RUTGERS
THE STATE UNIVERSITY
OF NEW JERSEY

ECE-332:437
DIGITAL SYSTEMS DESIGN (DSD)

Fall 2016 – Lecture 1

Nagi Naganathan
September 8, 2016

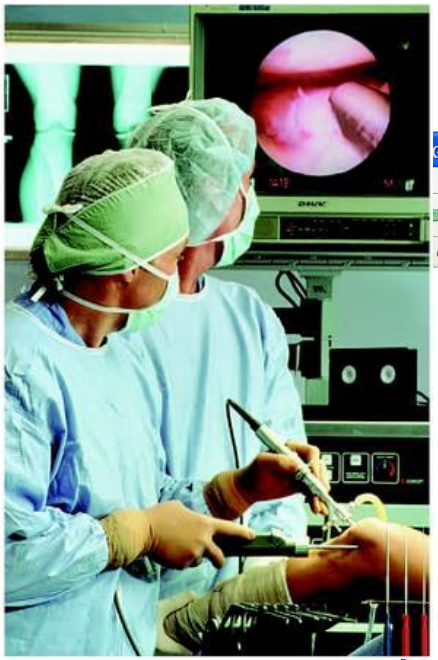
Chapter 1 :: Topics

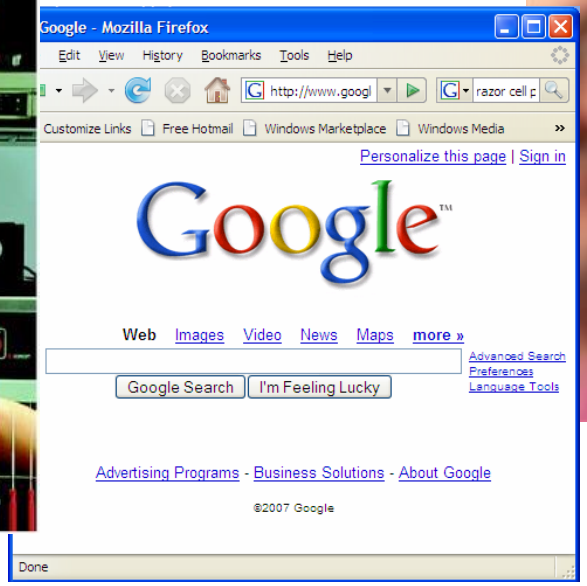
- **Background**
- **The Game Plan**
- **The Art of Managing Complexity**
- **The Digital Abstraction**
- **Number Systems**
- **Logic Gates**
- **Logic Levels**
- **CMOS Transistors**
- **Power Consumption**

FROM ZERO TO ONE

Background

- Microprocessors have revolutionized our world
  - Cell phones, Internet, rapid advances in medicine, etc.
- The semiconductor industry has grown from $21 billion in 1985 to over $300 billion in 2013





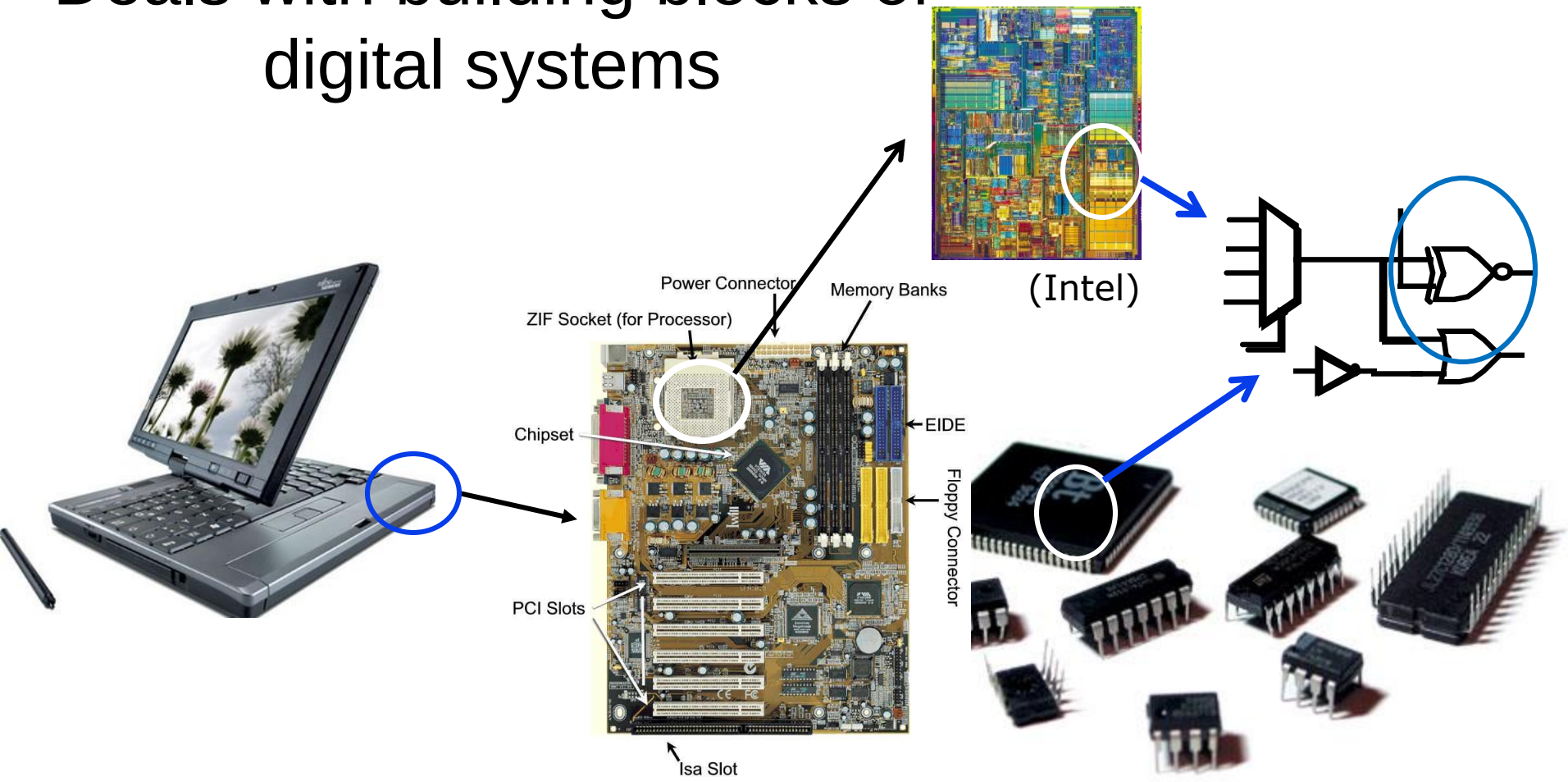
The Game Plan

- Purpose of course:
  - Understand what's under the hood of a computer
  - Learn the principles of digital design
  - Learn to systematically debug increasingly complex designs
  - Design and build a microprocessor

FROM ZERO TO ONE

Digital Logic Design

Deals with building blocks of digital systems



Topics to cover today – September 8, 2016

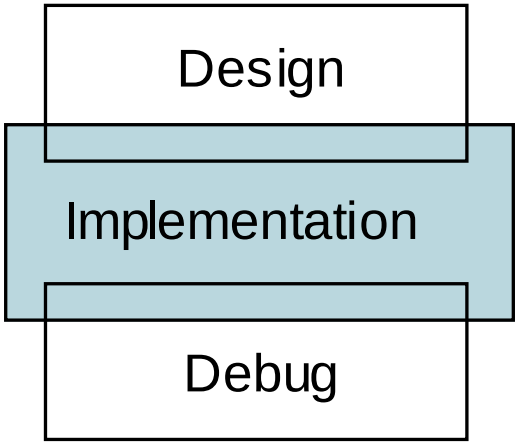
- Lecture 1
  - Historical Perspectives
  - Basic Hardware Concepts
  - Integrated Circuits
  - Digital Devices
  - Analog vs Digital
  - Basic Boolean Gates
- Lecture 2
  - Number systems
  - Representation of negative numbers
  - 2’s complement
  - Binary addition, subtraction

Part 1
Historical Perspectives

General Principles

- Technology changes fast
  - State-of-the-art this year becomes obsolete next week
- Understand general principles
  - Apply Problem Solving Principles
  - Optimization, Tradeoffs
- Concepts remain the same:
  - Example: relays -> tubes -> bipolar transistors -> MOS transistors

The Process of Design



Design

Initial concept: what is the function performed by the object?
Constraints: How fast? How much area? How much cost?
Refine abstract functional blocks into more concrete realizations

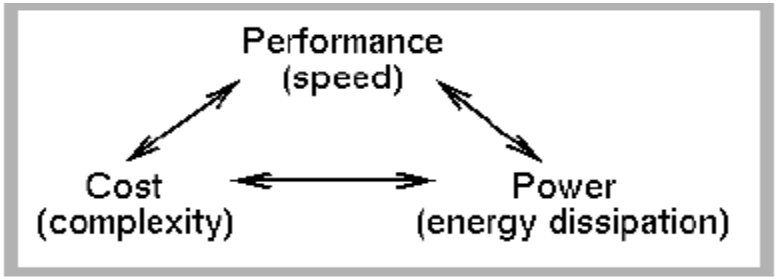
Implementation

Assemble primitives into more complex building blocks
Composition via wiring
Choose among alternatives to improve the design

Debug

Faulty systems: design flaws, composition flaws, component flaws
Design to make debugging easier
Hypothesis formation and troubleshooting skills

Design Tradeoffs



- You can improve on one at the expense of worsening one or both of the others.
- These tradeoffs exist at every level in the system design - every sub-piece and component.
- Design Specification -
  - Functional Description.
  - Performance, cost, power constraints.
- As a designer you must make the tradeoffs necessary to achieve the function within the constraints.

Themes in Digital Design

IMPORTANT THEMES IN DIGITAL DESIGN

- Good tools do not guarantee good design, but they help a lot by taking the pain out of doing things right.
- Digital circuits have analog characteristics.
- Know when to worry and when not to worry about the analog aspects of digital design.
- Always document your designs to make them understandable by yourself and others.
- Associate active levels with signal names and practice bubble-to-bubble logic design.
- Understand and use standard functional building blocks.
- Design for minimum cost at the system level, including your own engineering effort as part of the cost.
- State-machine design is like programming; approach it that way.
- Use programmable logic to simplify designs, reduce cost, and accommodate last-minute modifications.
- Avoid asynchronous design. Practice synchronous design until a better methodology comes along.
- Pinpoint the unavoidable asynchronous interfaces between different subsystems and the outside world, and provide reliable synchronizers.
- Catching a glitch in time saves nine.

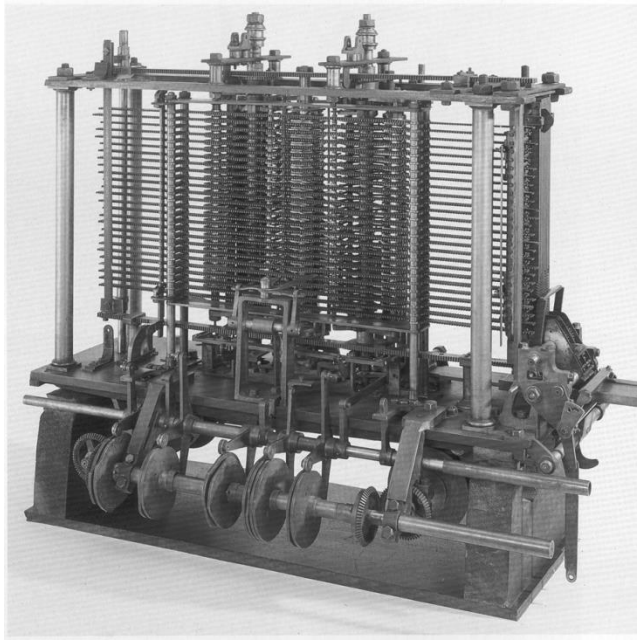
Areas for successful digital design

- Debugging
  - Trouble shooter, systematic planning and patience
- Business Requirements and Practices
  - Documentation, Feature Definitions
- Risk Taking
- Communication

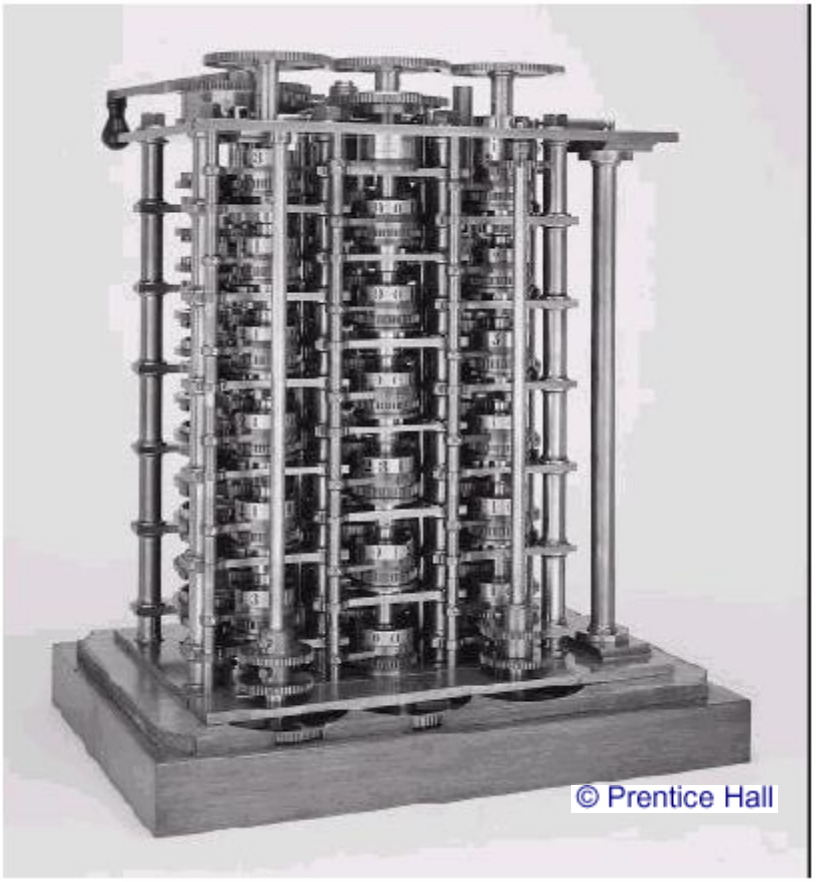
Historical Perspectives

The Analytical Engine

- Designed by Charles Babbage from 1834 – 1871
- Considered to be the first digital computer
- Built from mechanical gears, where each gear represented a discrete value (0-9)
- Babbage died before it was finished



History



The Babbage Difference Engine

- **Made in 1832**
- **25,000 Parts**
- **17,470 Pounds**

History

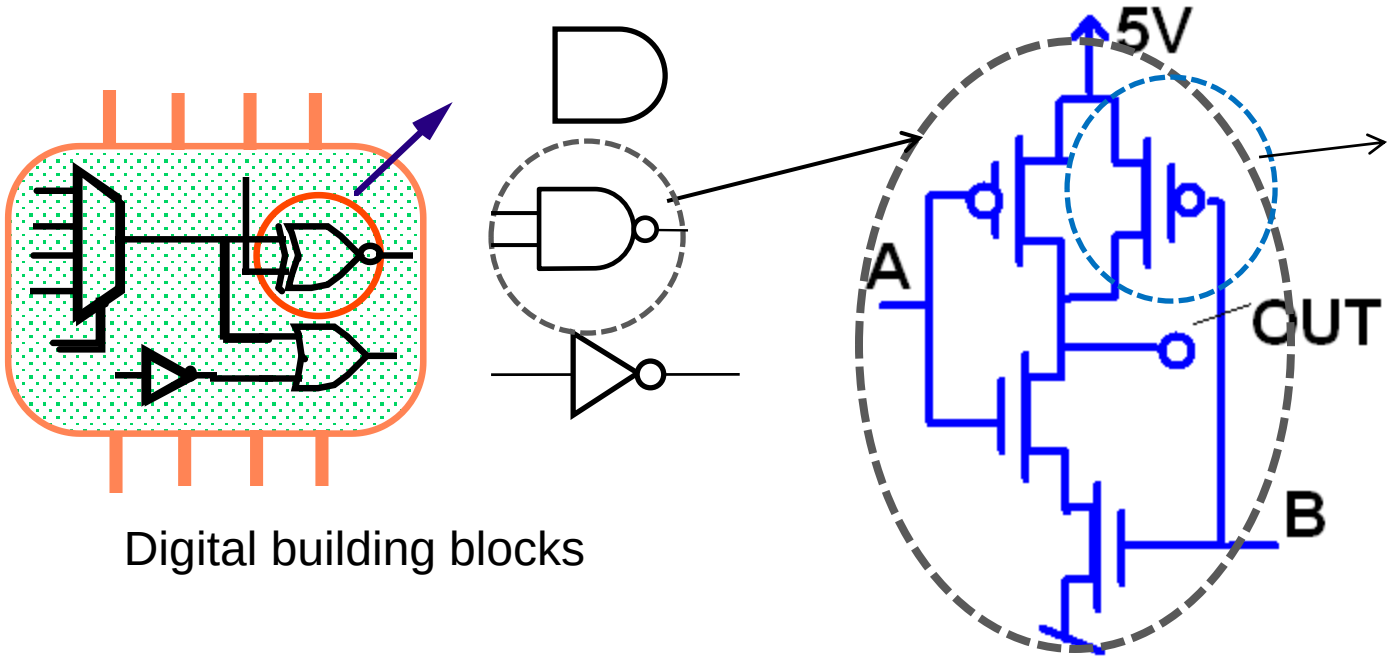
ENIAC - The first electronic Computer (1946)

(18000 vacuum tubes, 30 tons, 80'x8.5', 1900 adds/sec)

What are Logic Gates built from?

Transistors:

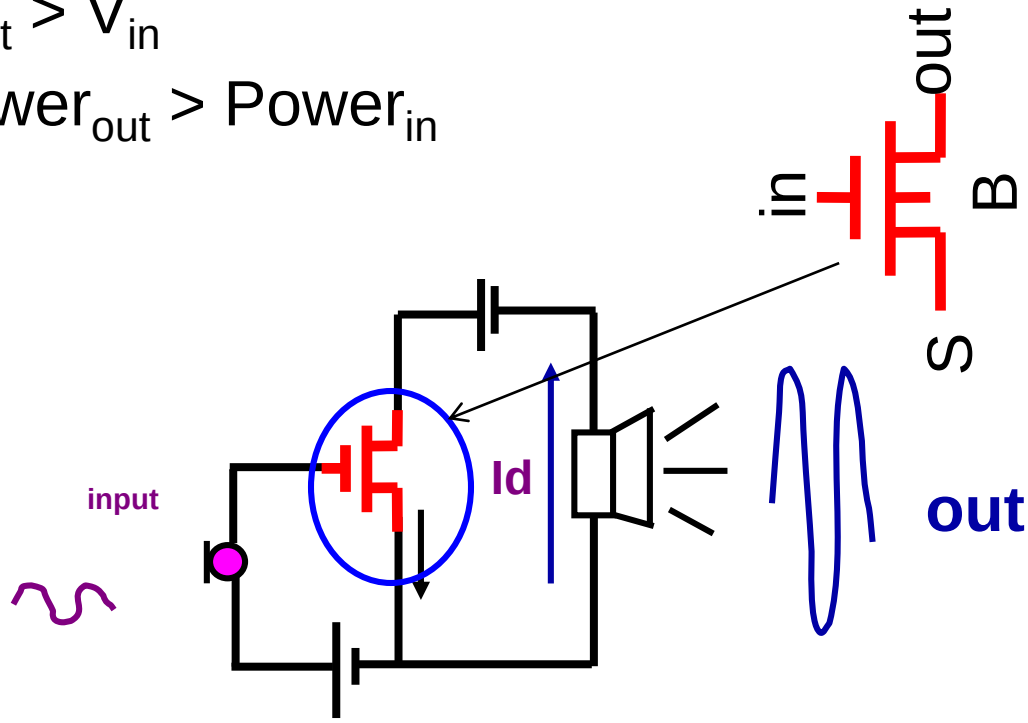
- The transistor is the workhorse of every electronic device.

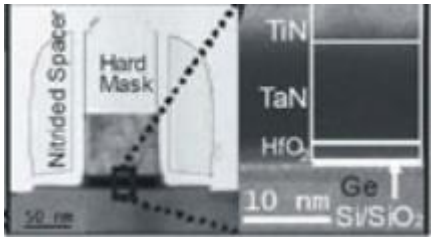


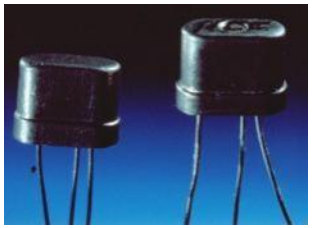
Digital building blocks

What is a Transistor?

- Electronic, solid-state device that can amplify an electric signal:
  - » $V_{out} > V_{in}$
  - » $Power_{out} > Power_{in}$

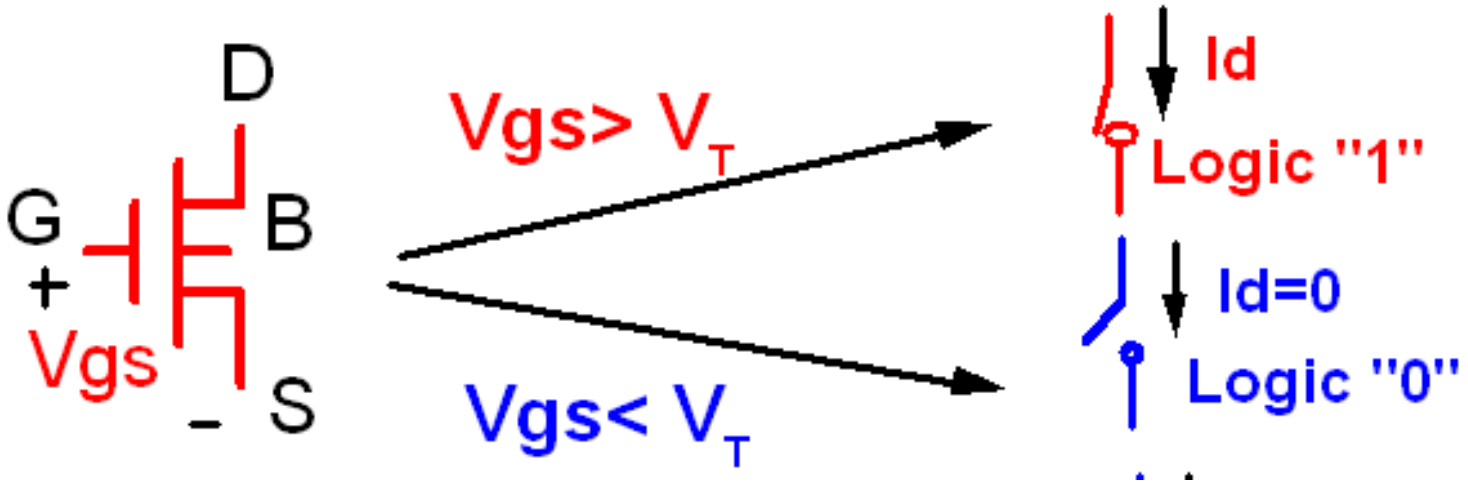






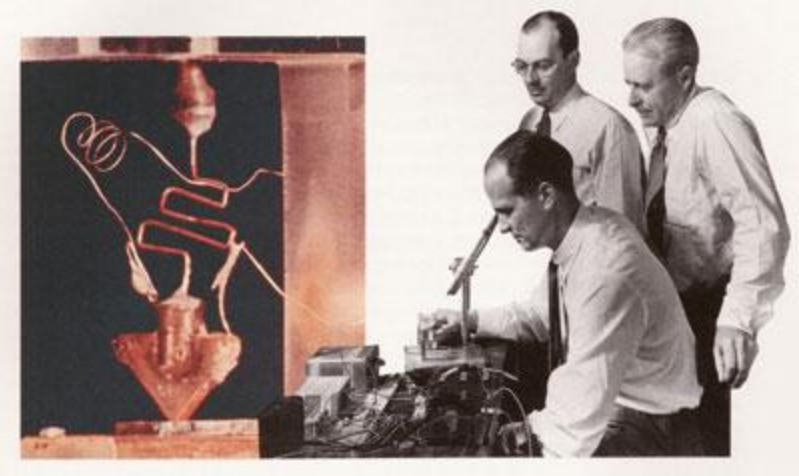
Digital Model of a Transistor

- We make abstraction of the signals: 0 or 1
- As a result a transistors can be considered a switch (on or off; 1 or ;):



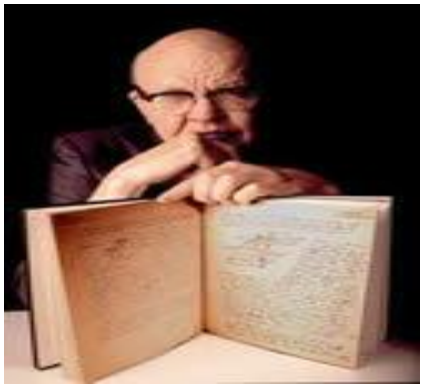
Invention of Transistor

- Bardeen, Brattain, Shockley invented in Bell Labs in 1947
- Nobel Prize



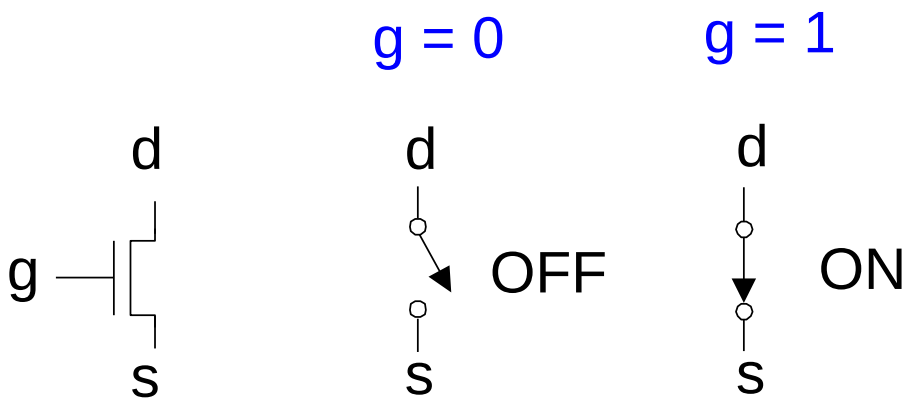
Integrated Circuit

- Jack Kilby of Texas Instruments built the first integrated circuit (IC) with 2 transistors – 1958
- Nobel Prize in Physics



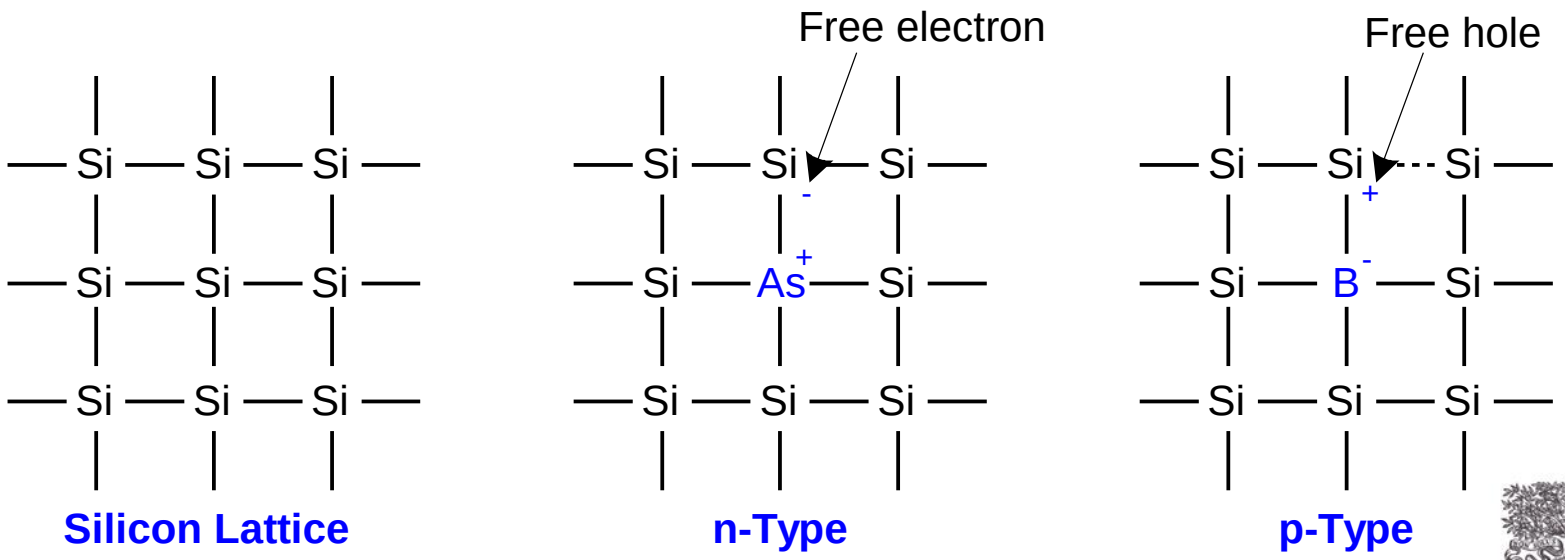
Transistors

- Logic gates built from transistors
- 3-ported voltage-controlled switch
  - 2 ports connected depending on voltage of 3rd
  - d and s are connected (ON) when g is 1



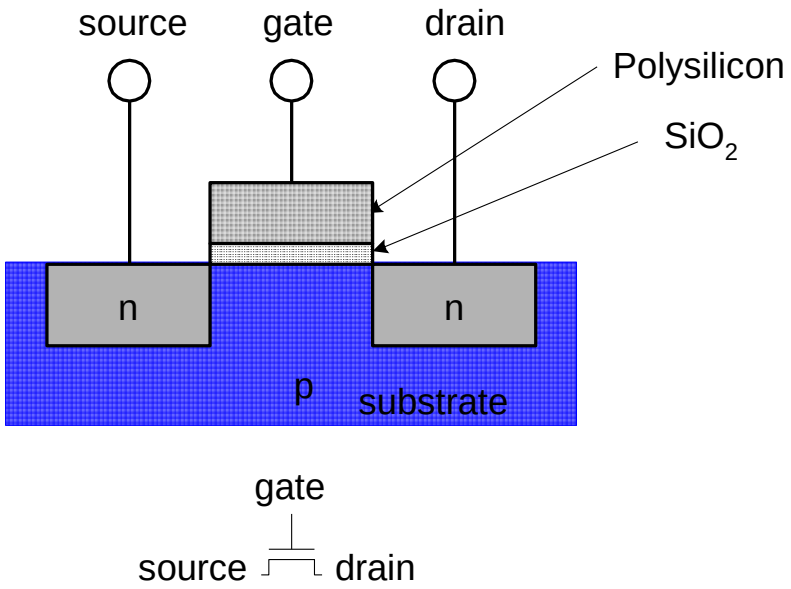
Silicon

- Transistors built from silicon, a semiconductor
- Pure silicon is a poor conductor (no free charges)
- Doped silicon is a good conductor (free charges)
  - n-type (free *n*egative charges, electrons)
  - p-type (free *p*ositive charges, holes)



MOS Transistors

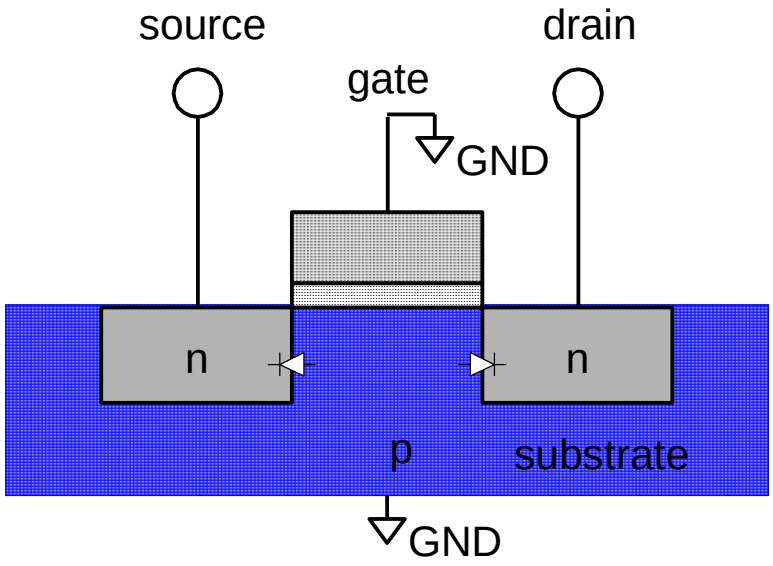
- **Metal oxide silicon (MOS) transistors:**
  - Polysilicon (used to be **metal**) gate
  - **Oxide** (silicon dioxide) insulator
  - Doped **silicon**



Transistors: nMOS

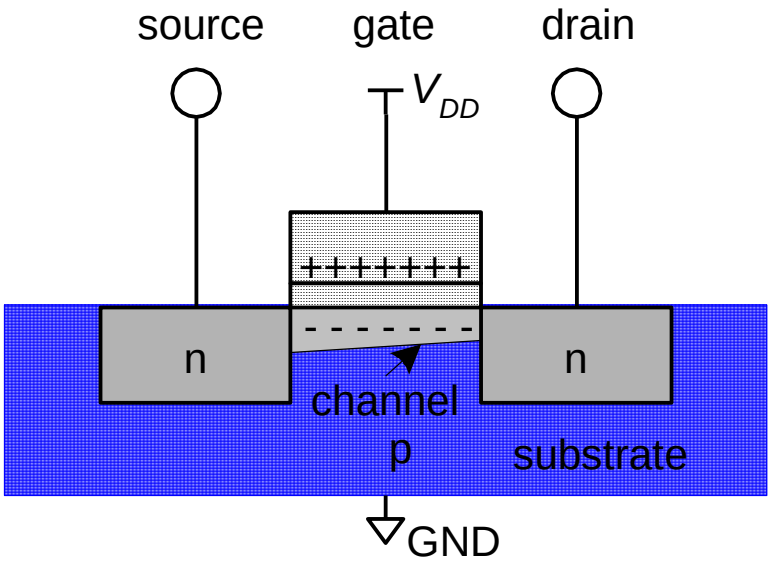
Gate = 0

OFF (no connection between source and drain)



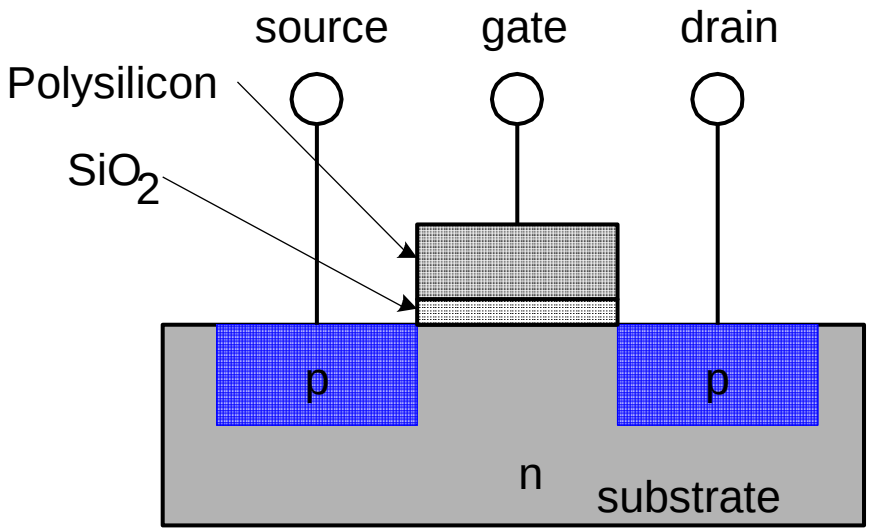
Gate = 1

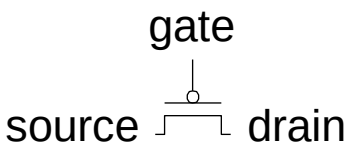
ON (channel between source and drain)



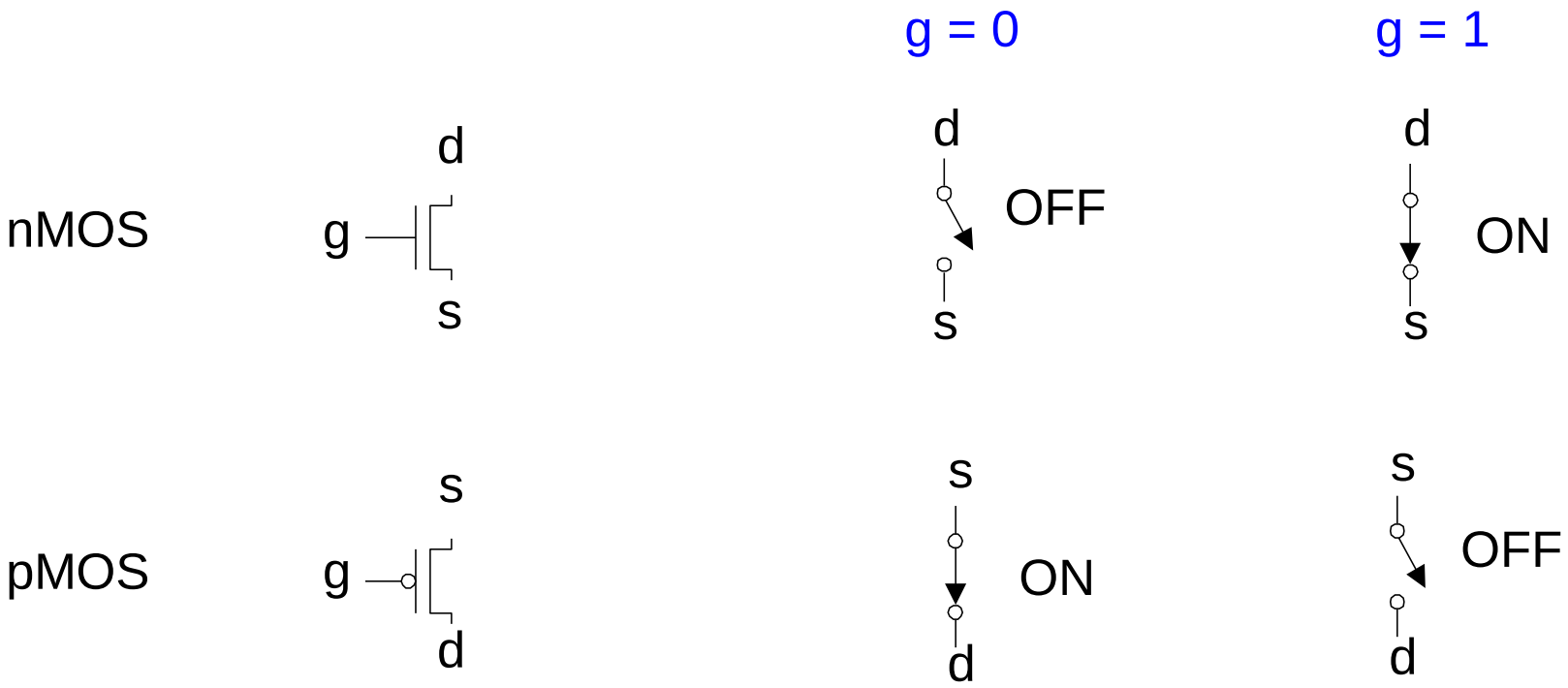
Transistors: pMOS

- pMOS transistor is opposite
  - ON when Gate = 0
  - OFF when Gate = 1



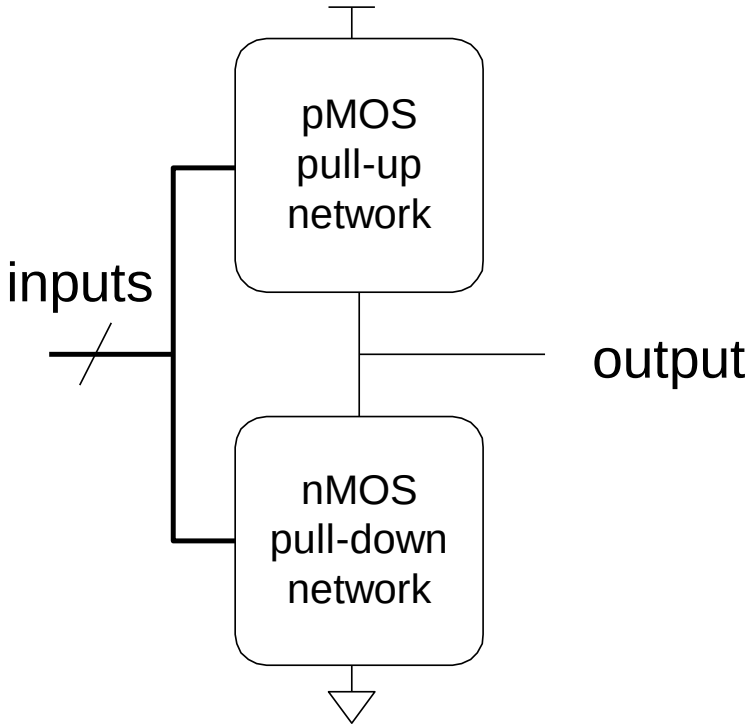


Transistor Function



Transistor Function

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

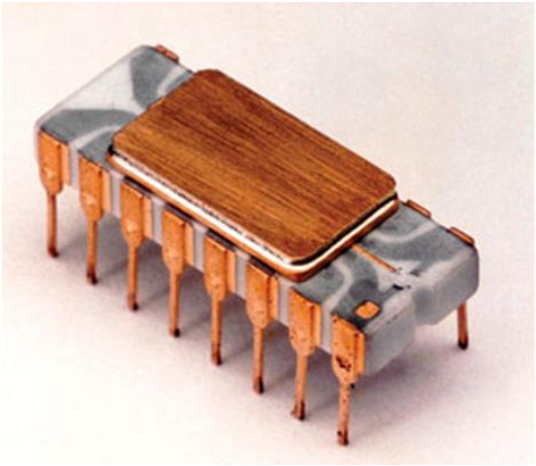


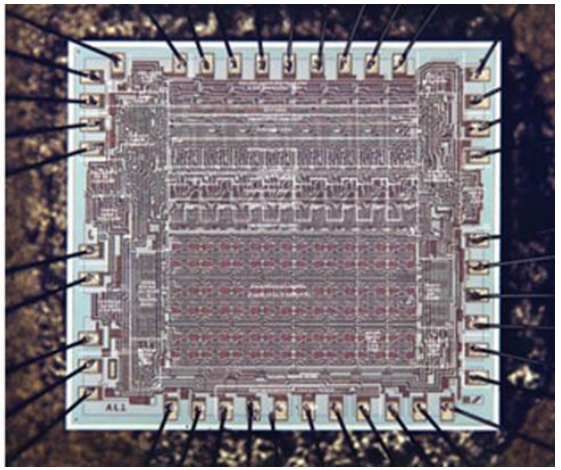
Robert Noyce, 1927 - 1990

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

Intel Microprocessor – 4004

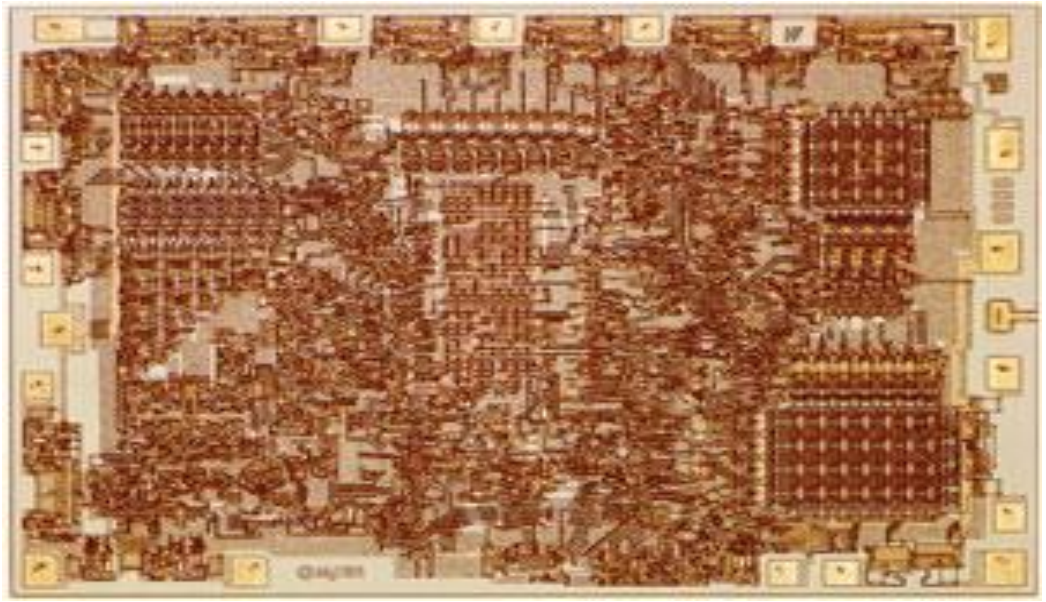
- Intel 4004 – 4-bit microprocessor – 1971 – 2300 Transistors
- First Microprocessor – Used in Calculator





Intel Microprocessor – 8008 -

- Intel 8008 – 8-bit microprocessor – 1972 – 3908 Transistors
- Intel Itanium microprocessor – 2 B transistors, 16 GB Flash Memory – 4 B transistors – 2008
-



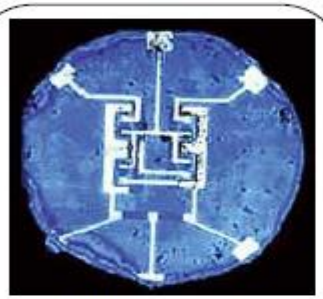
Brief History of IC

A Brief History of Integrated Circuits

1 Transistor
3 Resistors
1 Capacitor

Germanium IC
Jack Kilby
Texas Instruments

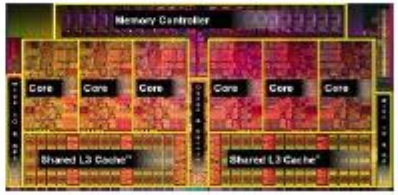
From



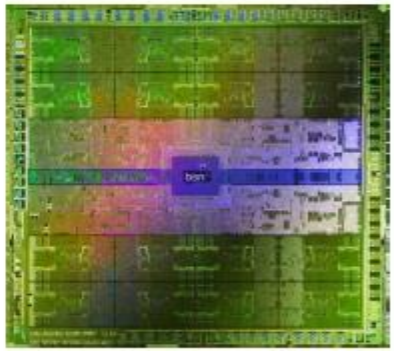
Silicon IC
Robert Noyce
Fairchild
Semiconductor

circa 1958

To



Intel Gulftown (six-core microprocessor)
1.17 Billion transistors
(32 nm)



NVIDIA GTX580
(Graphics processor with 512 "cores")
3.05 Billion transistors
(40 nm)

circa 2011

A 1,500,000,000 fold increase in 50 years

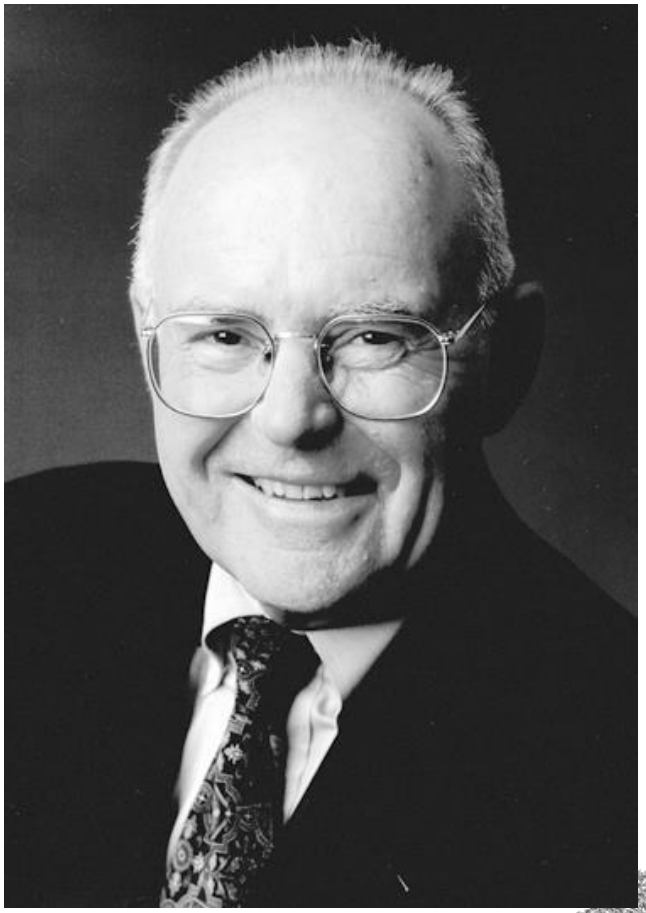
Growth Rate

- 53% compound annual growth rate over 50 years
  - No other technology has grown so fast so long
- Driven by miniaturization of transistors
  - Smaller is cheaper, faster, lower in power!
  - Revolutionary effects on society

[Moore65]
Electronics Magazine

Gordon Moore, 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.

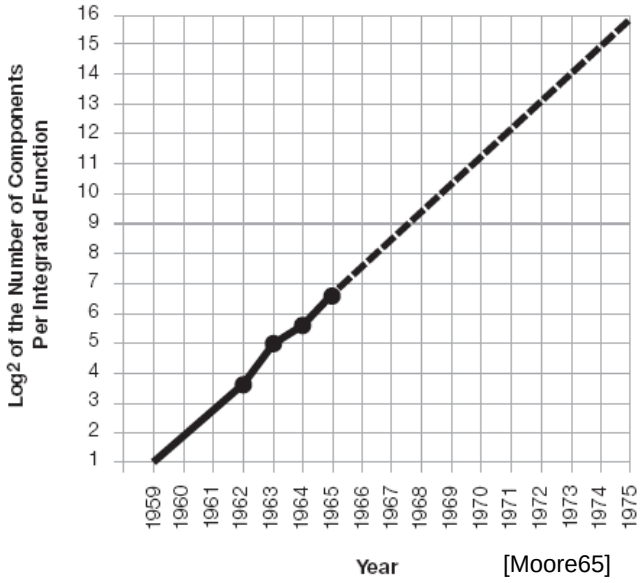


Moore's Law

- In 1965, Gordon Moore noted that the number of transistors on a chip doubled every 18 to 24 months.
- He made a prediction that semiconductor technology will double its effectiveness every 18 months

Moore's Law: Then

- 1965: Gordon Moore plotted transistor on each chip
  - Fit straight line on semilog scale
  - Transistor counts have doubled every 26 months



[Moore65]

Electronics Magazine

Integration Levels

SSI: 10 gates

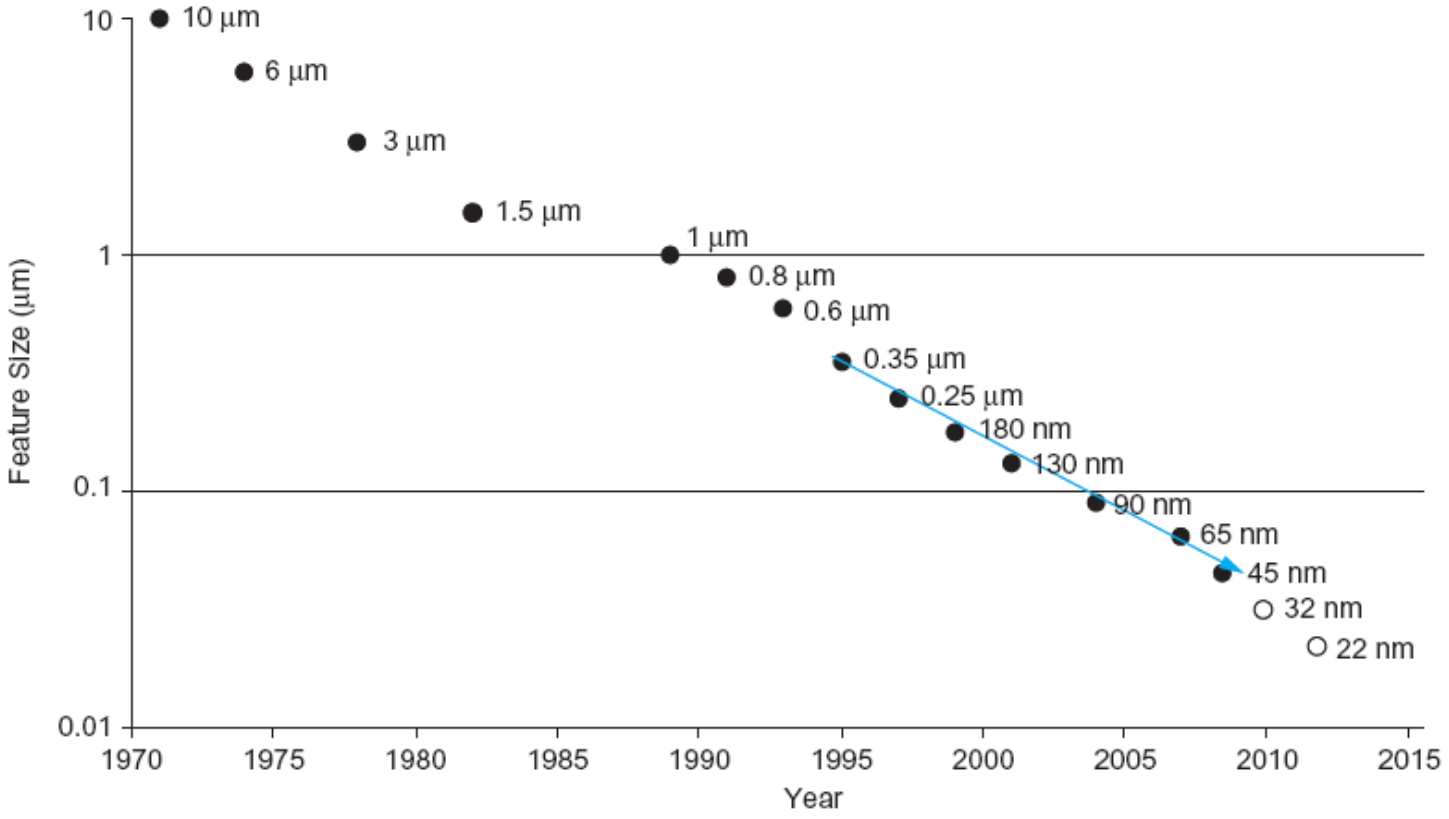
MSI: 1000 gates

LSI: 10,000 gates

VLSI: > 10k gates

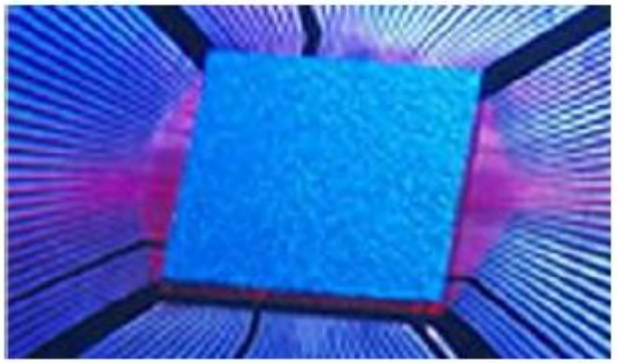
Feature Size

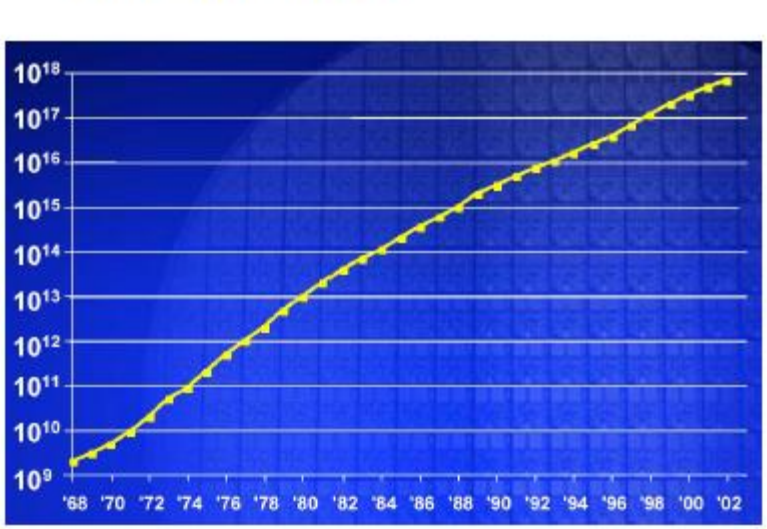
- Minimum feature size shrinking 30% every 2-3 years



We are drowning in transistors

- Ten quintillion: $10*10^{18}$
  - ~ The number of transistors shipped in 2004!
  - > The estimated number of grains of rice harvested in 2004
  - 100 times estimated number of ants on earth





Sources: "A Law of Continuing Returns", Los Angeles Times, April 17 2005
ISSCC 2004 Keynote, Gordon Moore

We are drowning in transistors

- Option #1 (general-purpose products)
  - Microprocessors, GPUs
    - Pre-2005: Deeper pipelines, more complex logic for instruction-level parallelism, more cache
    - 2005- : More cores, more cache
  - Memory chips
    - Easy – just keep increasing capacity
  - **???**

We are drowning in transistors

- Option #2 (98 % of "computing" systems)
  - Integrate more and more system functions onto a chip

- Benefits of integration
  - Size (miniaturization)
  - Cost
  - Performance
  - Power

System On a Chip (SoC)

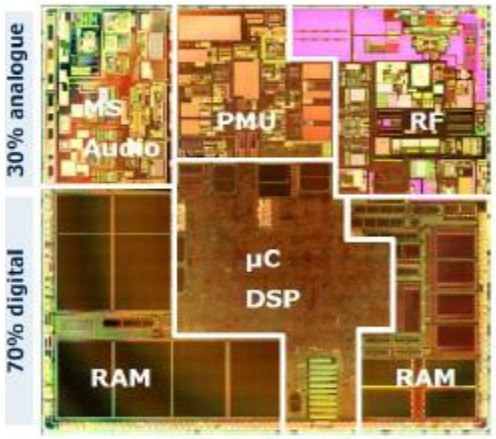
Honey I shrunk the Chip

How many people fit in a mini cooper? – How many IPs/Features in an SoC?

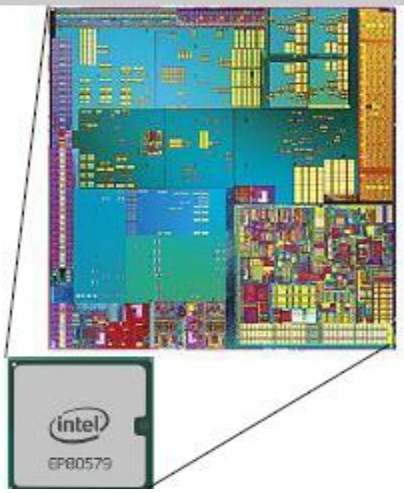
System on a Chip - SoC

- Direct consequence of increasing scales of integration (Moore's Law)
  - Integrate hitherto discrete system components into a single chip
  - Tremendous benefits in cost, size, performance, power consumption

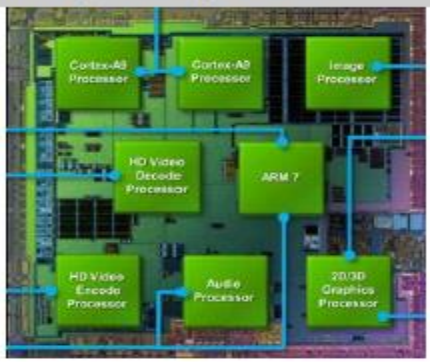
Infineon E-GOLDVoice "Phone-on-a-chip"



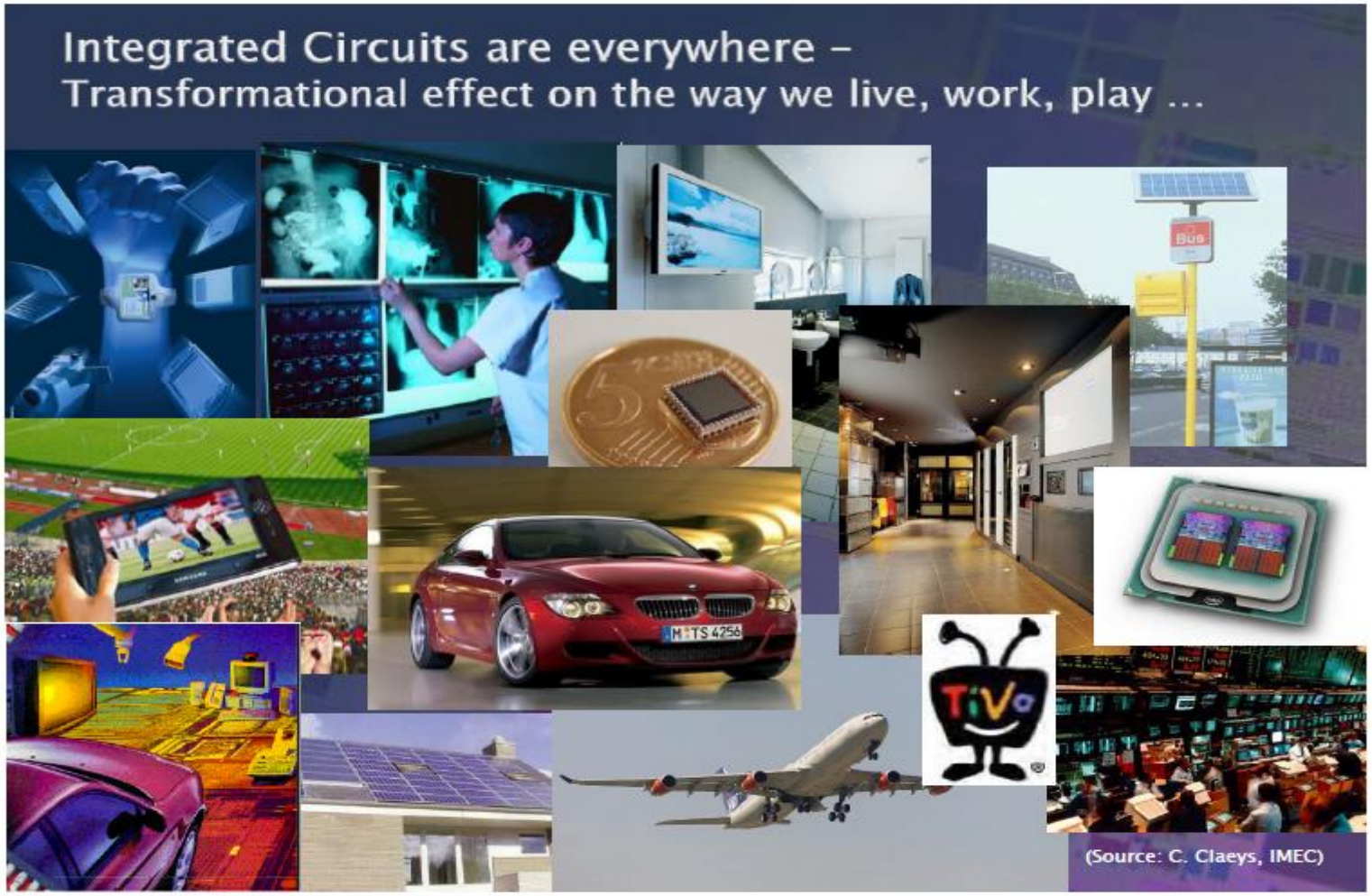
Intel EP80579 SoC for embedded systems (security, communications, storage)



NVIDIA Tegra2 SoC for tablets, slates, e-readers, MIDs, set-top boxes



Ubiquitous Integrated Circuits

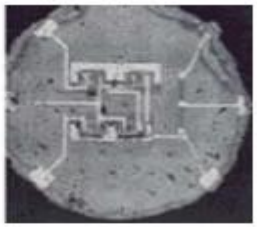


Historical Perspectives

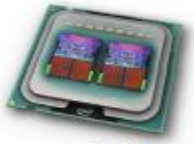
Technology Origins . . .

1947 Invention of the transistor (Bell Labs)

1954: First commercial transistor (TI)



1958: Invention IC
(J. Kilby, TI & Noyce, Fairchild)



Intel

and the rest is history …

1954: First *ISSCC*

Shockley (seated), Bardeen (left), and Brattain (Right)

SESSION IV
Friday, 2:30 p.m. - 5 p.m.

Nonlinear Applications of Junction Transistors
Chairman: A. W. Lo, *RCA Laboratories, Princeton, N. J.*

15. Large-Signal D-C Behavior of Junction Transistors
J. J. Ebers and J. L. Moll, *Bell Telephone Laboratories, Murray Hill, N. J.*

(about 3:30 p.m., Thursday)

7. Neutralization of Transistor Bandpass Amplifiers
F. P. Keiper, Jr., *Philco Corp., Philadelphia*

This paper will discuss the utilization of surface-barrier and junction transistors in bandpass amplifier circuits. The equivalent circuits useful in bandpass amplifier design will

ISSCC 1956

2:30 p.m.—5:30 p.m.—Irvine Auditorium
SWITCHING CIRCUITS
Chairman: H. E. Tompkins, *Burroughs Corporation*

2.1 Direct-Coupled Transistor Logic Circuitry in Digital Computers
J. R. Harris, *Bell Telephone Laboratories, Murray Hill*

(Source: http://sscs.org/History/isscc50/index.html)

Life Changers

A World Transformed: What Are the Top 30 Innovations of the Last 30 Years?

Published: February 18, 2009 in Knowledge@Wharton

1. Internet, broadband, WWW (browser and html)
2. PC/laptop computers
3. Mobile phones
4. E-mail
5. DNA testing and sequencing/Human genome mapping
6. Magnetic Resonance Imaging (MRI)
7. Microprocessors
8. Fiber optics
9. Office software (spreadsheets, word processors)
10. Non-invasive laser/robotic surgery (laparoscopy)
11. Open source software and services (e.g., Linux, Wikipedia)
12. Light emitting diodes
13. Liquid crystal display (LCD)
14. GPS systems
15. Online shopping/ecommerce/auctions (e.g., eBay)
16. Media file compression (jpeg, mpeg, mp3)
17. Microfinance
18. Photovoltaic Solar Energy
19. Large scale wind turbines
20. Social networking via the Internet
21. Graphic user interface (GUI)
22. Digital photography/videography
23. RFID and applications (e.g., EZ Pass)
24. Genetically modified plants
25. Bio fuels
26. Bar codes and scanners
27. ATMs
28. Stents
29. SRAM flash memory
30. Anti retroviral treatment for AIDS

Of these 30 innovations , 10 are directly related to advances in Digital Logic and Solid State Circuits;

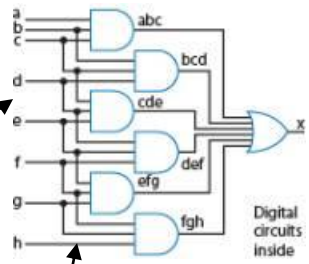
Another 8 are the indirect results of ICs.

Source: http://knowledge.wharton.upenn.edu/article.cfm?articleid=2163

Part 2 –
Basics of Digital Systems

Why Study Digital Design?

- Look “under the hood” of computers
  - Solid understanding --> confidence, insight, even better programmer when aware of hardware resource issues
- Electronic devices becoming digital
  - Enabled by shrinking and more capable chips
  - Enables:
    - Better devices: Sound recorders, cameras, cars, cell phones, medical devices,...
    - New devices: Video games, PDAs, ...
  - Known as “embedded systems”
    - Thousands of new devices every year
    - Designers needed: Potential career direction



Satellites
DVD players
Video recorders
Musical instruments
Portable music players
Cell phones
Cameras
TVs
???

1995 1997 1999 2001 2003 2005 2007

- Years shown above indicate when digital version began to *dominate*
  - (*Not* the first year that a digital version appeared)

Discipline

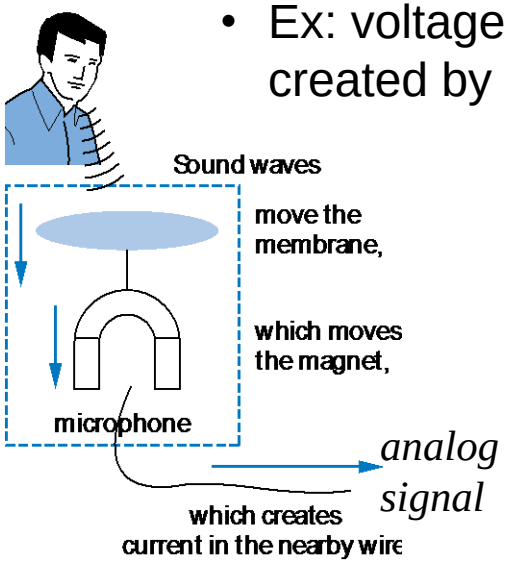
- Intentionally restrict design choices
- Example: Digital discipline
  - Discrete voltages instead of continuous
  - Simpler to design than analog circuits – can build more sophisticated systems
  - Digital systems replacing analog predecessors:
    - i.e., digital cameras, digital television, cell phones, CDs

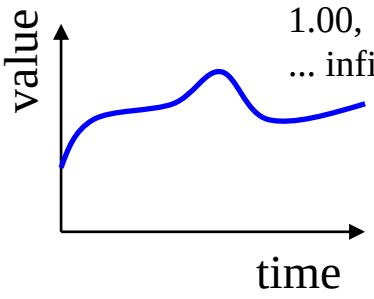
The Digital Abstraction

- Most physical variables are continuous, for example
  - Voltage on a wire
  - Frequency of an oscillation
  - Position of a mass
- Instead of considering all values, the digital abstraction considers only a discrete subset of values

What Does "Digital" Mean?

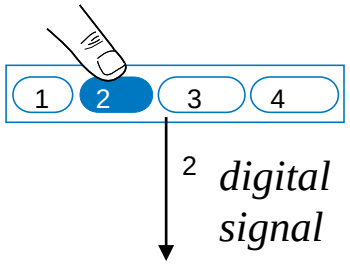
- Analog signal
  - Infinite possible values
    - Ex: voltage on a wire created by microphone

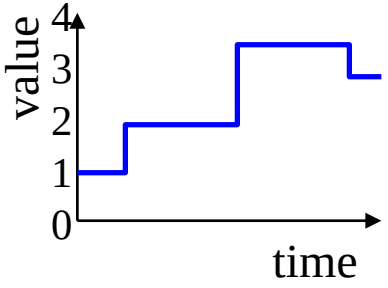




Possible values:
1.00, 1.01, 2.0000009, ... infinite possibilities

- Digital signal
  - Finite possible values
    - Ex: button pressed on a keypad

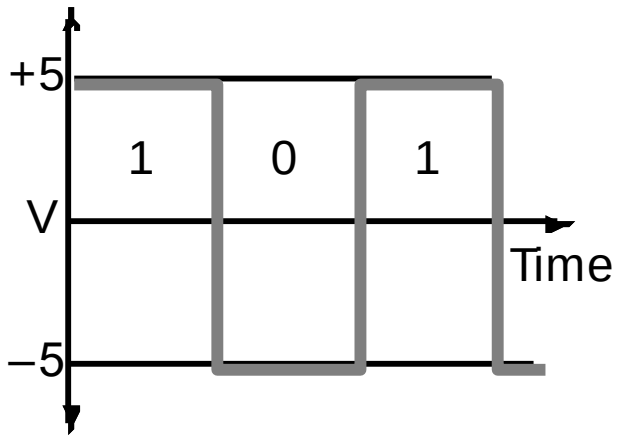


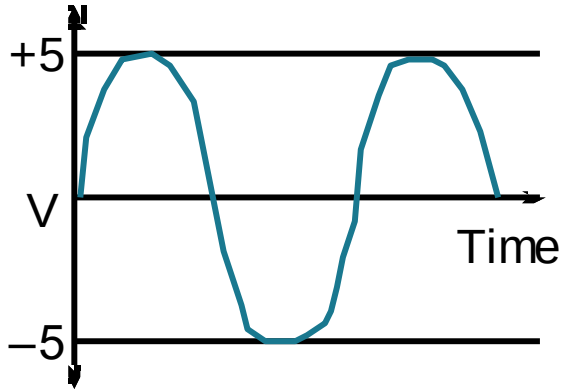


Possible values:
0, 1, 2, 3, or 4.
That's it.

Digital Systems

Digital vs. Analog Waveforms



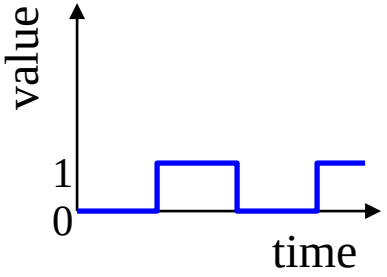


Digital:
only assumes discrete values

Analog:
values vary over a broad range continuously

Digital Signals with Only Two Values: Binary

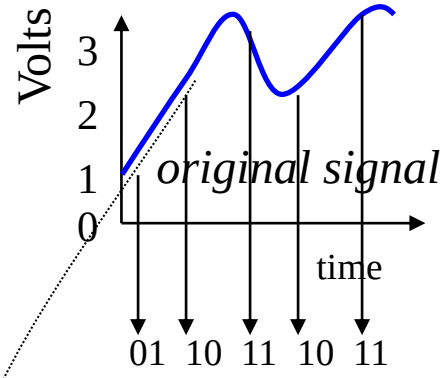
- ***Binary*** digital signal -- only *two* possible values
  - Typically represented as **0** and **1**
  - One *b*inary dig*it* is a ***bit***
  - We'll only consider *binary* digital signals
  - Binary is popular because
    - Transistors, the basic digital electric component, operate using *two* voltages (more in Chpt. 2)
    - Storing/transmitting one of *two* values is easier than three or more (e.g., loud beep or quiet beep, reflection or no reflection)

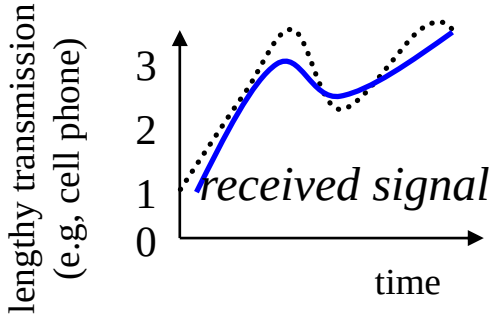


Example of Digitization Benefit

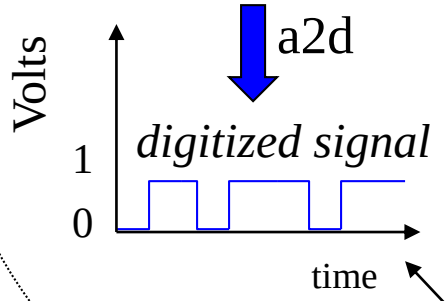
- Analog signal (e.g., audio, video) may lose quality
  - Voltage levels not saved/copied/transmitted perfectly
- Digitized version enables near-perfect save/cpy/tran.
  - "Sample" voltage at particular rate, save sample using bit encoding
  - Voltage levels still not kept perfectly
  - But we can distinguish 0s from 1s

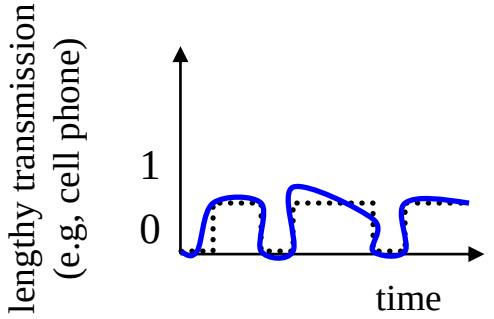
Let bit encoding be:
1 V: "01"
2 V: "10"
3 V: "11"





How fix -- higher, lower, ?

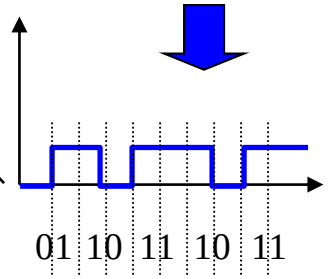


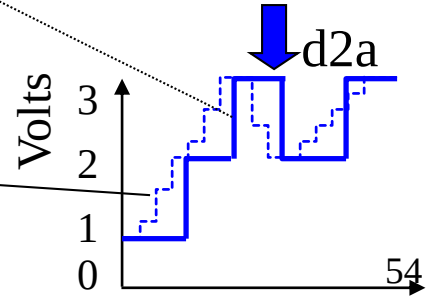


Can fix—distinguish 0s/1s, restore

same

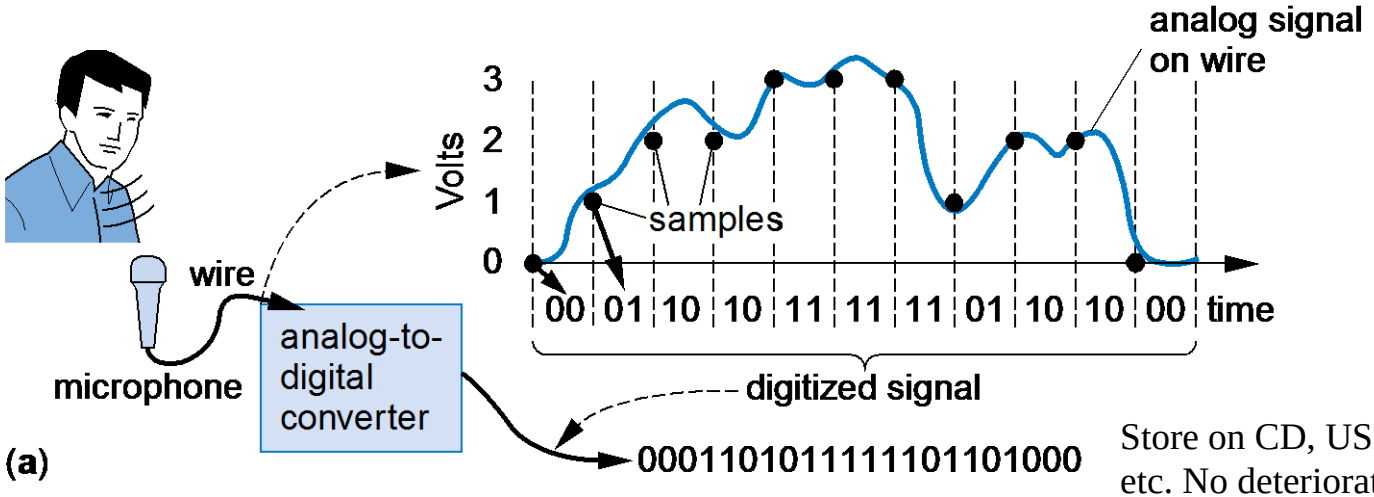
Not a perfect re-creation due to a2d and d2a



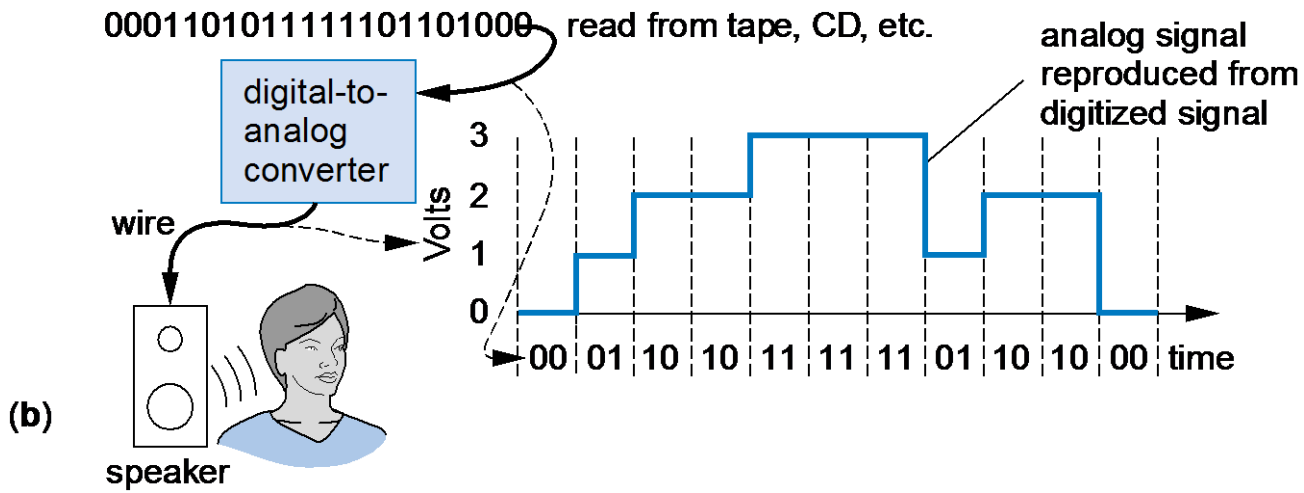


Higher sampling rate and more bits per encoding improves re-creation

Digitization Benefit: Can Store on Digital Media



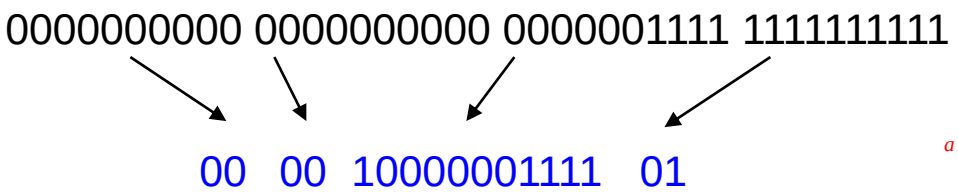
Store on CD, USB drive, etc. No deterioration.



Digitized Audio: Compression Benefit

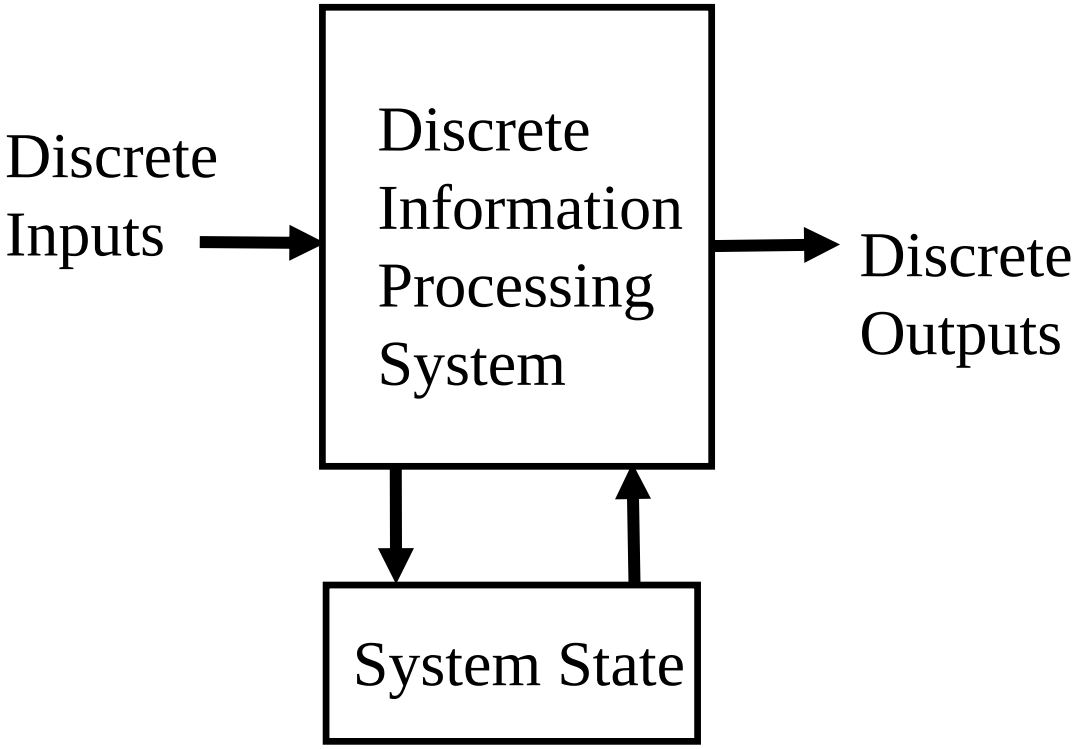
- Digitized audio can be compressed
  - e.g., MP3s
  - A CD can hold about 20 songs uncompressed, but about 200 compressed
- Compression also done on digitized pictures (jpeg), movies (mpeg), and more
- Digitization has many other benefits too

Example compression scheme:
00 means 0000000000
01 means 1111111111
1X means X



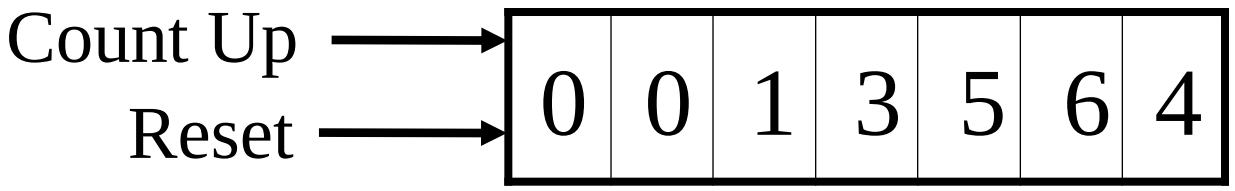
DIGITAL & COMPUTER SYSTEMS - Digital System

- **Takes a set of discrete information inputs and discrete internal information (system state) and generates a set of discrete information outputs.**



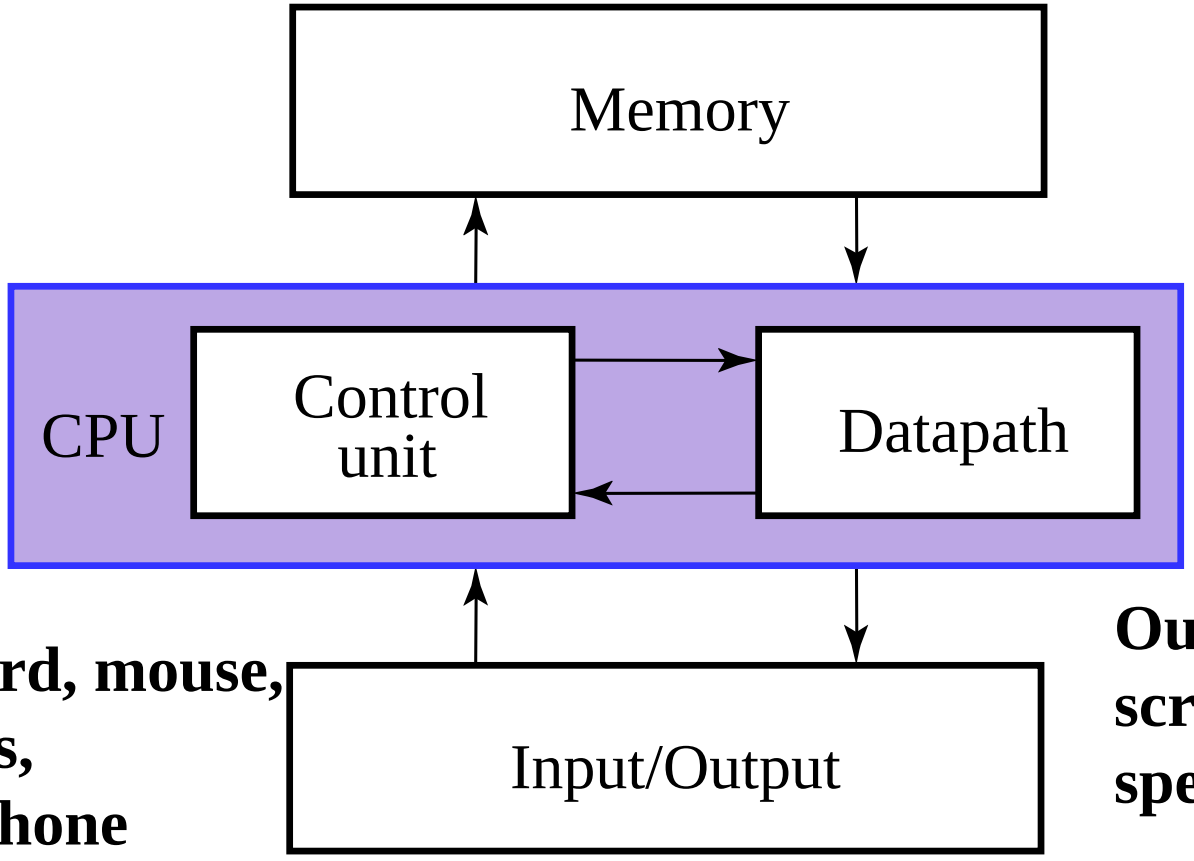
Digital System Example:

A Digital Counter (e. g., odometer):



Inputs: **Count Up, Reset**
Outputs: **Visual Display**
State: **"Value" of stored digits**

Digital Computer Example



Inputs: keyboard, mouse, wireless, microphone

Outputs: LCD screen, wireless, speakers

And Beyond – Embedded Systems

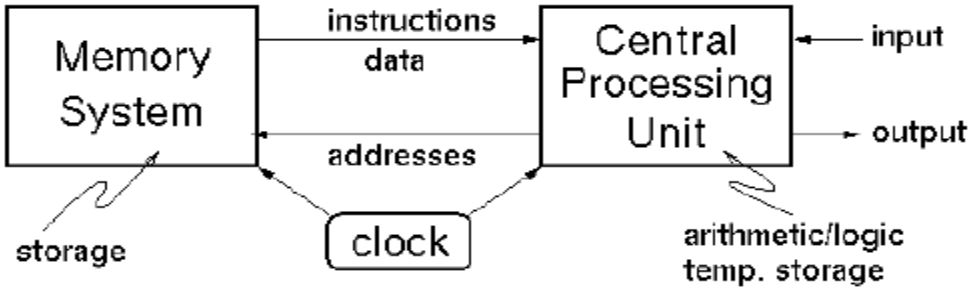
- Computers as integral parts of other products
- Examples of embedded computers
  - Microcomputers
  - Microcontrollers
  - Digital signal processors

Examples of Digital Revolution

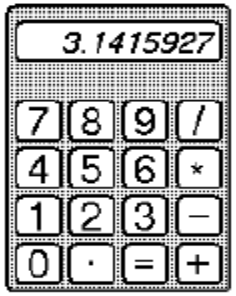
- Digital Cameras – Still Pictures
- Video Recordings
- Audio Recordings
- Traffic Lights
- Movie Effects

Examples of Digital Revolution

- Digital Computer



  - Usually design to maximize performance. "Optimized for speed"
- Handheld Calculator



- Usually designed to minimize cost. "Optimized for low cost"

- Of course, low cost comes at the expense of speed.

Advantages of Digital

- Digital Watch

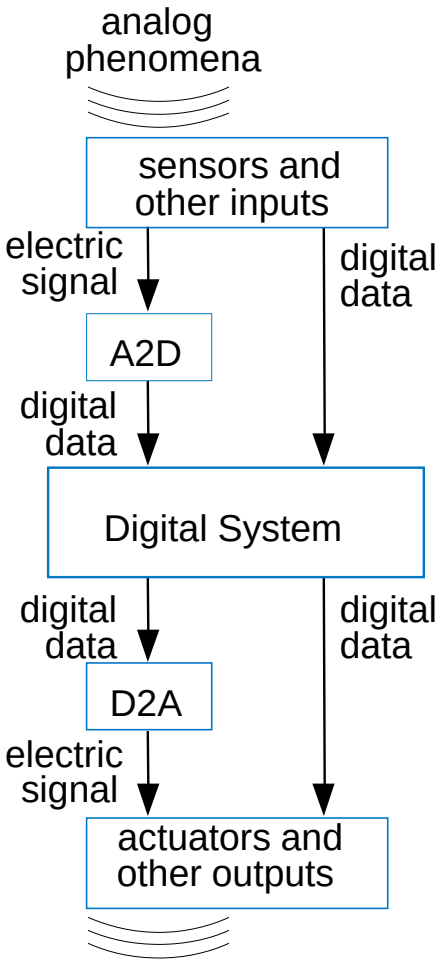
Designed to minimize power.
Single battery must last for years.

  - Low power operation comes at the expense of:
    - lower speed
    - higher cost

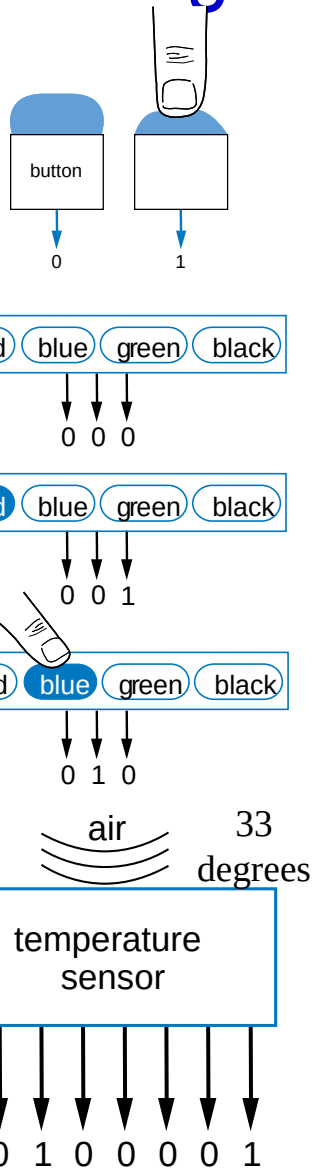
Advantages of Digital

- Reproducibility of results – Analog depends on temperature, power supply voltage, component aging etc.,
- Ease of design
  - Logic Design – No Math – Operation viewed mentally without insights about the operation of capacitors, transistors etc.,
- Flexibility and Functionality – scrambling voice
- Programmable
- Speed
- Economy
- Steadily advancing technology

How Do We Encode Data as Binary for Our Digital System?



- Some inputs inherently binary
  - Button: not pressed (0), pressed (1)
- Some inputs inherently digital
  - Just need encoding in binary
  - e.g., multi-button input: encode red=001, blue=010, ...
- Some inputs analog
  - Need analog-to-digital conversion
  - As done in earlier slide -- sample and encode with bits



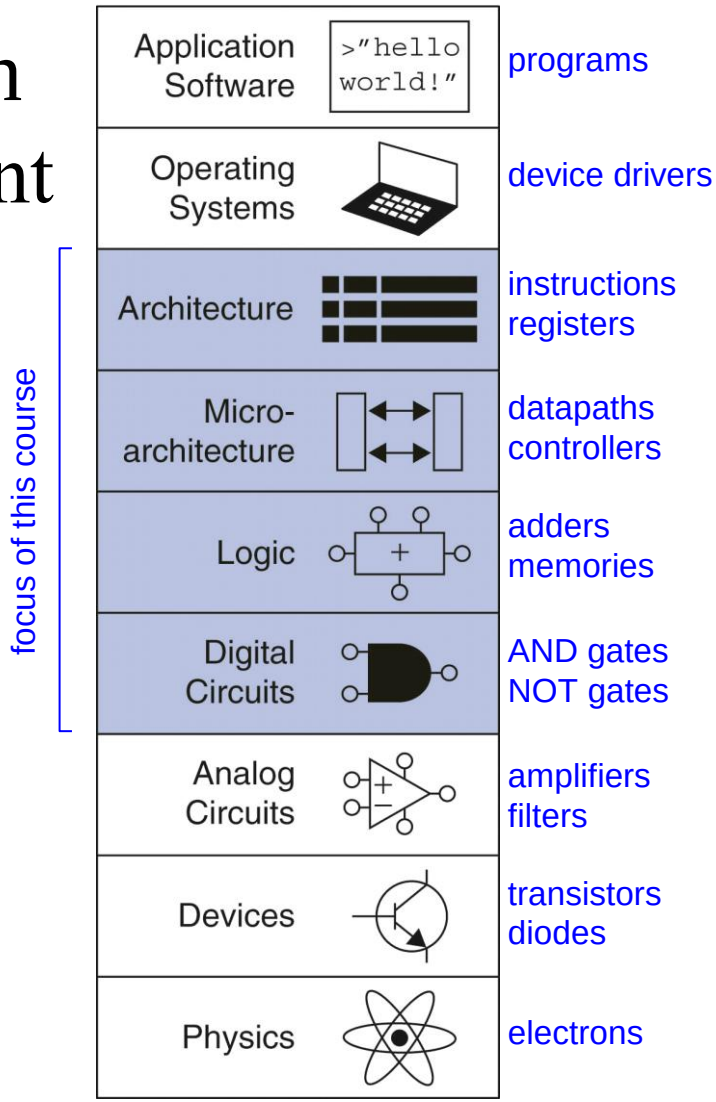
The Art of Managing Complexity

- Abstraction
- Discipline
- The Three –y's
  - Hierarch**y**
  - Modularit**y**
  - Regularit**y**

Thoughts on Complexity - Whack-A-Mole Design Fixes

Abstraction

- Hiding details when they aren't important



The Three -y's

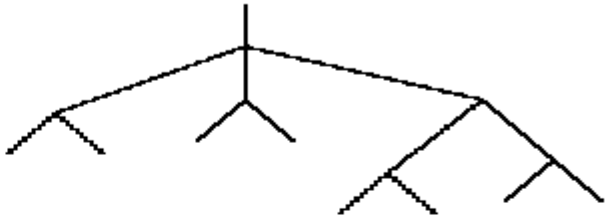
- **Hierarchy**
- **Modularity**
- **Regularity**

The Three -Y's

- Hierarchy
  - A system divided into modules and submodules
- Modularity
  - Having well-defined functions and interfaces
- Regularity
  - Encouraging uniformity, so modules can be easily reused

Design Hierarchy

- Helps control complexity -
  - by hiding details and reducing the total number of things to handle at any time.
- Modulalizes the design -
  - divide and conquer
  - simplifies implementation and debugging

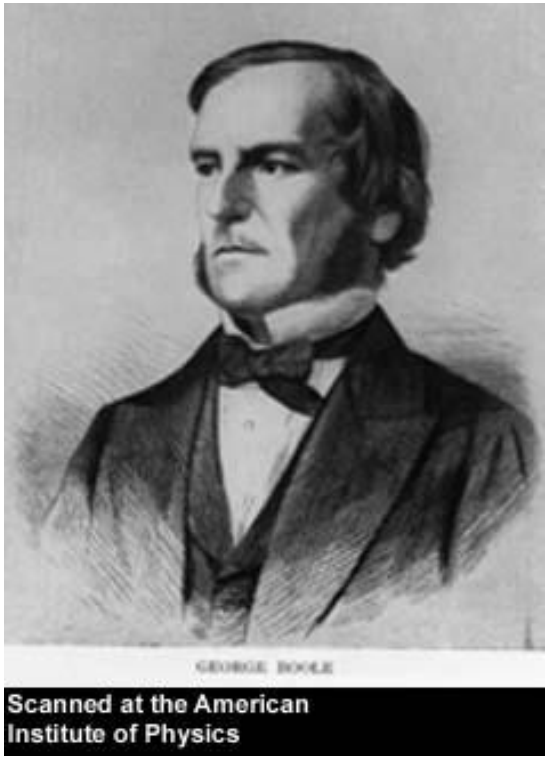


- Top-Down Design
  - Starts at the top (root) and works down by successive refinement.
- Bottom-up Design
  - Starts at the leaves & puts pieces together to build up the design.
- Which is better?
  - In practice both are needed & used.
    - Need top-down divide and conquer to handle the complexity.
    - Need bottom-up because in a well designed system, the structure is influence by what primitves are available.

Part 3
Basics of Digital Logic

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.



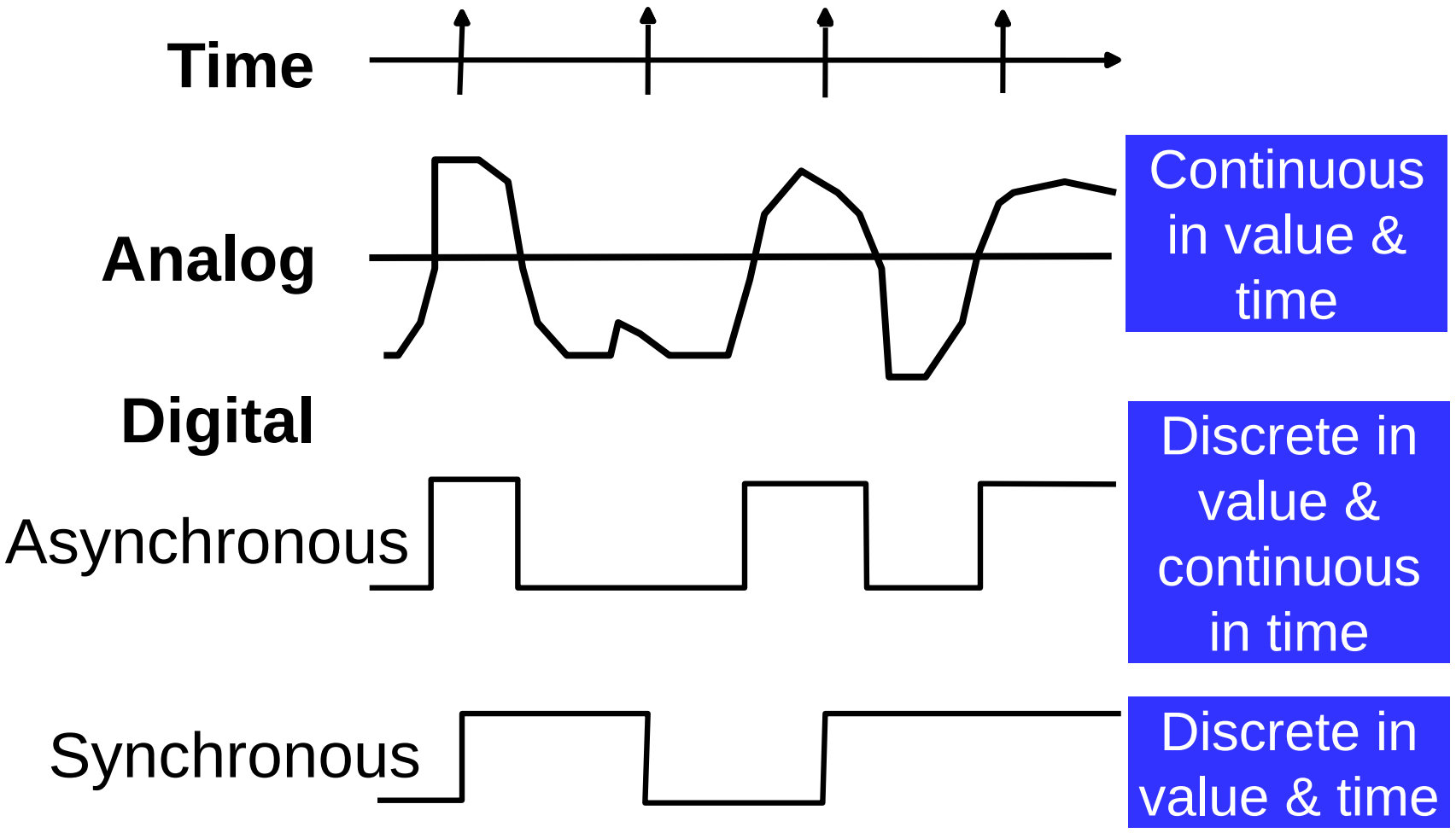
Digital Discipline: Binary Values

- 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*

INFORMATION REPRESENTATION - Signals

- **Information variables represented by physical quantities.**
- **For digital systems, the variables take on discrete values.**
- **Two level, or binary values are the most prevalent values in digital systems.**
- **Binary values are represented abstractly by:**
  - **digits 0 and 1**
  - **words (symbols) False (F) and True (T)**
  - **words (symbols) Low (L) and High (H)**
  - **and words On and Off.**
- **Binary values are represented by values or ranges of values of physical quantities**

Signal Examples Over Time



Boolean Algebra and its Relation to Digital Circuits

- To understand the benefits of “logic gates” vs. switches, we should first understand Boolean algebra
- “Traditional” algebra
  - Variables represent real numbers (x, y)
  - Operators operate on variables, return real numbers (2.5*x + y - 3)
- ***Boolean Algebra***
  - Variables represent 0 or 1 only
  - Operators return 0 or 1 only
  - Basic operators
    - AND: *a AND b* returns 1 only when both a=1 and b=1
    - OR: *a OR b* returns 1 if either (or both) a=1 or b=1
    - NOT: *NOT a* returns the opposite of a (1 if a=0, 0 if a=1)

a	b	AND
0	0	0
0	1	0
1	0	0
1	1	1

a	b	OR
0	0	0
0	1	1
1	0	1
1	1	1

a	NOT
0	1
1	0

Relating Boolean Algebra to Digital Design

Boolean algebra (mid-1800s)

Boole's intent: formalize human thought

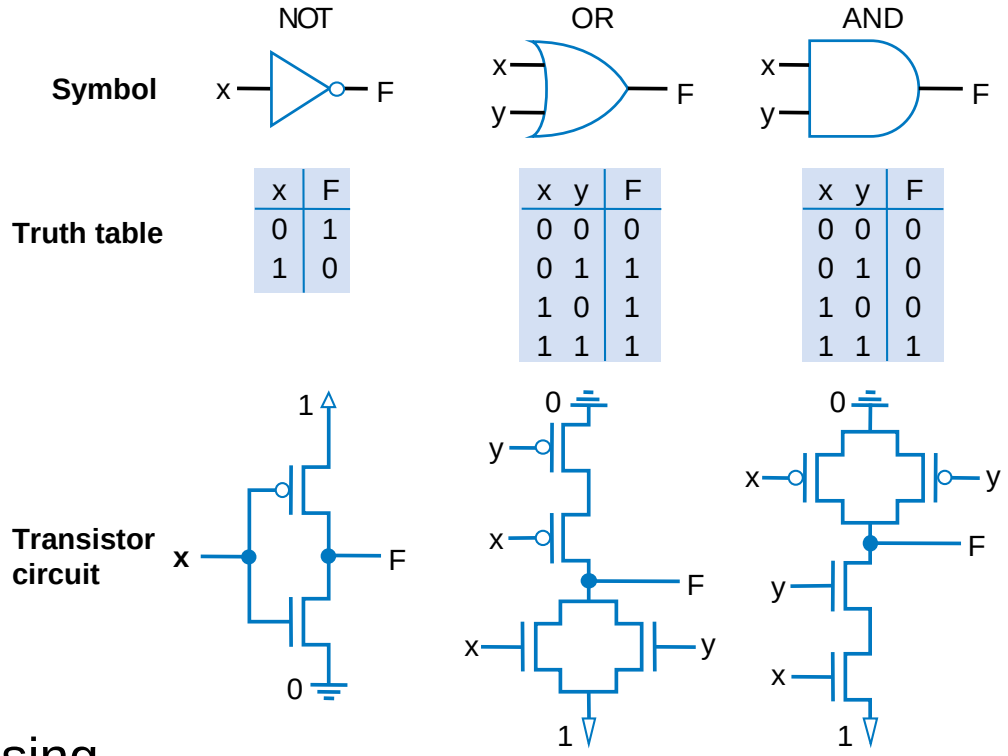
Switches (1930s)

For telephone switching and other electronic uses

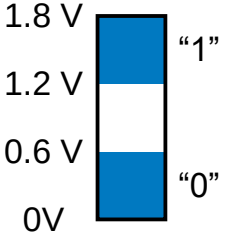
Shannon (1938)

Showed application of Boolean algebra to design of switch-based circuits

Digital design



- Implement Boolean operators using transistors
  - Call those implementations ***logic gates***.
  - ***Lets us build circuits by doing math*** -- powerful concept



1 and 0 each actually corresponds to a voltage range

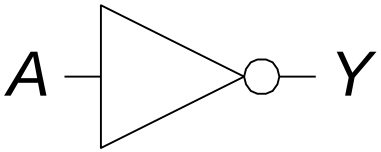
Next slides show how these circuits work.
Note: The above OR/AND implementations are inefficient; we'll show why, and show better ones, later.

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

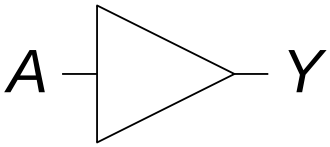
NOT



$Y = \overline{A}$

A	Y
0	
1	

BUF

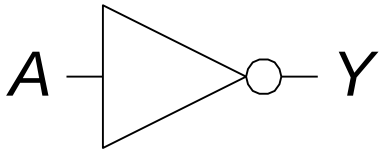


$Y = A$

A	Y
0	
1	

Single-Input Logic Gates

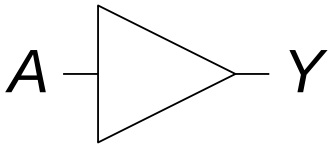
NOT



$Y = \overline{A}$

A	Y
0	1
1	0

BUF

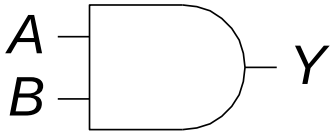


$Y = A$

A	Y
0	0
1	1

Two-Input Logic Gates

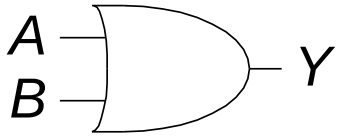
AND



$Y = AB$

A	B	Y
0	0	
0	1	
1	0	
1	1	

OR

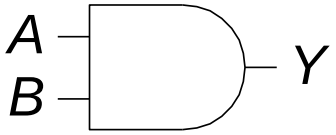


$Y = A + B$

A	B	Y
0	0	
0	1	
1	0	
1	1	

Two-Input Logic Gates

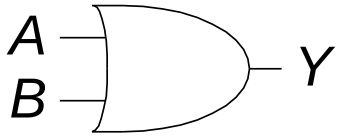
AND



$Y = AB$

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

OR

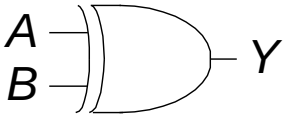


$Y = A + B$

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

More Two-Input Logic Gates

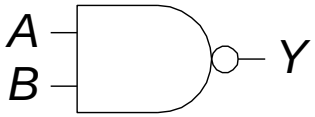
XOR



$Y = A \oplus B$

A	B	Y
0	0	
0	1	
1	0	
1	1	

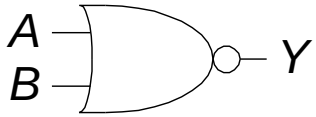
NAND



$Y = \overline{AB}$

A	B	Y
0	0	
0	1	
1	0	
1	1	

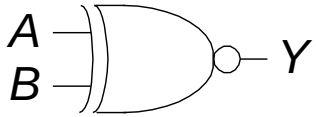
NOR



$Y = \overline{A + B}$

A	B	Y
0	0	
0	1	
1	0	
1	1	

XNOR

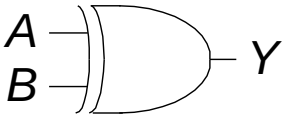


$Y = \overline{A \oplus B}$

A	B	Y
0	0	
0	1	
1	0	
1	1	

More Two-Input Logic Gates

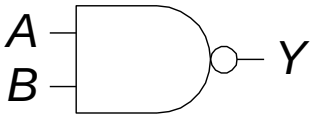
XOR



$Y = A \oplus B$

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

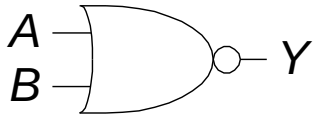
NAND



$Y = \overline{AB}$

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

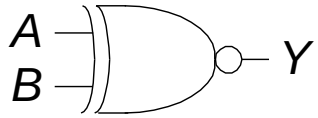
NOR



$Y = \overline{A + B}$

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

XNOR

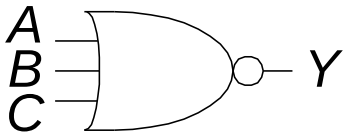


$Y = \overline{A \oplus B}$

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

Multiple-Input Logic Gates

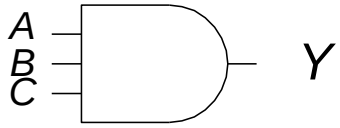
NOR3



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

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	

AND3



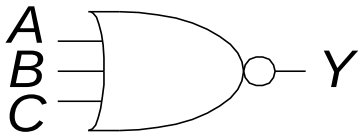
$$Y = ABC$$

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	

FROM ZERO TO ONE

Multiple-Input Logic Gates

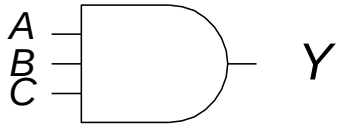
NOR3



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

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

AND3



$$Y = ABC$$

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

- Multi-input XOR: Odd parity

Logic Levels

- Discrete voltages represent 1 and 0
- For example:
  - 0 = *ground* (GND) or 0 volts
  - 1 = V_{DD} or 5 volts
- What about 4.99 volts? Is that a 0 or a 1?
- What about 3.2 volts?

Logic Levels

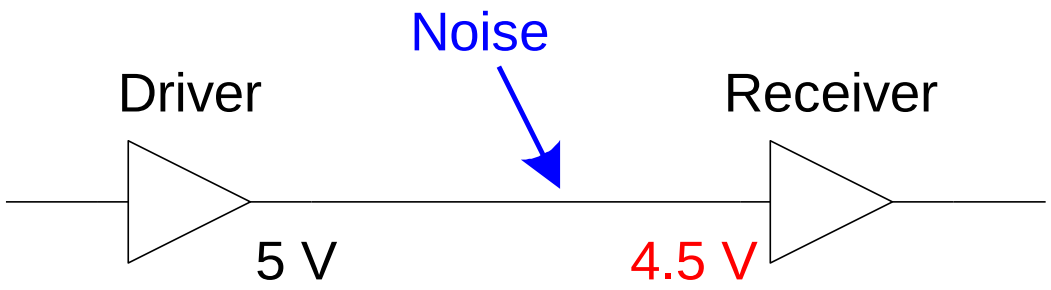
- *Range* of voltages for 1 and 0
- Different ranges for inputs and outputs to allow for *noise*

What is Noise?

FROM ZERO TO ONE

What is Noise?

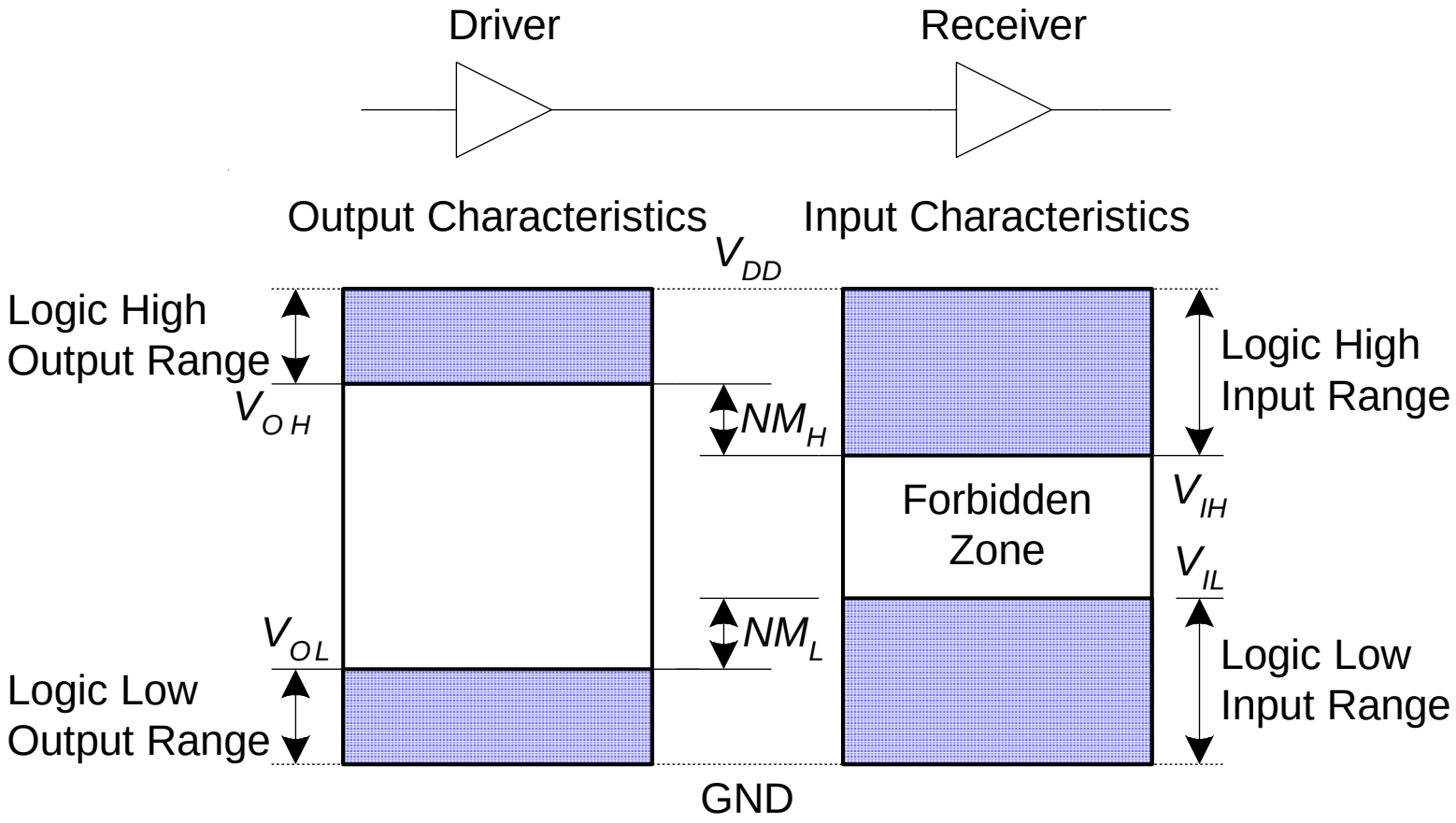
- **Anything that degrades the signal**
  - E.g., resistance, power supply noise, coupling to neighboring wires, etc.
- **Example:** a gate (driver) outputs 5 V but, because of resistance in a long wire, receiver gets 4.5 V



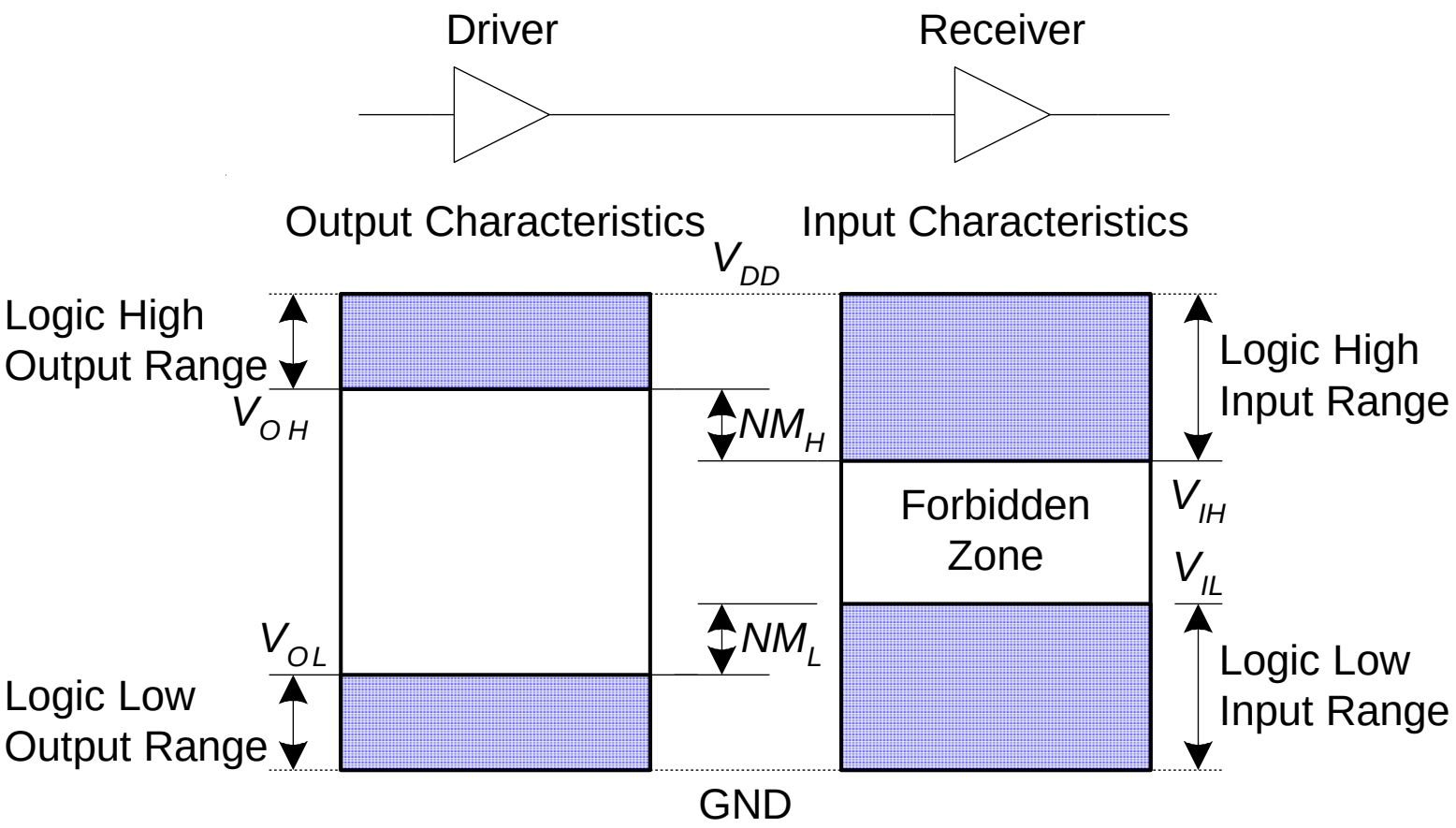
The Static Discipline

- With logically valid inputs, every circuit element must produce logically valid outputs
- Use limited ranges of voltages to represent discrete values

Logic Levels



Noise Margins

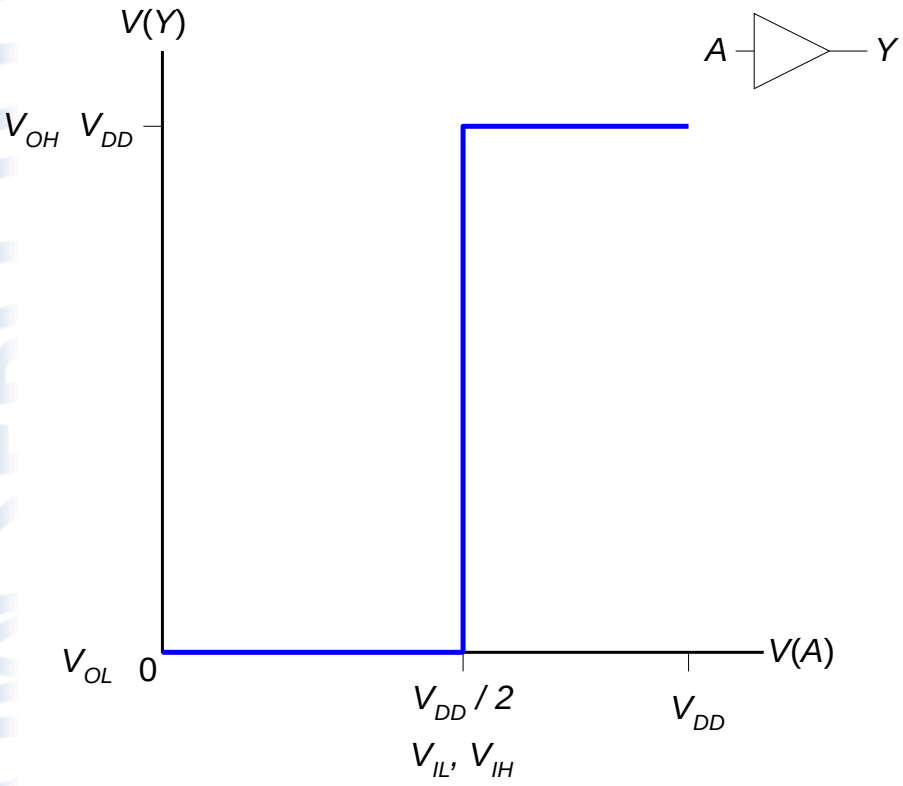


$$NM_H = V_{OH} - V_{IH}$$

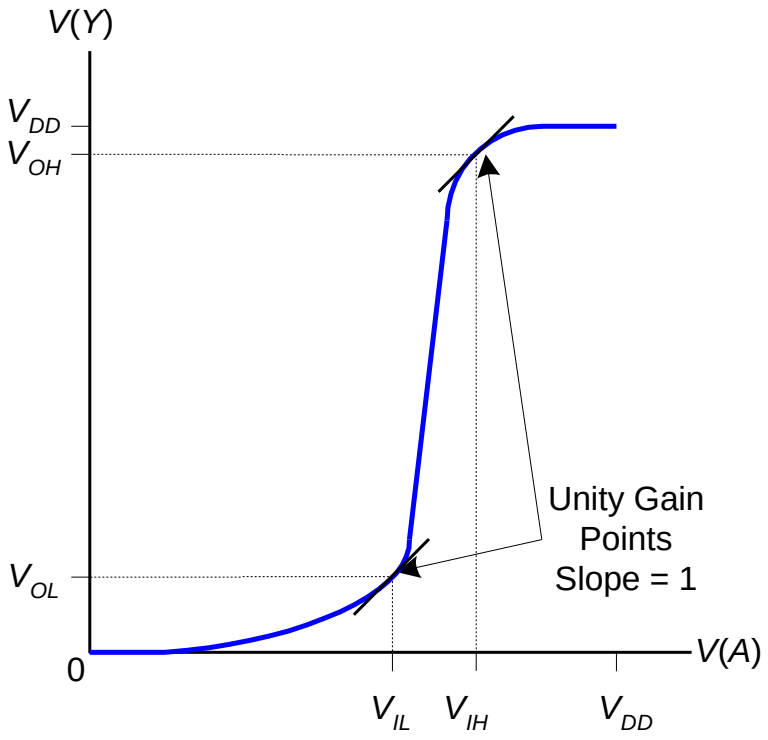
$$NM_L = V_{IL} - V_{OL}$$

DC Transfer Characteristics

Ideal Buffer:



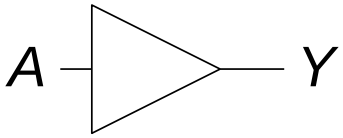
Real Buffer:

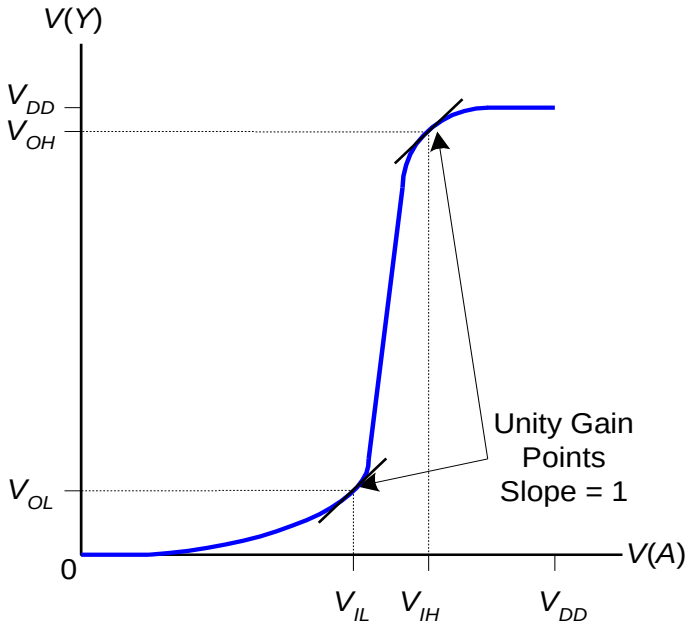


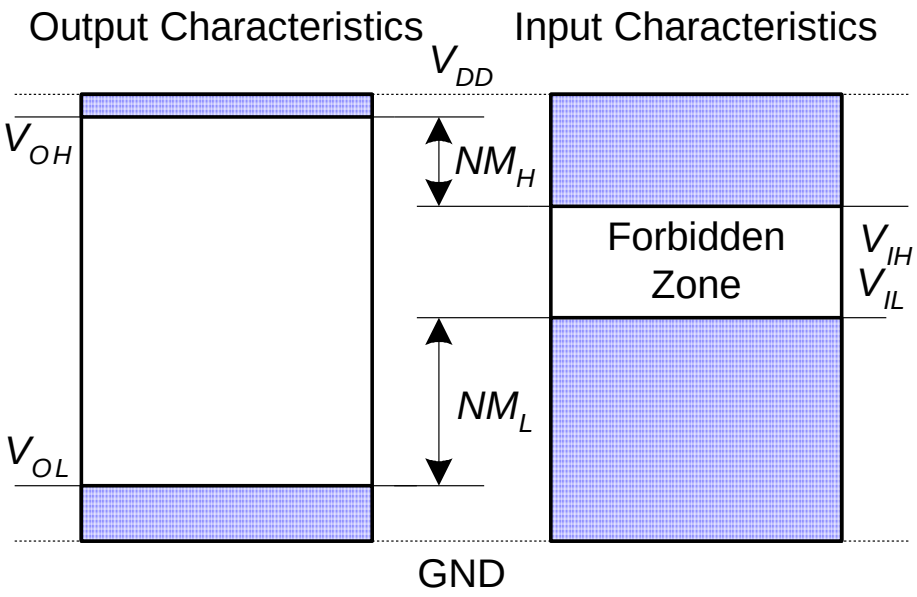
$NM_H = NM_L = V_{DD}/2$

$NM_H , NM_L < V_{DD}/2$

DC Transfer Characteristics







Logic Family Examples

Logic Family	V_{DD}	V_{IL}	V_{IH}	V_{OL}	V_{OH}
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
LVCMOS	3.3 (3 - 3.6)	0.9	1.8	0.36	2.7

Boolean Algebra and its Relation to Digital Circuits

- Developed mid-1800's by George Boole to formalize human thought
  - Ex: "I'll go to lunch if Mary goes OR John goes, AND Sally does not go."
    - Let F represent my going to lunch (1 means I go, 0 I don't go)
    - Likewise, m for Mary going, j for John, and s for Sally
    - Then **F = (m OR j) AND NOT(s)**
  - Nice features
    - Formally evaluate
      - m=1, j=0, s=1 --> F = (1 OR 0) AND NOT(1) = 1 AND 0 = **0**
    - Formally transform
      - F = (m and NOT(s)) OR (j and NOT(s))
        - » Looks different, but same function
        - » We'll show transformation techniques soon
    - Formally prove
      - Prove that if Sally goes to lunch (s=1), then I don't go (F=0)
      - F = (m OR j) AND NOT(1) = (m OR j) AND 0 = 0

a	b	AND
0	0	0
0	1	0
1	0	0
1	1	1

a	b	OR
0	0	0
0	1	1
1	0	1
1	1	1

a	NOT
0	1
1	0

Converting to Boolean Equations

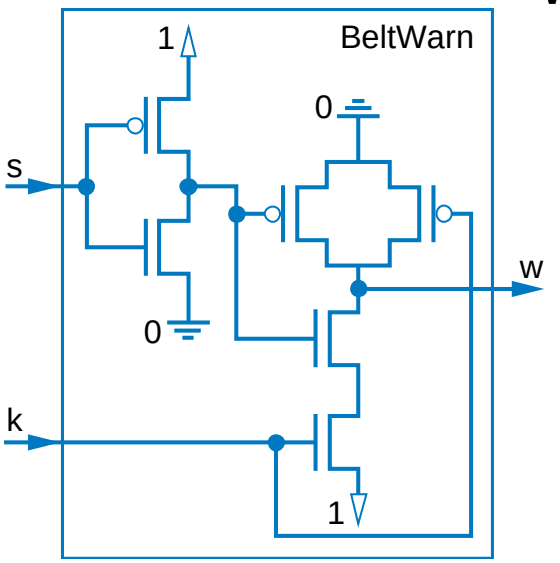
- Q1. A fire sprinkler system should spray water if high heat is sensed and the system is set to enabled.
  - Answer: Let Boolean variable h represent “high heat is sensed,” e represent “enabled,” and F represent “spraying water.” Then an equation is: F = h AND e.
- Q2. A car alarm should sound if the alarm is enabled, and either the car is shaken or the door is opened.
  - Answer: Let a represent “alarm is enabled,” s represent “car is shaken,” d represent “door is opened,” and F represent “alarm sounds.” Then an equation is: F = a AND (s OR d).
  - (a) Alternatively, assuming that our door sensor d represents “door is closed” instead of open (meaning d=1 when the door is closed, 0 when open), we obtain the following equation: F = a AND (s OR NOT(d)).

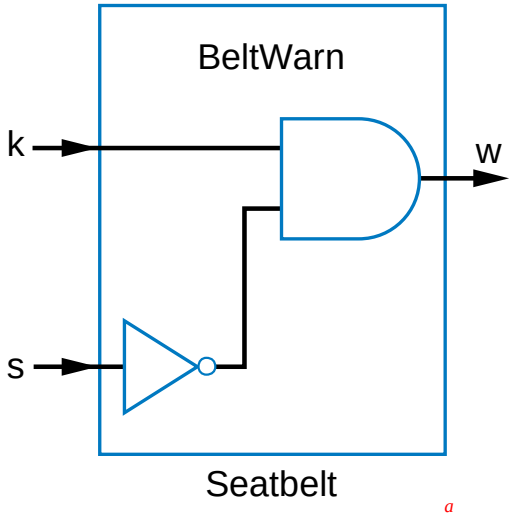
Gates vs. switches

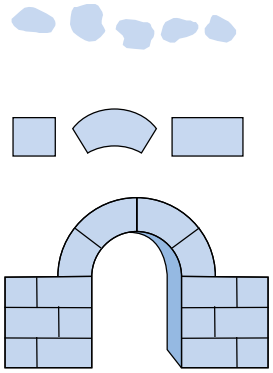
Notice

- Boolean algebra enables easy capture as equation and conversion to circuit
  - How design with switches?
  - Of course, logic gates are built from switches, but we think at level of logic gates, not switches

w = NOT(s) AND k



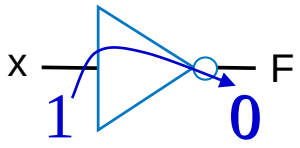




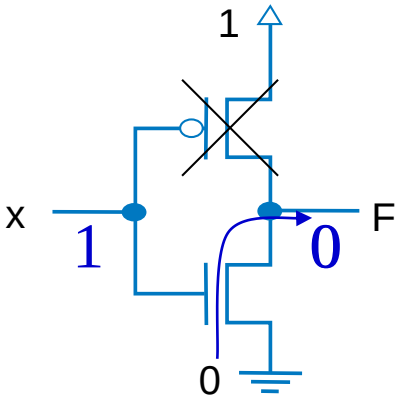
State Representing a bit

Technology	State Representing Bit	
	0	1
Relay logic	Circuit open	Circuit closed
CMOS logic	0-1.5V	3.5-5V
Transistor-transistor logic	0-0.8V	2-5V
Fiber optics	Light off	Light on
Dynamic memory	Capacitor discharged	Capacitor charged
Nonvolatile, erasable memory	Electrons trapped	Electrons released
Bipolar read-only memory	Fuse blown	Fuse intact
Magnetic tape or disk	Flux direction N	Flux direction S
Read-only compact disc	No pit	Pit
Writeable compact disc (CD-R)	Dye in crystalline state	Dye in noncrystalline state

Relating a Logic Gate Symbol, Circuit, Truth Table, and Timing Diagram



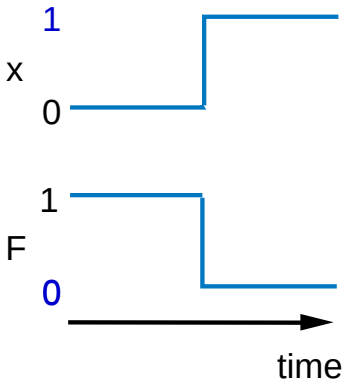
Symbol



Transistor circuit

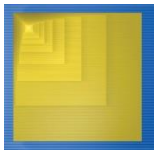
x	F
0	1
1	**0**

Truth table

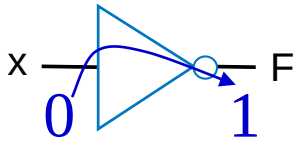


Timing diagram

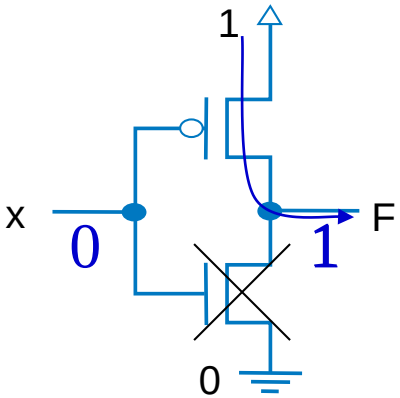
Setting x to 1 causes F to be 0



Relating a Logic Gate Symbol, Circuit, Truth Table, and Timing Diagram



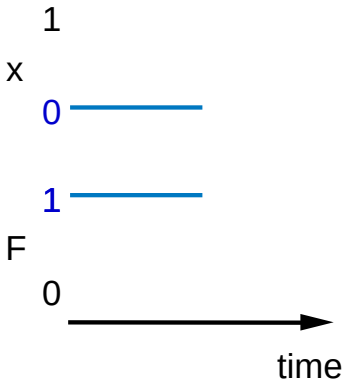
Symbol



Transistor circuit

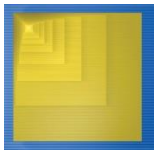
x	F
0	**1**
1	0

Truth table



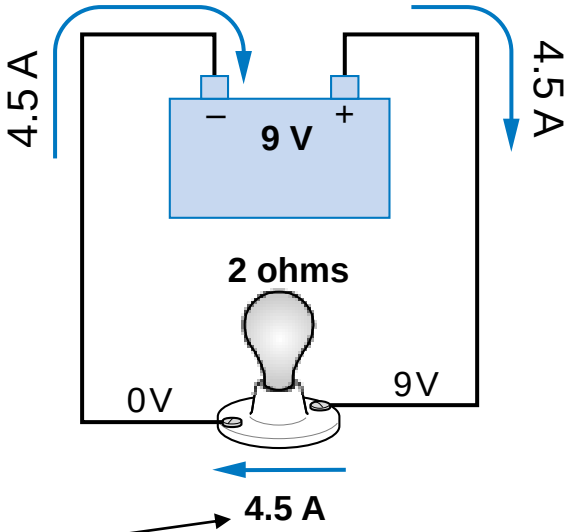
Timing diagram

Setting x to 0 causes F to be 1



Switches

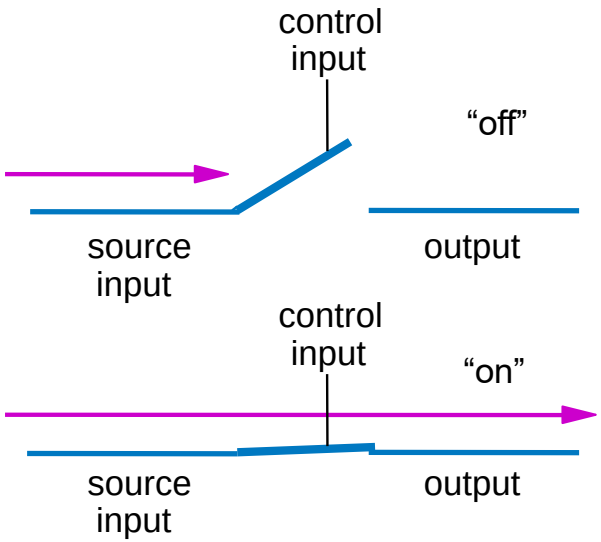
- Electronic switches are the basis of binary digital circuits
  - Electrical terminology
    - ***Voltage***: Difference in electric potential between two points (volts, V)
      - Analogous to water pressure
    - ***Resistance***: Tendency of wire to resist current flow (ohms, Ω)
      - Analogous to water pipe diameter
    - ***Current***: Flow of charged particles (amps, A)
      - Analogous to water flow
    - V = I * R (Ohm's Law)
      - 9 V = I * 2 ohms
      - I = 4.5 A

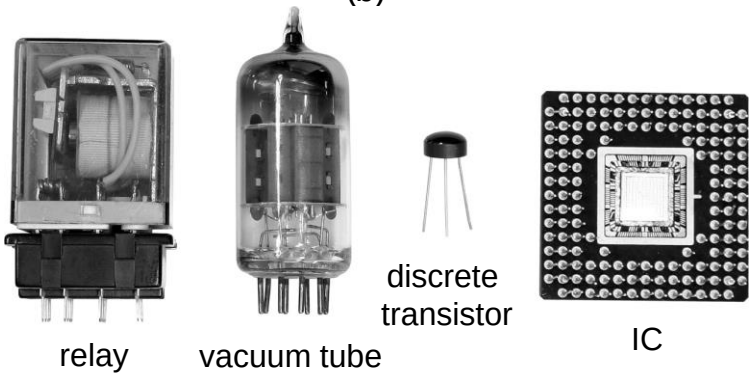


If a 9V potential difference is applied across a 2 ohm resistor, then 4.5 A of current will flow.

Switches

- A switch has three parts
  - Source input, and output
    - Current tries to flow from source input to output
  - Control input
    - Voltage that controls whether that current can flow
- The amazing shrinking switch
  - 1930s: Relays
  - 1940s: Vacuum tubes
  - 1950s: Discrete transistor
  - 1960s: Integrated circuits (ICs)
    - Initially just a few transistors on IC
    - Then tens, hundreds, thousands...

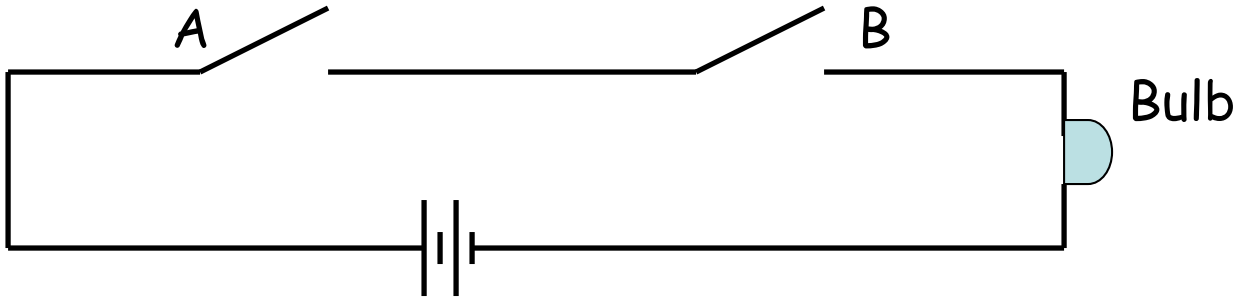




quarter
(to see the relative size)

Logic gates

As a first example consider the following lighting circuit with two switches A and B



Q. Under what conditions will the bulb light ?

A.

The bulb will light
ONLY when both switches A AND B are closed. This can be represented in a **Truth Table**.

Switch A	Switch B	Output Y
Open	Open	NO
Open	Closed	NO
Closed	Open	NO
Closed	Closed	YES

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

In binary logic we will denote a zero or low voltage by a digital 0 and a high voltage by a digital 1.

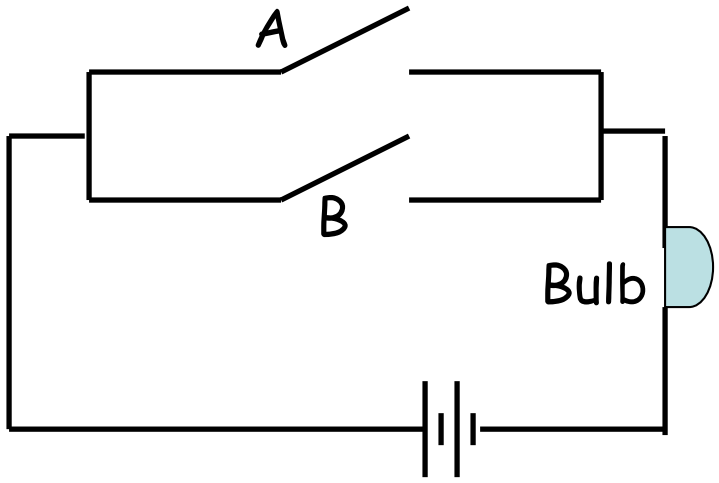
This type of truth table is called an **AND** table.

2 Input AND truth table. 1 output

Symbolically (Boolean) it is written as
$Y = A \bullet B$ and is read as

'A AND B gives the output Y'

Here is another possible arrangement for the two switches.



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

In this arrangement the bulb can light if either switch A or B is closed.

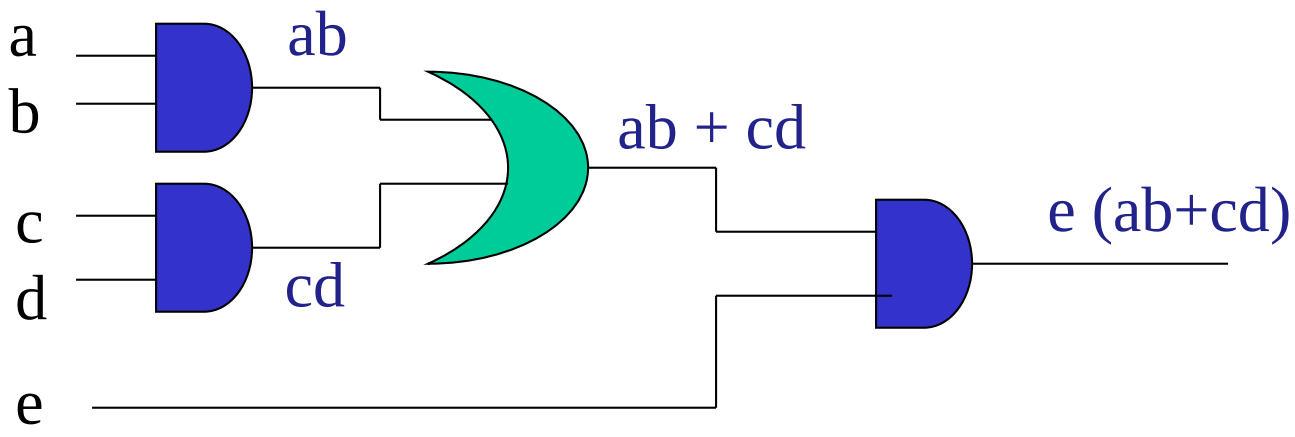
This is an example of a circuit based on an OR gate.

The 2 input OR table shows that if one or both inputs A and B are digital 1, the output is a digital 1.

The Boolean expression for an OR gate is $Y = A + B$ and is read as

'Y is the output of A OR B'.

Combinational Logic vs Boolean Algebra



Schematic Diagram:

5 primary inputs

4 components

9 signal nets

12 pins

Boolean Algebra:

5 literals

4 operators

Digital Design: Switches

A switch connects two points under control signal.

Normally Open

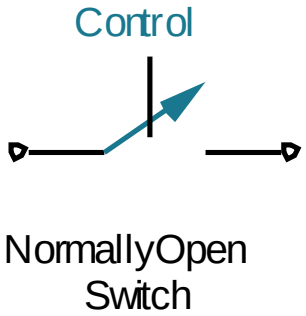
when the control signal is 0 (false), the switch is open

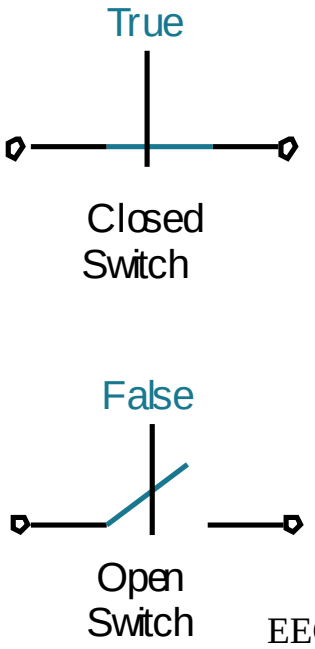
when it is 1 (true), the switch is closed

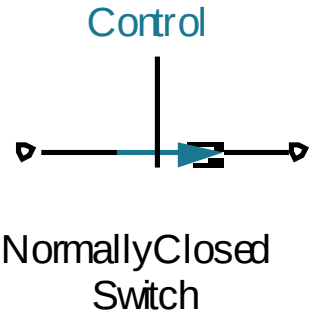
Normally Closed

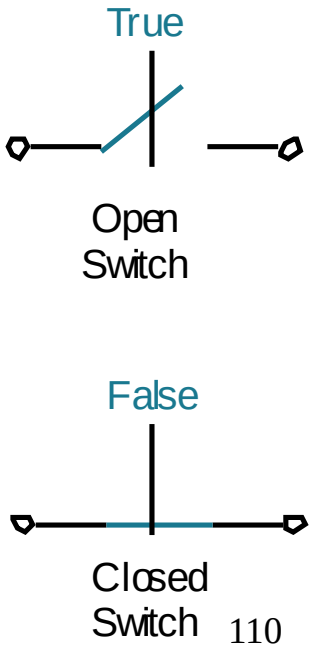
when control is 1 (true), switch is open

when control is 0 (false), switch is closed



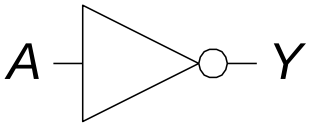






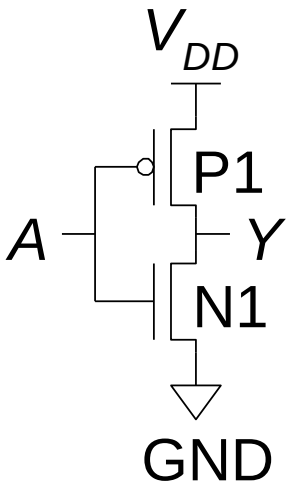
CMOS Gates: NOT Gate

NOT



$Y = \overline{A}$

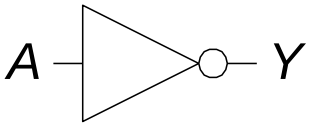
A	Y
0	1
1	0



A	P1	N1	Y
0			
1			

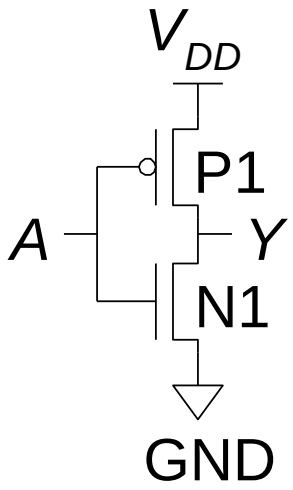
CMOS Gates: NOT Gate

NOT



$Y = \overline{A}$

A	Y
0	1
1	0



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

Power Consumption

- Power = Energy consumed per unit time
  - Dynamic power consumption
  - Static power consumption

Dynamic Power Consumption

- **Power to charge transistor gate capacitances**
  - Energy required to charge a capacitance, C, to V_{DD} is CV_{DD}^2
  - Circuit running at frequency f: transistors switch (from 1 to 0 or vice versa) at that frequency
  - Capacitor is charged $f/2$ times per second (discharging from 1 to 0 is free)
- Dynamic power consumption:

$$P_{dynamic} = \tfrac{1}{2}CV_{DD}^2 f$$

Static Power Consumption

- Power consumed when no gates are switching
- Caused by the *quiescent supply current*, I_{DD} (also called the *leakage current*)
- Static power consumption:

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

Power Consumption Example

- Estimate the power consumption of a wireless handheld computer
  - V_{DD} = 1.2 V
  - C = 20 nF
  - f = 1 GHz
  - I_{DD} = 20 mA

Power Consumption Example

- Estimate the power consumption of a wireless handheld computer
  - V_{DD} = 1.2 V
  - C = 20 nF
  - f = 1 GHz
  - I_{DD} = 20 mA

$$P = \tfrac{1}{2}CV_{DD}^2 f + I_{DD}V_{DD}$$

$$= \tfrac{1}{2}(20\text{ nF})(1.2\text{ V})^2(1\text{ GHz}) + (20\text{ mA})(1.2\text{ V})$$

$$= (14.4 + 0.024)\text{ W} \approx 14.4\text{ W}$$

Chapter Summary

- Digital systems surround us
  - Inside computers
  - Inside many other electronic devices (embedded systems)
- Digital systems use 0s and 1s
  - Encoding analog signals to digital can provide many benefits
    - e.g., audio—higher-quality storage/transmission, compression, etc.
  - Encoding integers as 0s and 1s: Binary numbers
- Microprocessors (themselves digital) can implement many digital systems easily and inexpensively
  - But often not good enough—need custom digital circuits