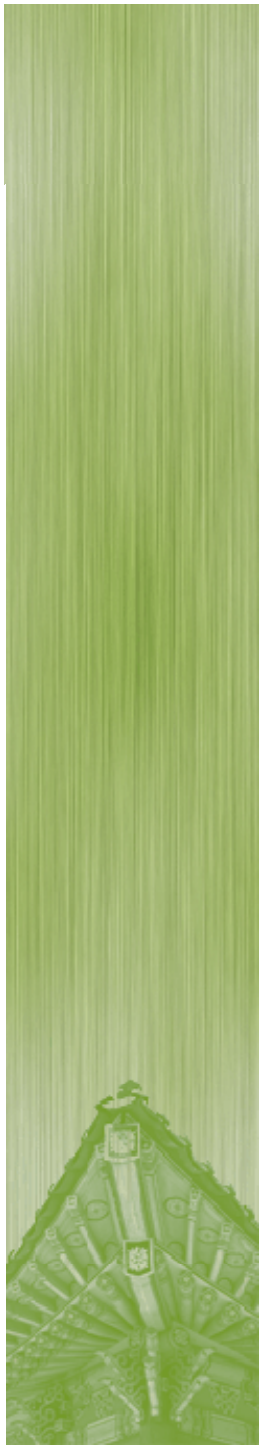
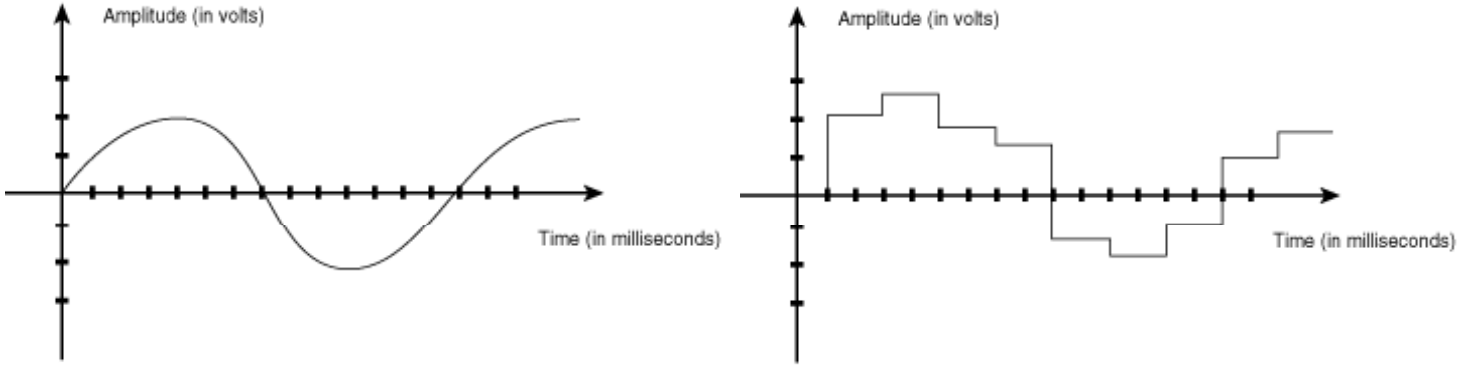


Digital Systems

Jin-Soo Kim (jinsookim@skku.edu)
Computer Systems Laboratory
Sungkyunkwan University
http://csl.skku.edu

Introduction

- **The advent of the digital age**
  - Analog vs. digital?



  - Compact Disc (CD)
    - 44.1 KHz, 16-bit, 2-channel
  - MP3
    - A digital audio encoding with lossy data compression

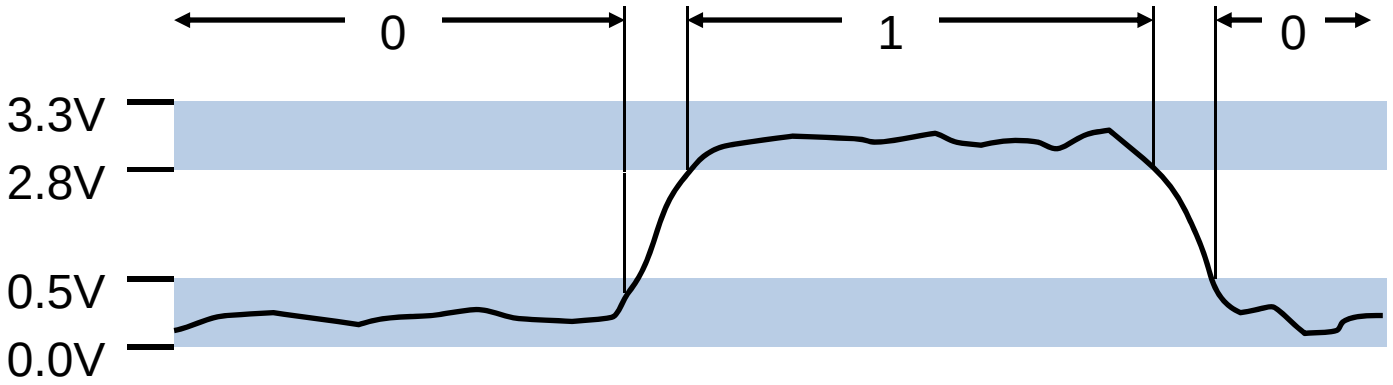
Representing Information

- **Information = Bits + Context**
  - Computers manipulate representations of things.
  - Things are represented as binary digits.
  - What can you represent with N bits?
    - 2^N things
    - Numbers, characters, pixels, positions, source code, executable files, machine instructions, …
    - Depends on what operations you do on them.

	01110011 01101011 01101011 01110101 01110011 01100101 01101101 01101001							
(char)	**'s'**	**'k'**	**'k'**	**'u'**	**'s'**	**'e'**	**'m'**	**'i'**
(int)	**1969974131**				**1768777075**			
(double)	**7.031689903291708081 78… x 10^{199}**							

Binary Representations

- **Why not base 10 representation?**
  - Easy to store with bistable elements
  - Straightforward implementation of arithmetic functions
  - Reliably transmitted on noisy and inaccurate wires
- **Electronic implementation**



Encoding Byte Values

- **Byte = 8 bits**
  - Binary: 00000000_2 to 11111111_2
  - Octal: 000_8 to 377_8
    - An integer constant that begins with `0` is an octal number in C
  - Decimal: 0_{10} to 255_{10}
    - First digit must not be `0` in C
  - Hexadecimal: 00_{16} to FF_{16}
    - Base 16 number representation
    - Use characters '`0`' to '`9`' and '`A`' to '`F`'
    - Write $FA1D37B_{16}$ in C as **`0xFA1D37B`** or **`0xfa1d37b`**

Hex	Decimal	Binary
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

Boolean Algebra (1)

- **Developed by George Boole in 1849**
  - Algebraic representation of logic
    - Encode "True" as 1 and "False" as 0

- **And**
  - A&B = 1 when both A=1 and B=1

&	0	1
0	0	0
1	0	1

- **Or**
  - A|B = 1 when either A=1 or B=1

|	0	1
0	0	1
1	1	1

- **Not**
  - ~A = 1 when A=0

~	
0	1
1	0

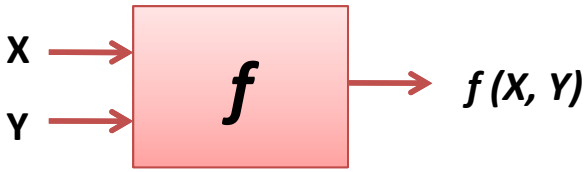
- **Exclusive-Or (Xor)**
  - A^B = 1 when either A=1 or B=1, but not both

^	0	1
0	0	1
1	1	0

Boolean Algebra (2)

0	0	1	1	X
0	1	0	1	Y

0	0	0	0	Constant 0
0	0	0	1	X & Y ; AND
0	0	1	0	~ (X → Y)
0	0	1	1	X
0	1	0	0	~ (Y → X)
0	1	0	1	Y
0	1	1	0	X ^ Y ; XOR
0	1	1	1	X | Y ; OR
1	0	0	0	~ (X | Y) ; NOR
1	0	0	1	~ (X ^ Y) ; X-NOR
1	0	1	0	~ Y
1	0	1	1	Y → X
1	1	0	0	~ X
1	1	0	1	X → Y
1	1	1	0	~ (X & Y) ; NAND
1	1	1	1	Constant 1

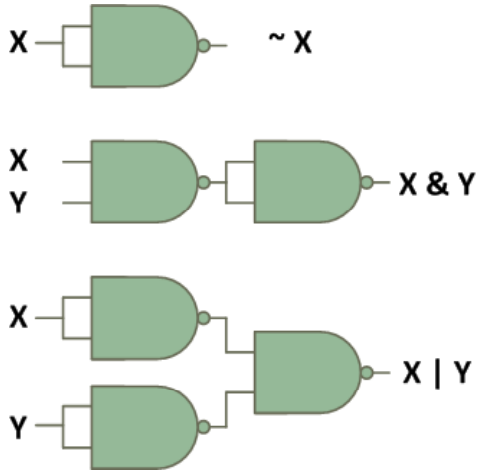


Basic operations: AND(&), OR(|), NOT(~)

X ^ Y = (X & ~Y) | (~X & Y)

X→Y = ~X | Y

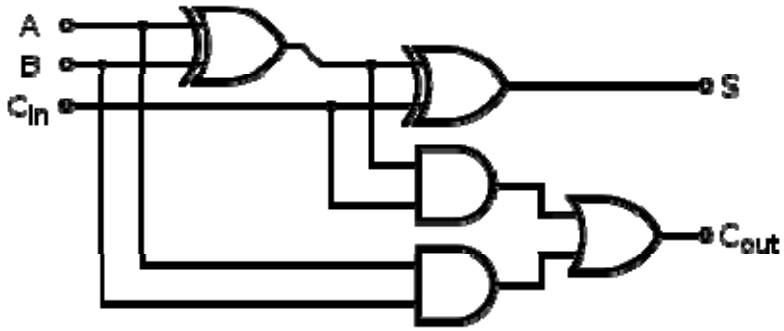
A complete set: NAND = ~ (X & Y)

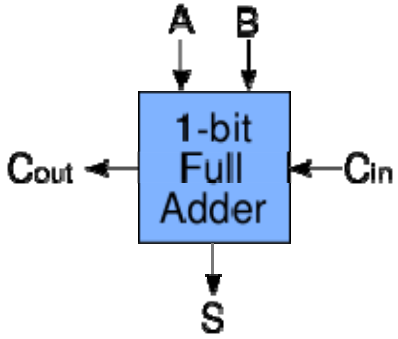


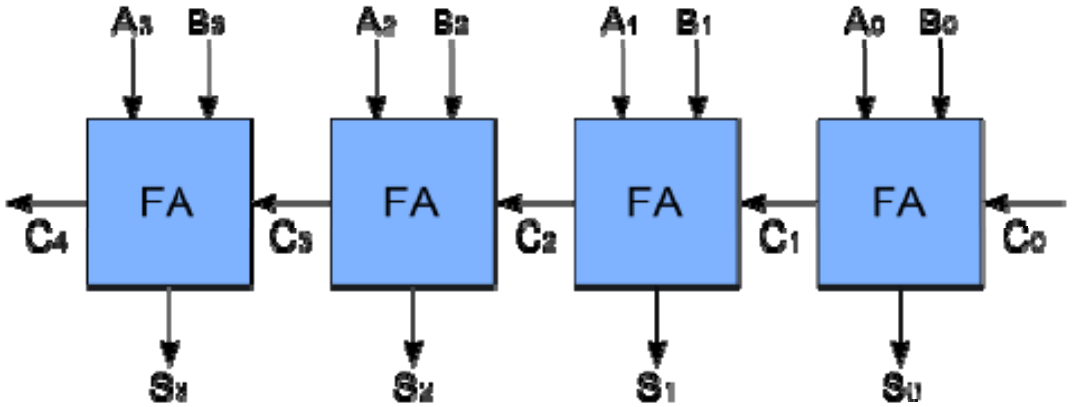
Combinational Logic

- **Adder**

Full Adder



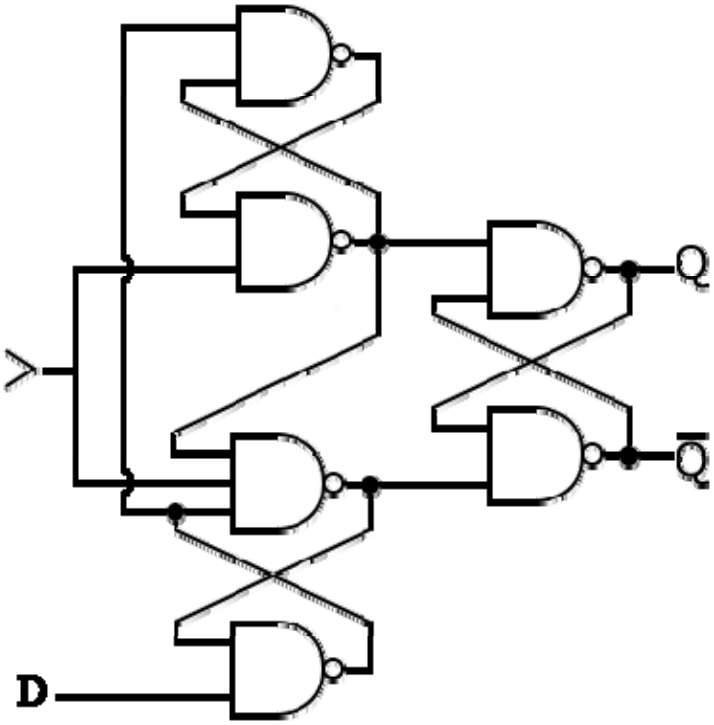




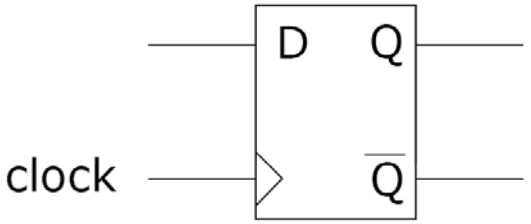
4-bit Ripple Carry Adder

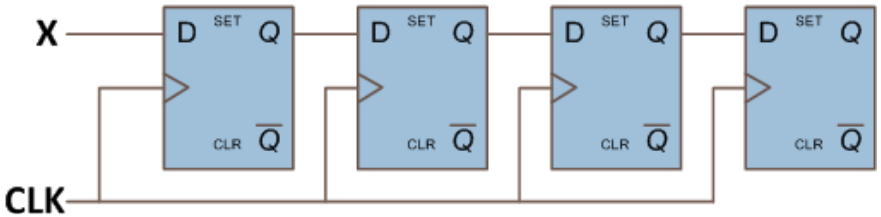
Sequential Logic

- **Flip-flops**

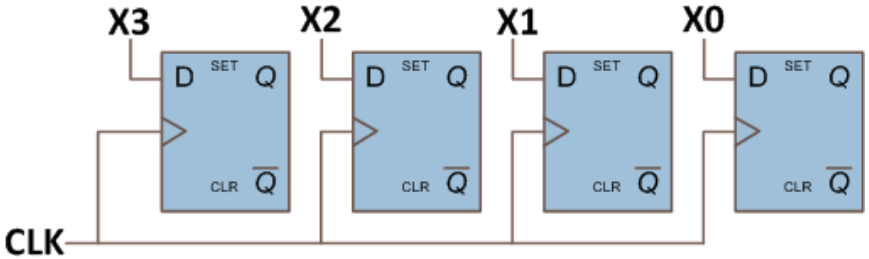


Edge triggered D flip-flop





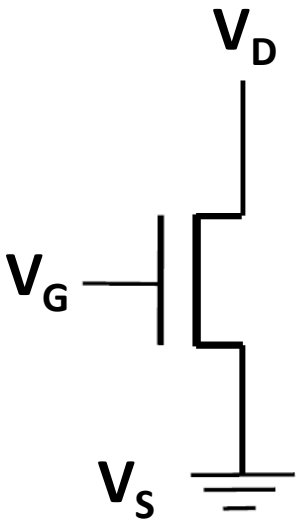
Shifter

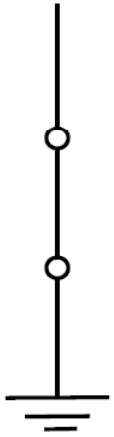


4-bit register

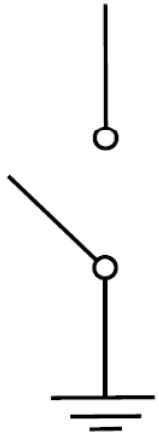
Transistors (1)

- **Transistor = Electronic switch**
  - Controlled by voltages
    - e.g., Logic 1 = 5V, Logic 0 = 0V
  - NMOS transistor





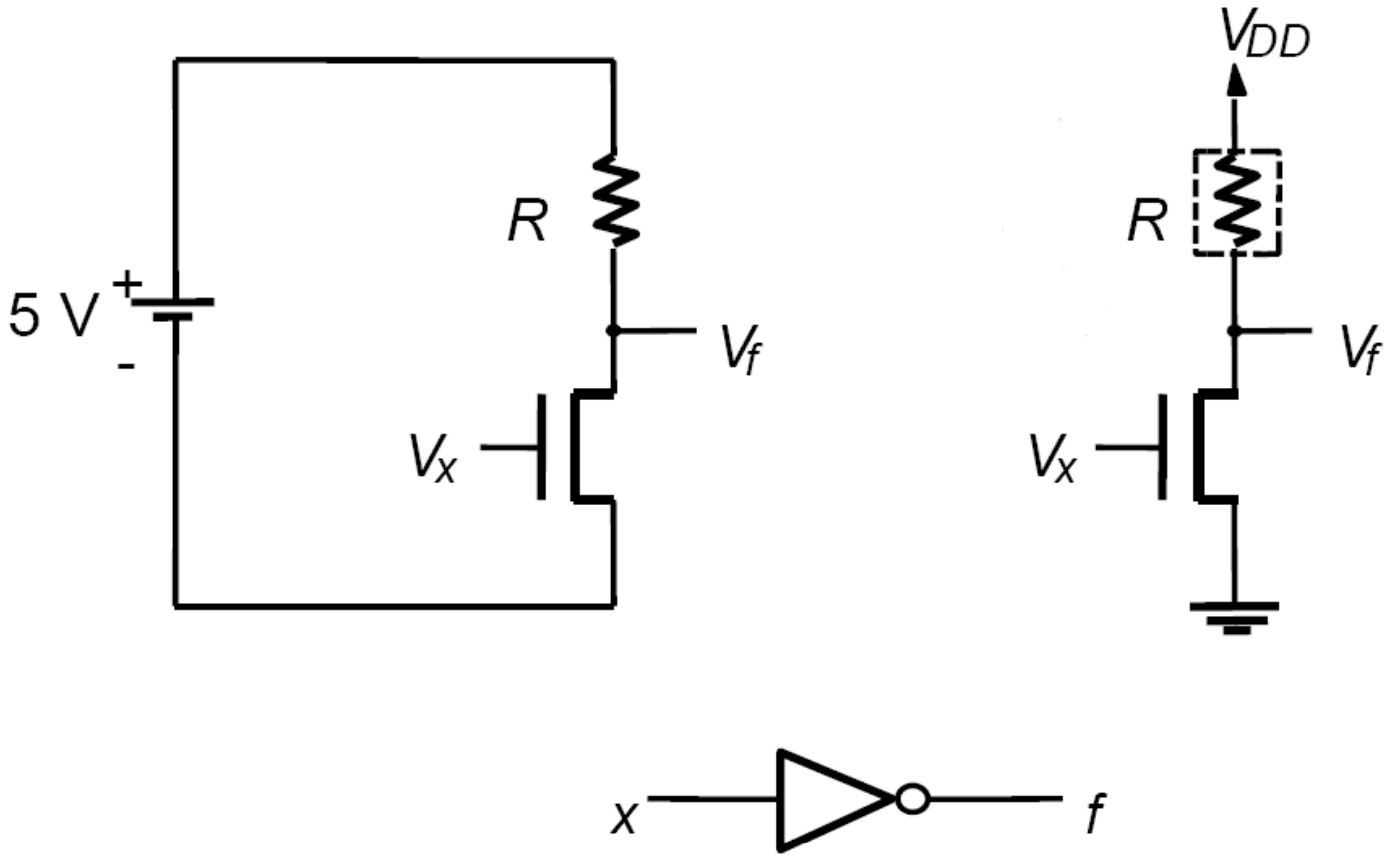
Closed switch
(V_G = 5V)



Open switch
(V_G = 0V)

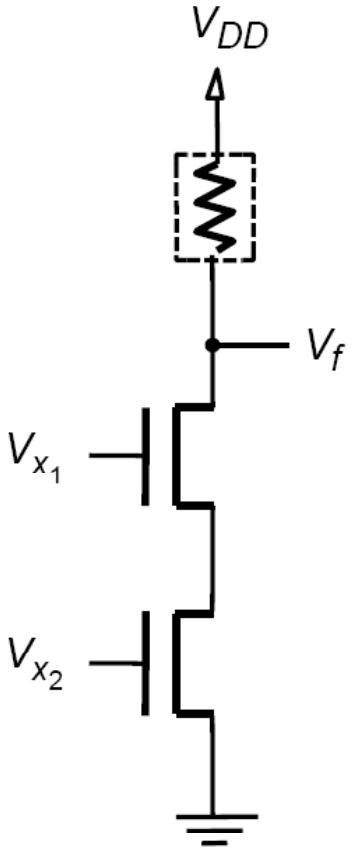
Transistors (2)

- **NOT logic built with NMOS technology**



Transistors (3)

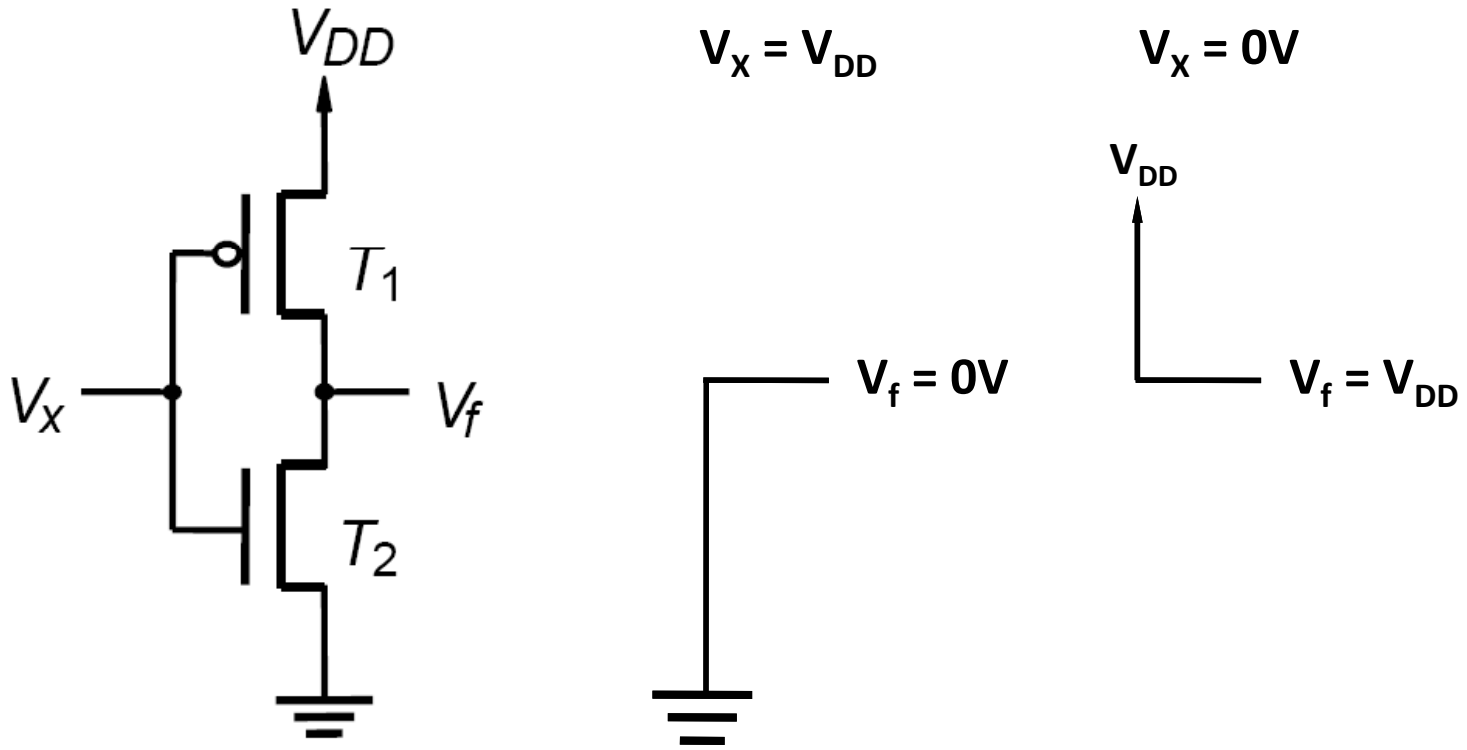
- NAND logic built with NMOS technology



x_1	x_2	f
0	0	1
0	1	1
1	0	1
1	1	0

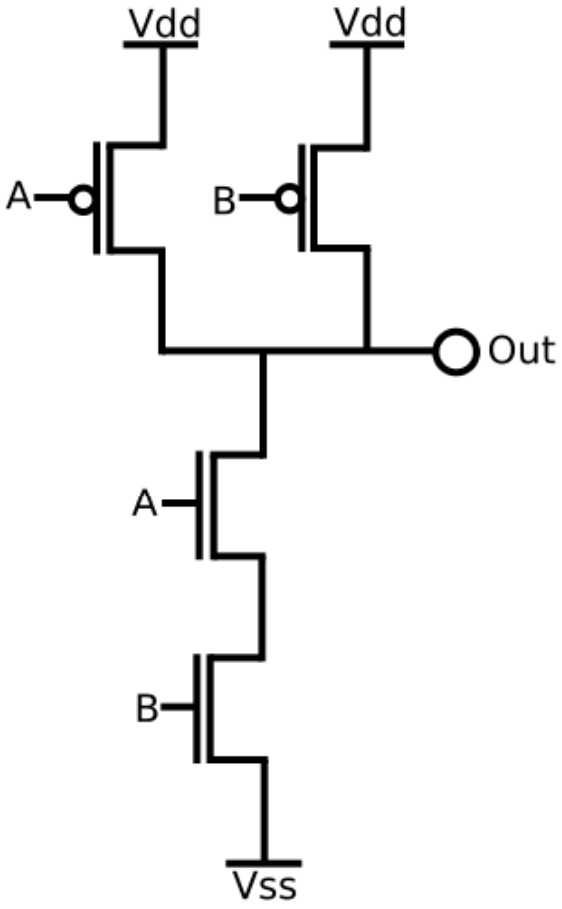
Transistors (4)

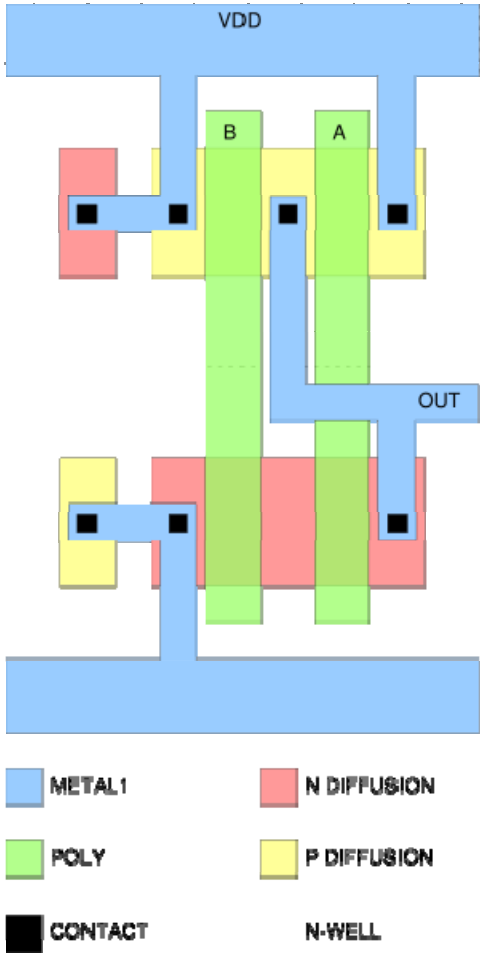
- NOT logic built with CMOS technology



Transistors (5)

- NAND logic built with CMOS technology





Digital Systems

- **Summary**
  - Boolean algebra is a mathematical foundation for modern digital systems
  - Boolean algebra provides an effective means of describing circuits built with switches
    - Claude Shannon in the late 1930's.
  - You can build any digital systems with NAND gates
  - A NAND gate can be easily built with CMOS transistors
  - The transistor is the basic building block for digital systems