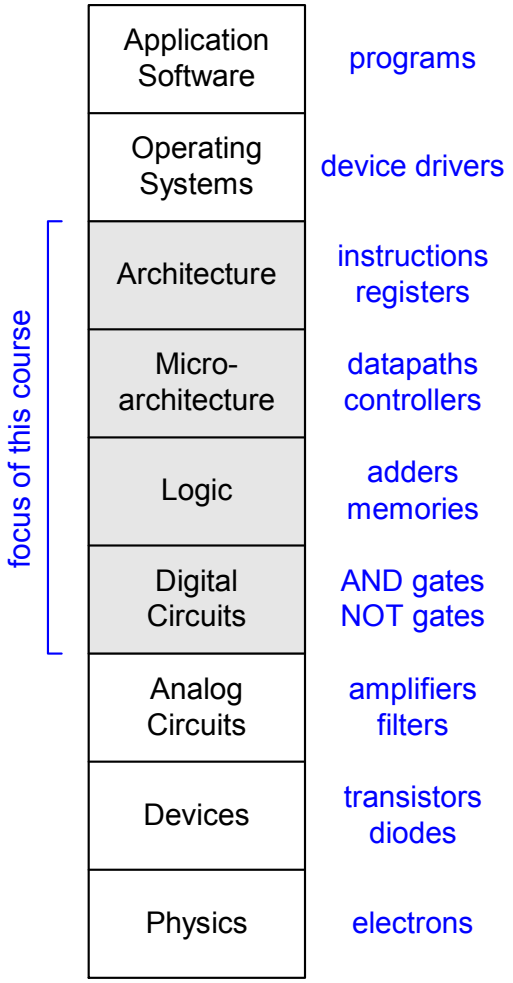
The Game Plan (계획? / 수업의 목적)

- The purpose of this course is that you:
  - Learn what's under the hood of a computer
    컴퓨터 내부에 대해서 배운다!
  - **Learn the principles of digital design**
    디지털 설계에 대한 원칙에 대해서 배운다!
  - Design and build a microprocessor
    마이크로프로세서를 설계 및 구축한다!

Abstraction

- Hiding details when they aren't important





Discipline

- Intentionally restricting your design choices (의도적으로 설계 선택사항을 제약하기)
  - to work more productively at a higher level of abstraction

- Example: Digital discipline
  - Considering **discrete voltages** instead of continuous voltages used by analog circuits
  - Digital circuits are simpler to design than analog circuits – can build more sophisticated systems
  - Digital systems replacing analog predecessors:
    - I.e., digital cameras, digital television, cell phones, CDs

The Digital Abstraction

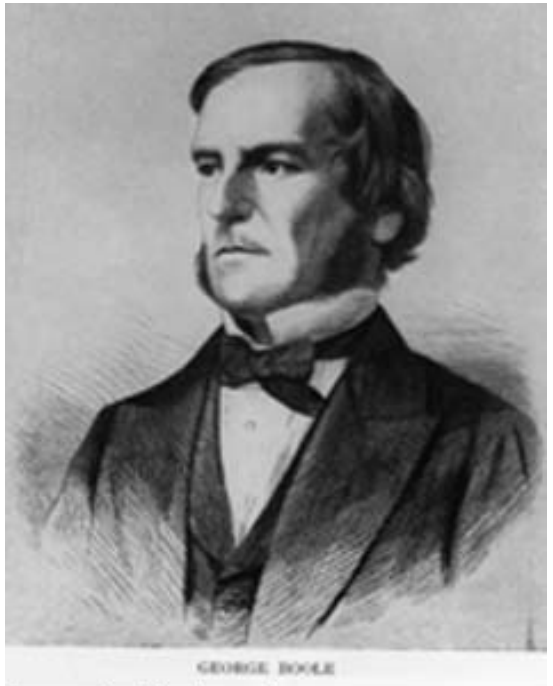
- Most physical variables are continuous, for example
  - Voltage on a wire (와이어 상의 전압)
  - Frequency of an oscillation (발진의 주파수)
  - Position of a mass (질량의 위치)
- Instead of considering all values, the digital abstraction considers only a discrete subset of values

Digital Discipline: Binary (이진!)

- Typically consider only **two** discrete values:
  - 1's and 0's
  - **1, TRUE, HIGH**
  - **0, FALSE, LOW**
- 1 and 0 can be represented by specific voltage levels, rotating gears, fluid levels, etc.
- Digital circuits usually depend on specific voltage levels to represent 1 and 0
- *Bit*: *B*inary dig*it*

George Boole, 1815 - 1864

- Born to working class parents
- Taught himself mathematics and joined the faculty of Queen's College in Ireland.
- Wrote *An Investigation of the Laws of Thought* (1854)
- Introduced **binary variables**
- Introduced the three fundamental logic operations: AND, OR, and NOT.



Scanned at the American Institute of Physics

Number Systems (수의 체계)

- Decimal numbers (십진수)

1's column
10's column
100's column
1000's column

$$5374_{10} = 5 \times 10^3 + 3 \times 10^2 + 7 \times 10^1 + 4 \times 10^0$$

five thousands | three hundreds | seven tens | four ones

- Binary numbers (이진수)

1's column
2's column
4's column
8's column

$$1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13_{10}$$

one eight | one four | no two | one one

Number Conversion (수의 변환)

- Decimal to binary conversion:
  - Convert 10101_2 to decimal

- Decimal to binary conversion:
  - Convert 47_{10} to binary

Binary Values (이진값) and Range (범위)

- Consider an *N*-digit decimal number
  - Represents 10^N possible values
  - Range is: [0, 10^N - 1]
  - For example, a 3-digit decimal number represents 10^3 = 1000 possible values, with a range of [0, 999]
- Consider an *N*-bit binary number
  - Represents 2^N possible values
  - Range is: [0, 2^N - 1]
  - For example, a 3-digit binary number represents $2^3 = 8$ possible values, with a range of [0, 7] (i.e., 000_2 to 111_2)

Hexadecimal Numbers (16진수)

Hex Digit	Decimal Equivalent	Binary Equivalent
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

Hexadecimal Numbers

- Base 16
- Shorthand for Binary

Hexadecimal to Binary Conversion

- Hexadecimal to binary conversion:
  - Convert **$4AF_{16}$** (also written 0x4AF) to binary

- Hexadecimal to decimal conversion:
  - Convert **0x4AF** to decimal

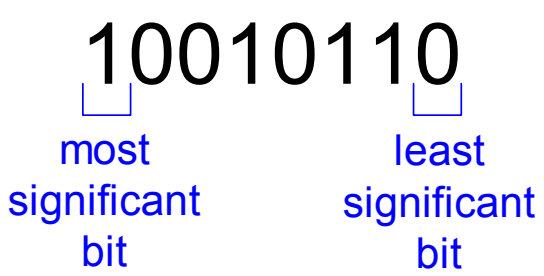
Hexadecimal to Binary Conversion (16진수의 이진수 변환)

- **Hexadecimal to binary conversion:**
  - Convert $4AF_{16}$ (also written 0x4AF) to binary
  - **010010101111_2**

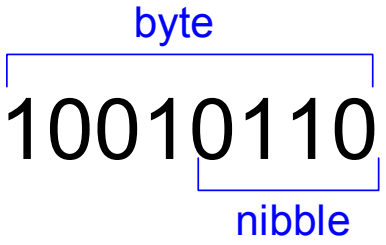
- **Hexadecimal to decimal conversion:**
  - Convert 0x4AF to decimal
  - 010010101111_2 $= 1 + 2 + 4 + 8 + 32 + 128 + 1024$
    $= \mathbf{1199_{10}}$
  - 0x4AF $= (15 \times 16^0) + (10 \times 16^1) + (4 \times 16^2)$
    $= \mathbf{1199_{10}}$

Bits, Bytes, Nibbles…

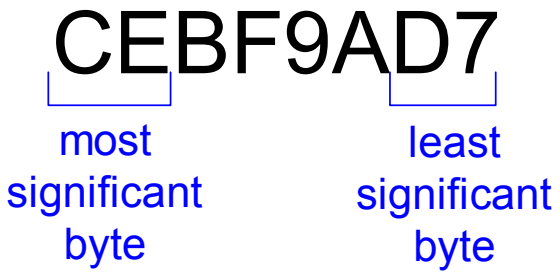
- Bits



- Bytes & Nibbles



- Bytes



Powers of Two (2의 짜승수)

- $2^{10} = 1$ kilo ≈ 1000 (1024)
- $2^{20} = 1$ mega ≈ 1 million (1,048,576)
- $2^{30} = 1$ giga ≈ 1 billion (1,073,741,824)

Estimating Powers of Two

- What is the value of 2^{22}?
  - $2^2 \times 2^{20} = 4$ Mega
- How many values can a 32-bit variable represent?
  - $2^2 \times 2^{30} = 4$ Giga

Addition (더하기)

- Decimal

```
   11  ← carries
    3734
+   5168
-------
    8902
```

- Binary

```
   11  ← carries
    1011
+   0011
-------
    1110
```

Binary Addition Examples

- Add the following 4-bit binary numbers

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

- Add the following 4-bit binary numbers

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

Overflow (오버플로우, 넘침)

- Digital systems operate on a fixed number of bits
- Addition overflows when the result is too big to fit in the available number of bits
- Example: add 13 and 5 using 4-bit numbers

```
  11  1
   1101
+  0101
------
  10010
```

Signed Binary Numbers
(부호를 가진 수의 표현)

- Sign/Magnitude Numbers (부호/절대값)
- Two's Complement Numbers (2의 보수)

Sign/Magnitude Numbers

- 1 sign bit, N-1 magnitude bits
- Sign bit is the most significant (left-most) bit
- Negative number: sign bit = 1
- Positive number: sign bit = 0
- Example, 4-bit representations of ± 5:

  $-5 = 1101_2$

  $+5 = 0101_2$

- Range of an *N*-bit sign/magnitude number:

  $[-(2^{N-1}-1), 2^{N-1}-1]$

Sign/Magnitude Numbers

- Problems:
  - Addition **doesn't** work, for example -5 + 5:

$$\begin{array}{r} 1101 \\ +\ 0101 \\ \hline 10010 \end{array}$$

  - Two representations of 0 (± 0):

1000

0000

Two's Complement Numbers

- Don't have same problems as sign/magnitude numbers:
  - Addition works
  - Single representation for 0

Two's Complement Numbers

- Same as unsigned binary, but the most significant bit (msb) has value of -2^{N-1}
- Most positive 4-bit number: 0111_2 (7_{10})
- Most negative 4-bit number: 1000_2 ($-2^3 = -8_{10}$)
- The most significant bit still indicates the sign (1 = negative, 0 = positive)
- Range of an *N*-bit two's complement number: $[-2^{N-1}, 2^{N-1}-1]$

"Taking the Two's Complement"

- Reversing the sign of a two's complement number (2의 보수 표현에서 음수화 작업)
- **Method:**
  1. Invert the bits
  2. Add 1
- Example: Reverse the sign of 0111_2
  1. 1000
  2. $\begin{array}{r} +\quad 1 \\ \hline 1001 \end{array}$

Two's Complement Examples

- Take the two's complement of 0110_2
  1. 1001
  2. + 1

     1010

- Take the two's complement of 1101_2
  1. 0010
  2. + 1

     0011

Two's Complement Addition

- Add 6 + (-6) using two's complement numbers

```
  111
   0110
+  1010
------
  10000
```

- Add -2 + 3 using two's complement numbers

```
  111
   1110
+  0011
------
  10001
```

Increasing Bit Width

- A value can be extended from N bits to M bits (where $M > N$) by using:
  - Sign-extension
  - Zero-extension

Sign-Extension (부호 확장)

- **Sign bit is copied into most significant bits**.
- Number value remains the same.

- **Example 1:**
  - 4-bit representation of 3 = 0011
  - 8-bit sign-extended value: 00000011
- **Example 2:**
  - 4-bit representation of -5 = 1011
  - 8-bit sign-extended value: 11111011

Zero-Extension

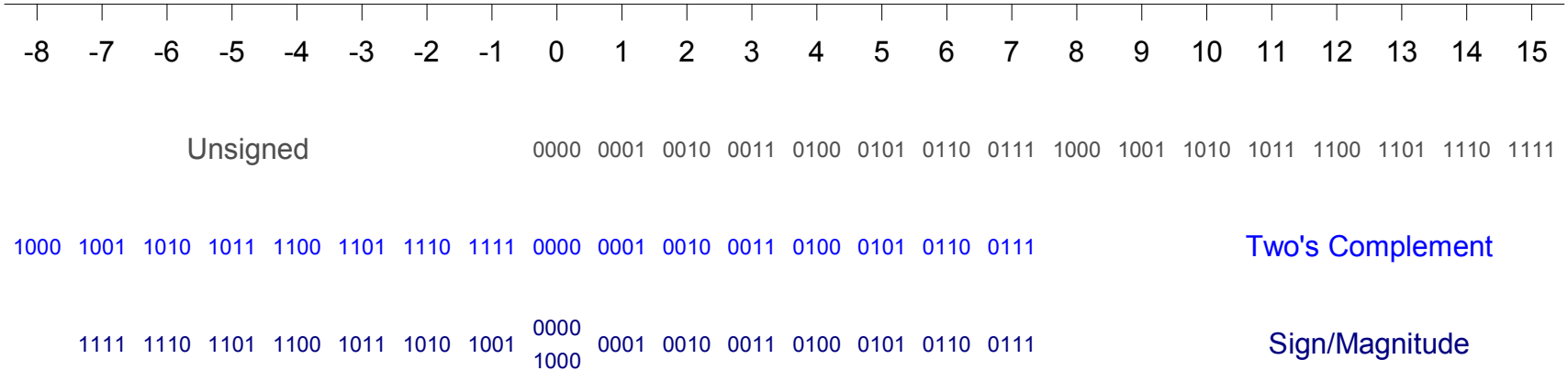
- Zeros are copied into most significant bits.
- Number value may change.

- **Example 1:**
  - 4-bit value = 0011
  - 8-bit zero-extended value: 00000011
- **Example 2:**
  - 4-bit value = 1011
  - 8-bit zero-extended value: 00001011

Number System Comparison

Number System	Range
Unsigned	$[0, 2^N-1]$
Sign/Magnitude	$[-(2^{N-1}-1), 2^{N-1}-1]$
Two's Complement	$[-2^{N-1}, 2^{N-1}-1]$

For example, 4-bit representation:

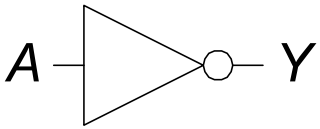


Logic Gates

- Perform logic functions: inversion (NOT), AND, OR, NAND, NOR, etc.
- Single-input:
  - NOT gate, buffer
- Two-input:
  - AND, OR, XOR, NAND, NOR, XNOR
- Multiple-input

Single-Input Logic Gates

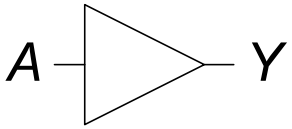
NOT



$Y = \overline{A}$

A	Y
0	1
1	0

BUF

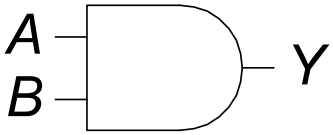


$Y = A$

A	Y
0	0
1	1

Two-Input Logic Gates

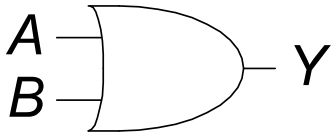
AND



$Y = AB$

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

OR

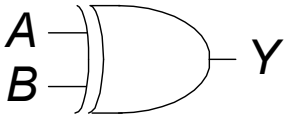


$Y = A + B$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

More Two-Input Logic Gates

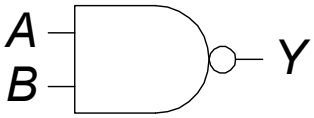
XOR



$Y = A \oplus B$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

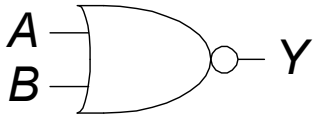
NAND



$Y = \overline{AB}$

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

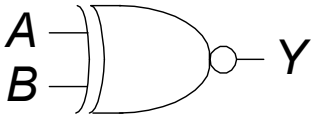
NOR



$Y = \overline{A + B}$

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

XNOR

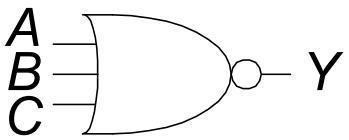


$Y = \overline{A \oplus B}$

A	B	Y
0	0	
0	1	
1	0	
1	1	

Multiple-Input Logic Gates

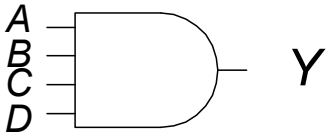
NOR3



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

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	

AND4



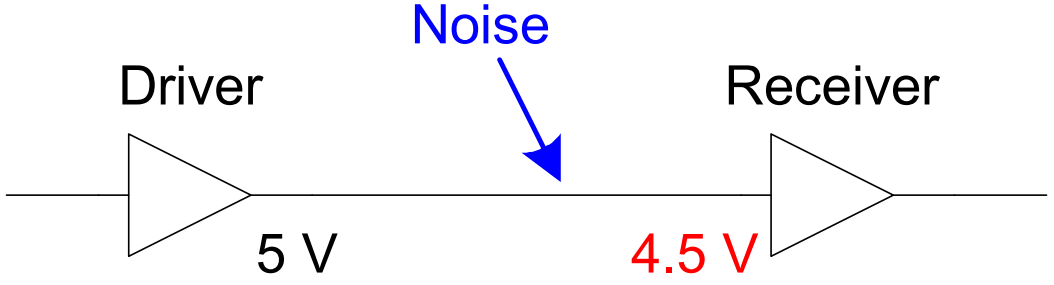
$$Y = ABCD$$

Logic Levels

- Define discrete voltages to represent 1 and 0
- For example, we could define:
  - 0 to be *ground* or 0 volts
  - 1 to be V_{DD} or 5 volts
- But what if our gate produces, for example, 4.99 volts? Is that a 0 or a 1?
- What about 3.2 volts?

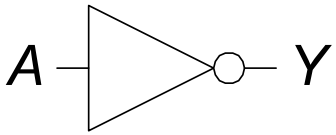
Logic Levels

- Define a ***range*** of voltages to represent 1 and 0
- Define different ranges for outputs and inputs to allow for *noise* in the system
- Noise is anything that degrades the signal
- For example, a gate (driver) could output a 5 volt signal but, because of losses in the wire and other noise, the signal could arrive at the receiver with a degraded value, for example, 4.5 volts



How do We Build Logic Gates?

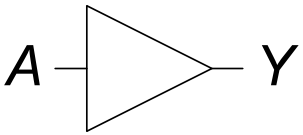
NOT



$Y = \overline{A}$

A	Y
0	1
1	0

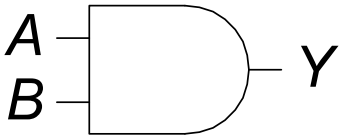
BUF



$Y = A$

A	Y
0	0
1	1

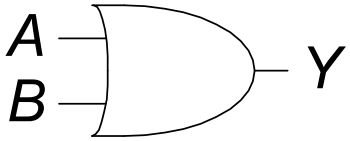
AND



$Y = AB$

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

OR



$Y = A + B$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

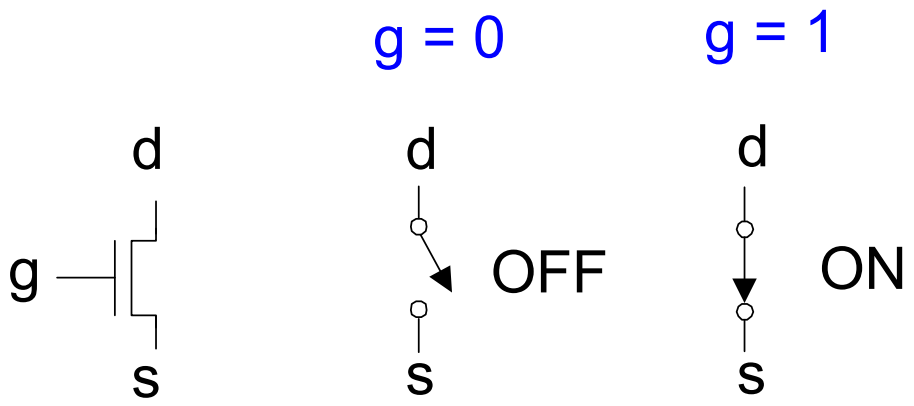
Transistors!

Robert Noyce, 1927 - 1990

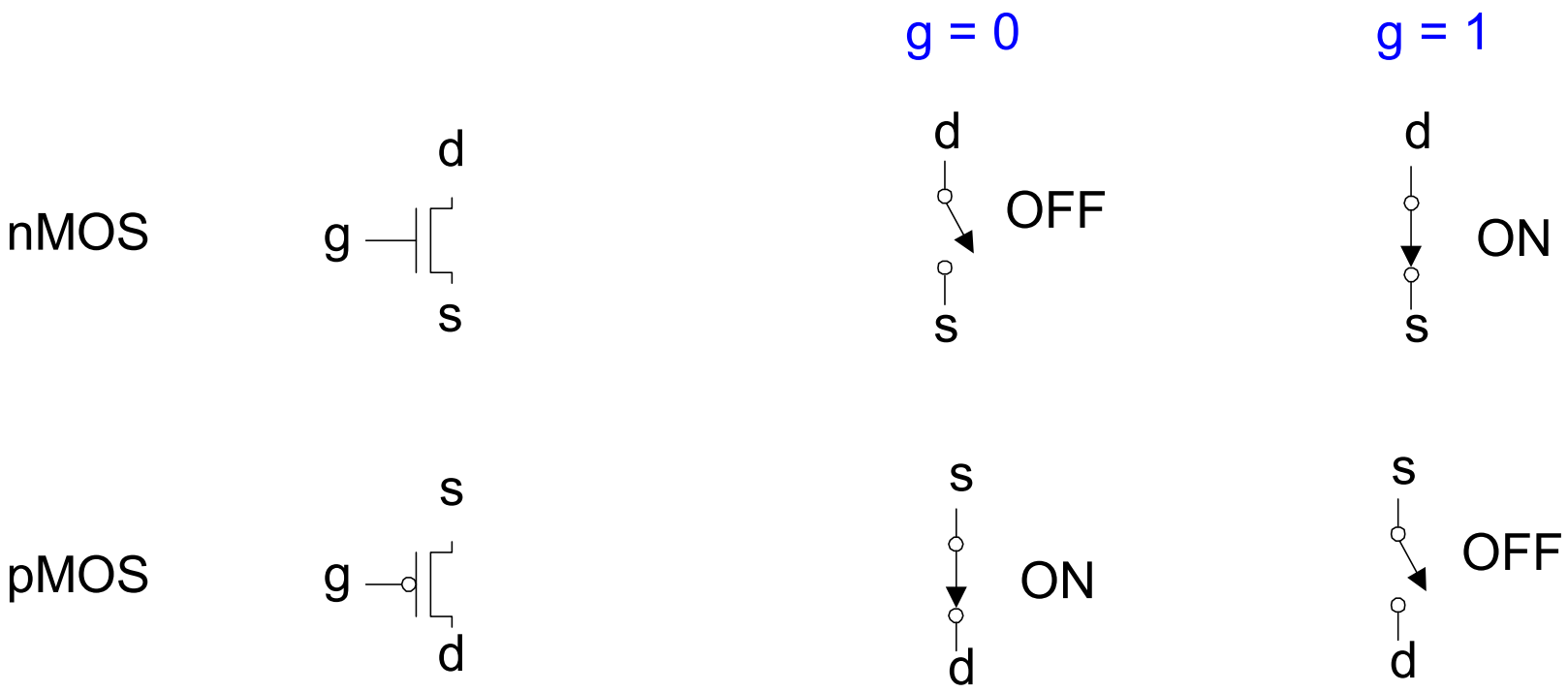
- Nicknamed “Mayor of Silicon Valley”
- Cofounded Fairchild Semiconductor in 1957
- Cofounded Intel in 1968
- Co-invented the integrated circuit

Transistor Basics (트랜지스터의 기초)

- Transistor is a three-ported voltage-controlled switch
  - Two of the ports are connected depending on the voltage on the third port
  - For example, in the switch below the two terminals (d and s) are connected (ON) only when the third terminal (g) is 1

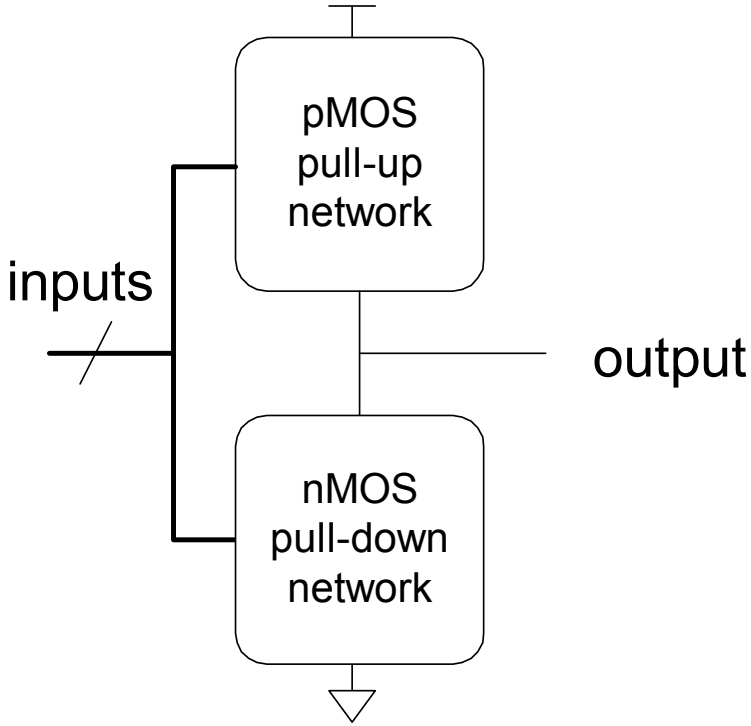


Transistor Function (트랜지스터의 기능)



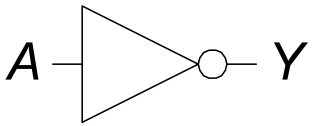
Transistor Function (트랜지스터의 기능)

- nMOS transistors pass good 0's, so connect source to GND
- pMOS transistors pass good 1's, so connect source to V_{DD}



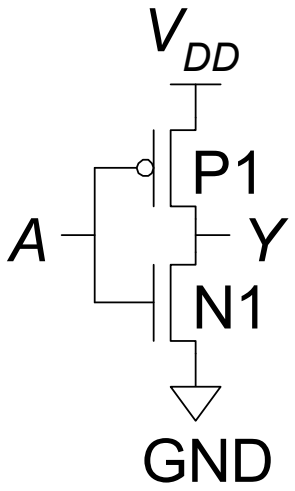
CMOS Gates: NOT Gate

NOT



$Y = \overline{A}$

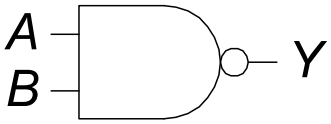
A	Y
0	1
1	0



A	P1	N1	*Y*
0	ON	OFF	1
1	OFF	ON	0

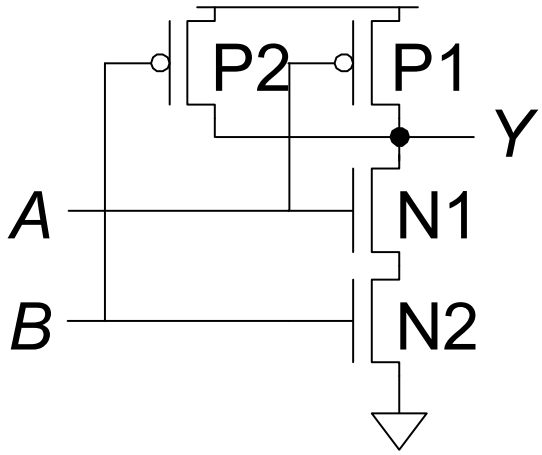
CMOS Gates: NAND Gate

NAND



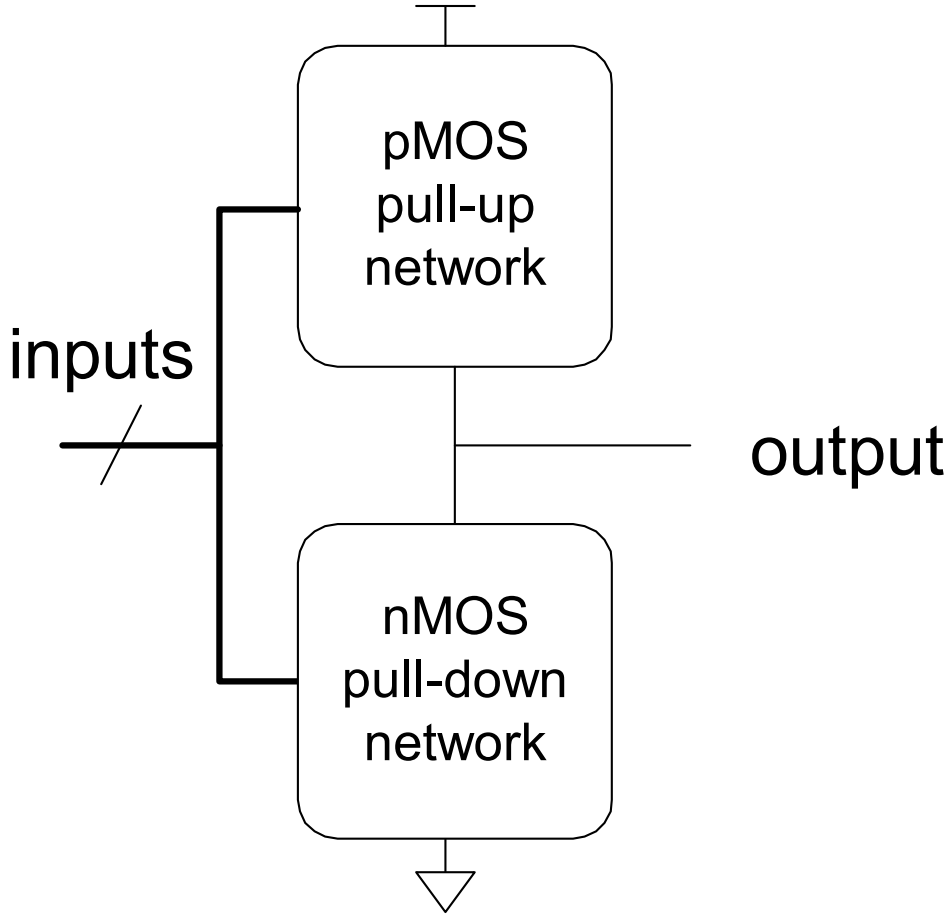
$Y = \overline{AB}$

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0



A	*B*	P1	P2	N1	N2	*Y*
0	0	ON	ON	OFF	OFF	1
0	1	ON	OFF	OFF	ON	1
1	0	OFF	ON	ON	OFF	1
1	1	OFF	OFF	ON	ON	0

CMOS Gate Structure

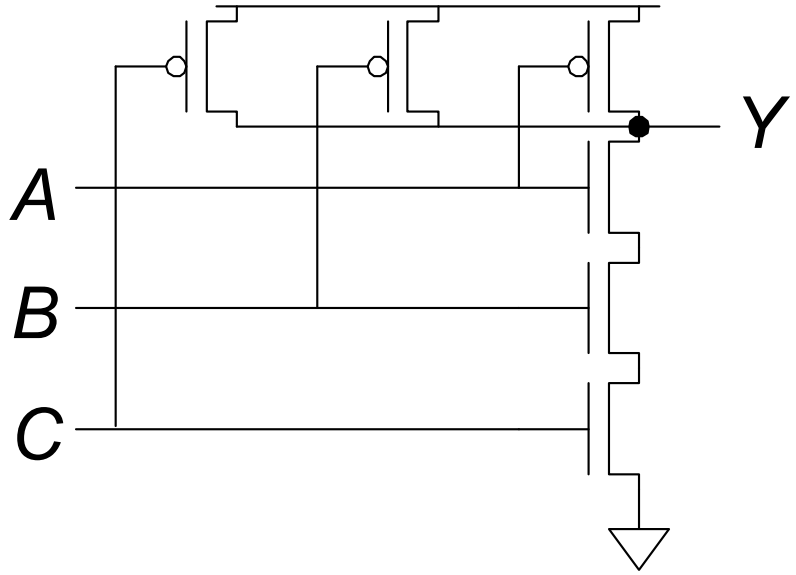


NOR Gate

How do you build a three-input NAND gate?

NOR Gate

Three-input NAND gate

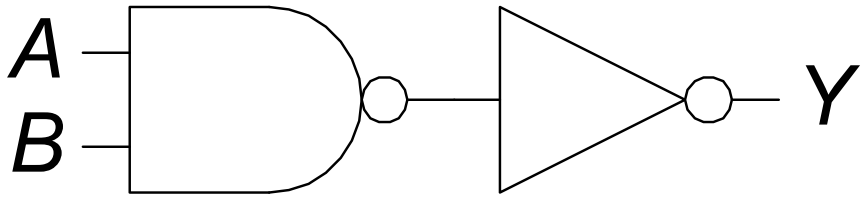


Other CMOS Gates

How do you build a two-input AND gate?

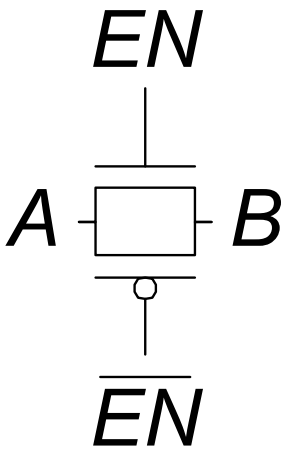
Other CMOS Gates

Two-input AND gate



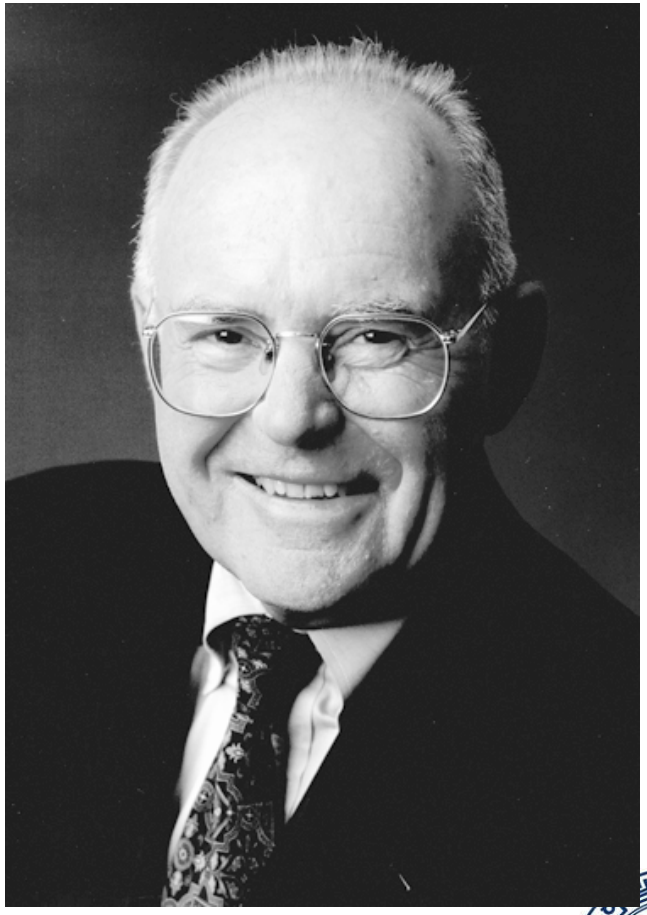
Transmission Gates (전달 게이트)

- A transmission gate is an ideal switch: it passes a good 1 and a good 0
- When $EN = 1$, the switch is ON:
  - $\overline{EN} = 0$ and A is connected to B
- When $EN = 0$, the switch is OFF:
  - A is not connected to B



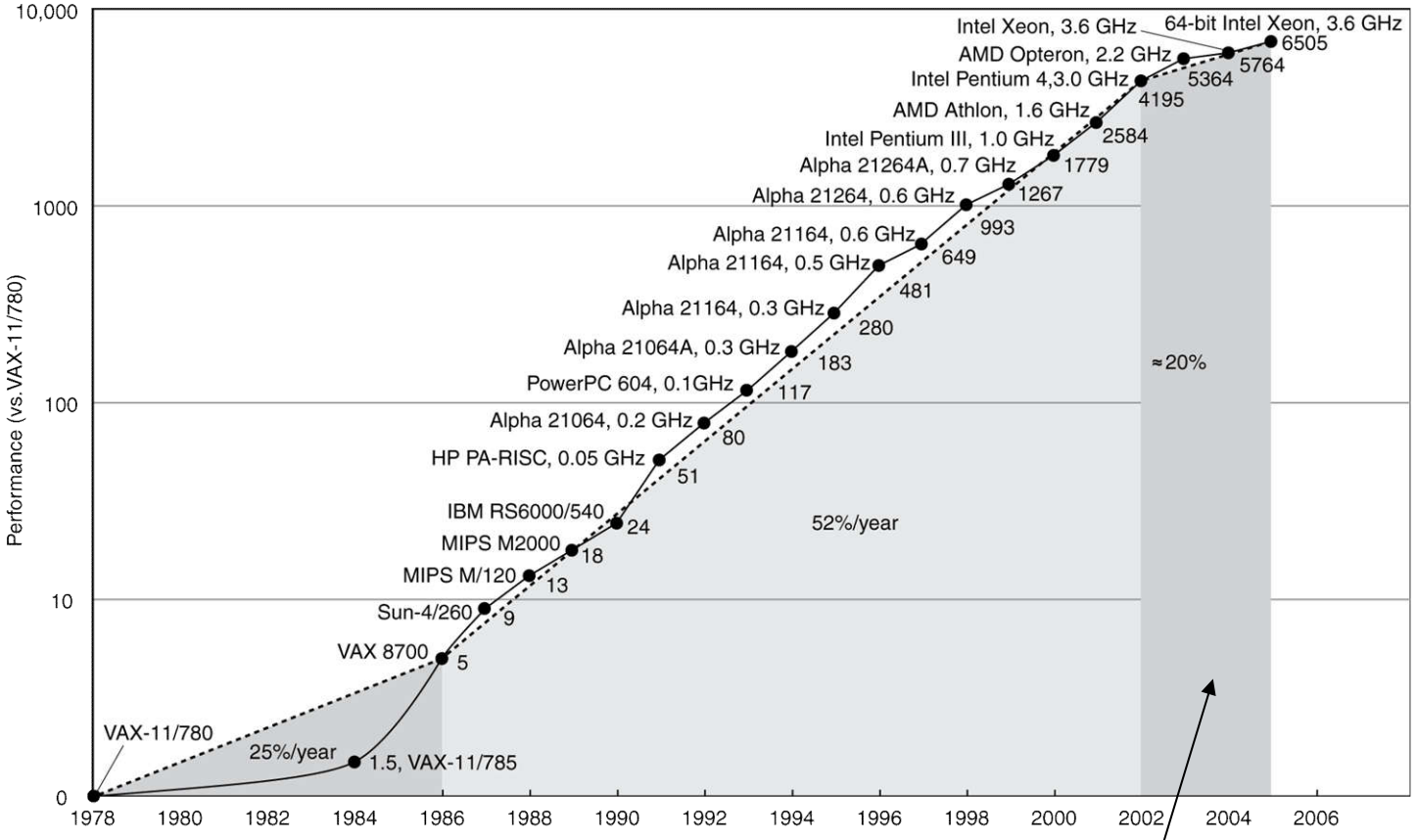
Gordon Moore (고든 무어), 1929 -

- Cofounded Intel in 1968 with Robert Noyce.
- **Moore's Law:** the number of transistors on a computer chip doubles every year (observed in 1965)
- Since 1975, transistor counts have doubled every two years.



http://en.wikipedia.org/wiki/Transistor_count

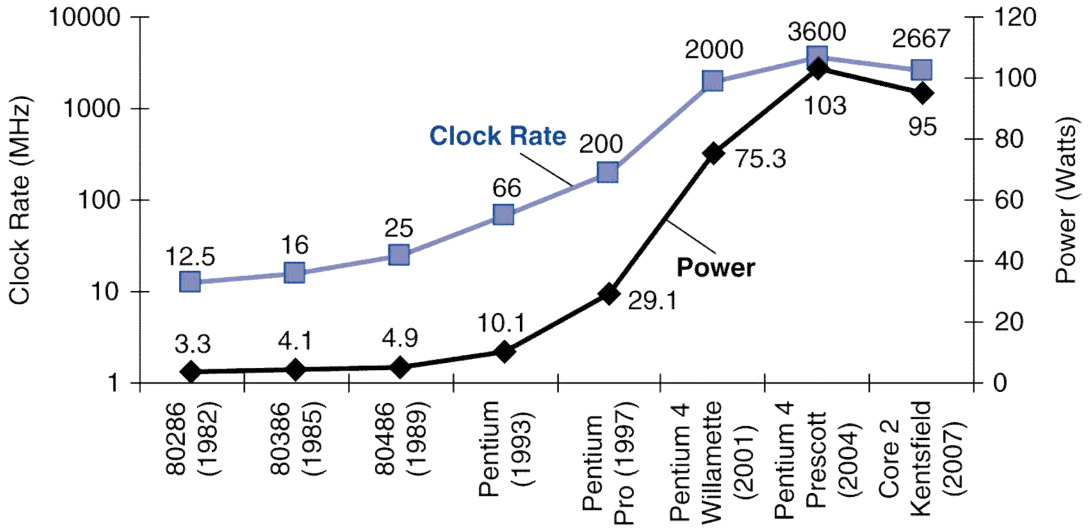
Uniprocessor Performance



Constrained by power, instruction-level parallelism, memory latency

Power Consumption (전력 소모)

- Energy consumption per unit time
- Two types of power consumption:
  - Dynamic power consumption (동적 전력 소모)
  - Static power consumption (정적 전력 소모)



Dynamic Power Consumption

- Power to charge transistor gate capacitances
- The energy required to charge a capacitance, C, to V_{DD} is CV_{DD}^2
- If the circuit is running at frequency f, and all transistors switch (from 1 to 0 or vice versa) at that frequency, the capacitor is charged $f/2$ times per second (discharging from 1 to 0 is free).
- Thus, the total dynamic power consumption is:

$$P_{dynamic} = \frac{1}{2} C V_{DD}^2 f$$

Static Power Consumption

- Power consumed when no gates are switching
- It is caused by the *quiescent supply current*, I_{DD}, also called the *leakage current*
- Thus, the total static power consumption is:

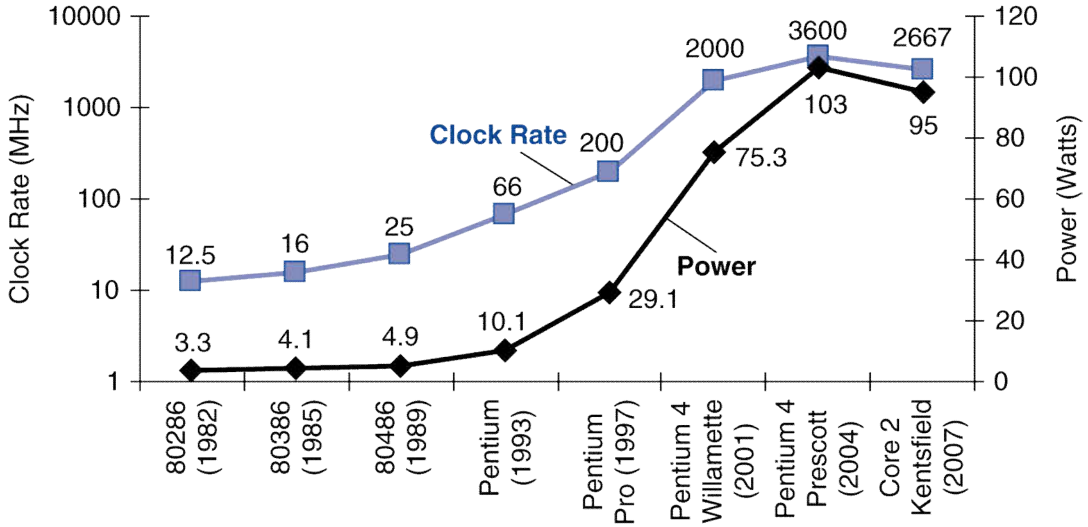
$$P_{static} = I_{DD} V_{DD}$$

Power Consumption Example

- $V_{DD} = 1.2$ V
- $C = 20$ nF
- $f = 1$ GHz
- $I_{DD} = 20$ mA

$$P = \tfrac{1}{2}CV_{DD}^2 f + I_{DD}V_{DD}$$
$$= \tfrac{1}{2}(20\text{ nF})(1.2\text{ V})^2(1\text{ GHz}) + (20\text{ mA})(1.2\text{ V})$$
$$= 14.4\text{ W}$$

Power Trends



- In CMOS IC technology

$$\text{Power} = \text{Capacitive load} \times \text{Voltage}^2 \times \text{Frequency}$$

$\times 30$ | $5\text{V} \rightarrow 1\text{V}$ | $\times 1000$

Reducing Power

- Suppose a new CPU has
  - 85% of capacitive load of old CPU
  - 15% voltage and 15% frequency reduction

$$\frac{P_{new}}{P_{old}} = \frac{C_{old} \times 0.85 \times (V_{old} \times 0.85)^2 \times F_{old} \times 0.85}{C_{old} \times V_{old}^{\ 2} \times F_{old}} = 0.85^4 = 0.52$$

- The power wall (전력 소모의 벽)
  - We can't reduce voltage further
  - We can't remove more heat
- How else can we improve performance?

Multiprocessors

- Multicore microprocessors (멀티코어)
  - More than one processor per chip

- Requires **explicitly** parallel programming
  - Compare with **instruction level parallelism**
    - Hardware executes multiple instructions at once
    - Hidden from the programmer
  - Hard to do
    - Programming for performance
    - Load balancing
    - Optimizing communication and synchronization