Introduction to Computer Architecture

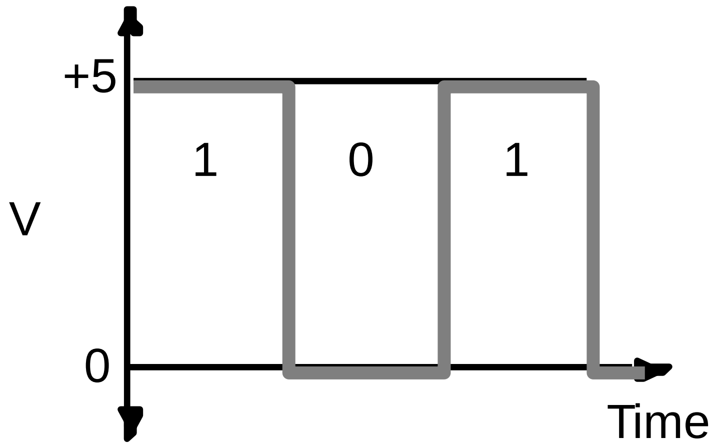
# Digital Logic Circuits

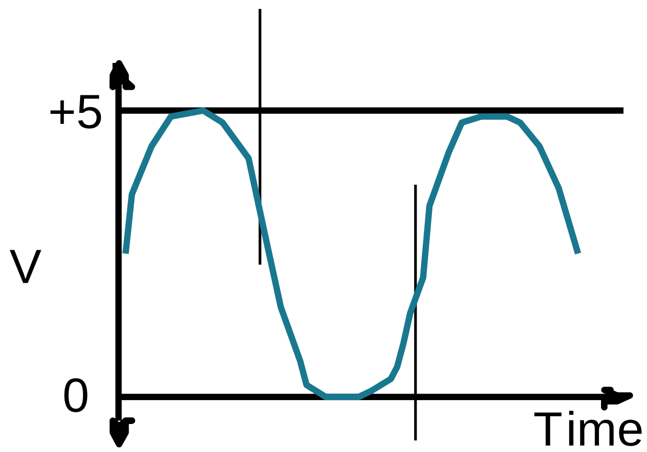

Hyungmin Cho

Department of Computer Science and Engineering
Sungkyunkwan University

# Digital Signals

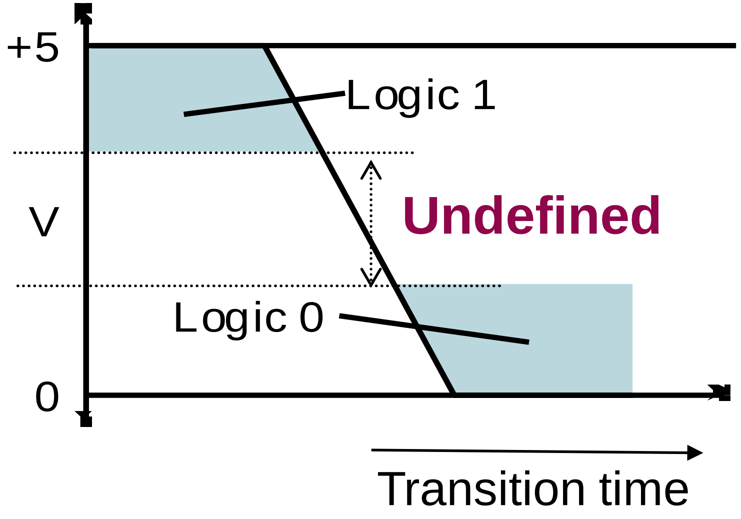
- Digital vs. Analog Waveforms



# Digital Circuit

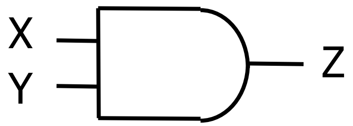
- A signal above a certain voltage level: “1” (True)
- A signal below a certain voltage level: “0” (False)



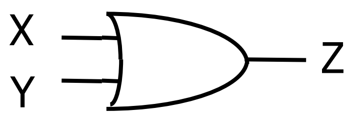

- Different components in a computer have different voltage levels
  - CPU (Core 2 Duo): 1.325 V
  - Chipsets: 1.45 V
  - Peripheral devices: 3.3V, 1.5V

# Boolean Algebra and Logical Operations

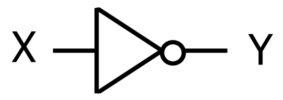
| X | Y | X AND Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |



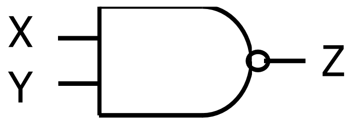

| X | Y | X OR Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |



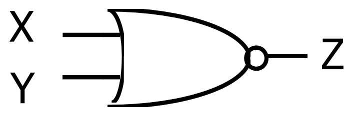

| X | NOT X |
|---|---|
| 0 | 1 |
| 1 | 0 |



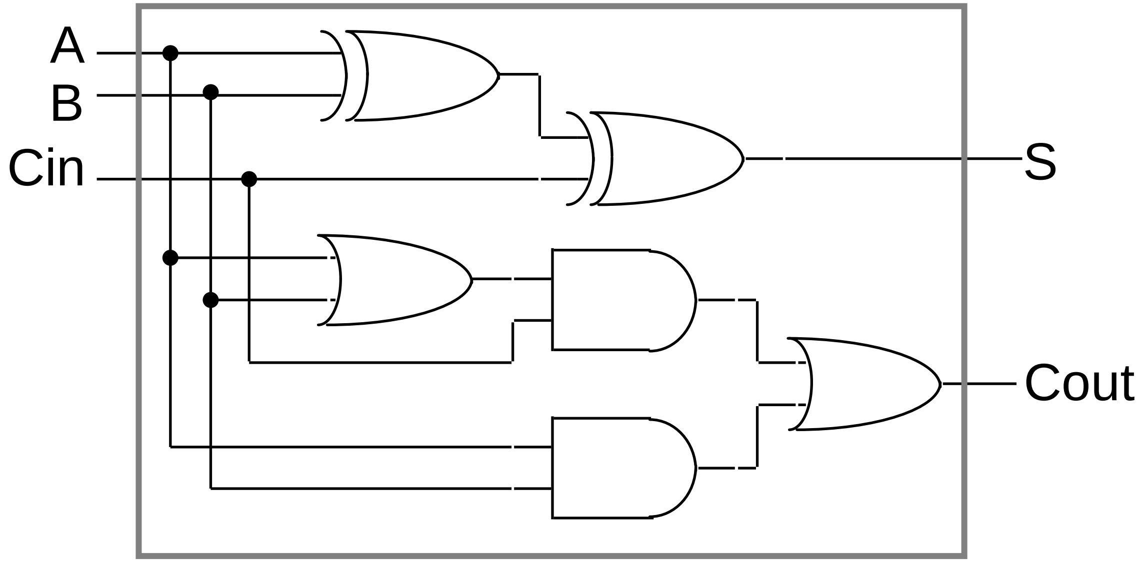

| X | Y | X NAND Y |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |




| X | Y | X NOR Y |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

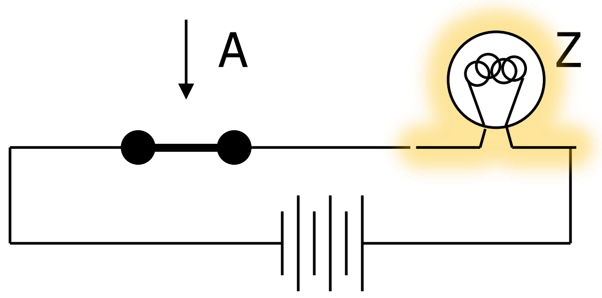


# Digital Logic Circuit Example

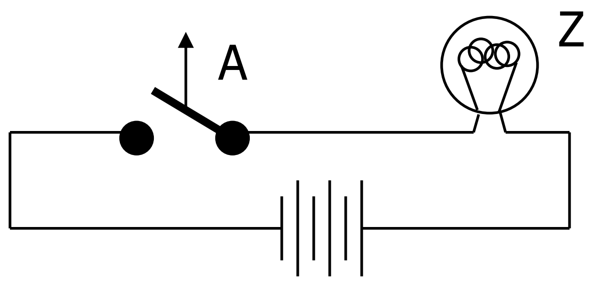


# Switches: Basic Element of Physical Implementations

- Implementing a simple circuit



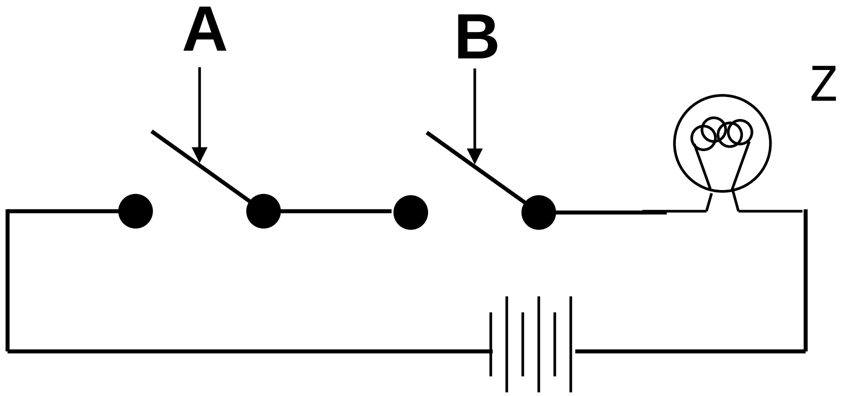

close switch (if A is "1")
and turn on light bulb (Z)



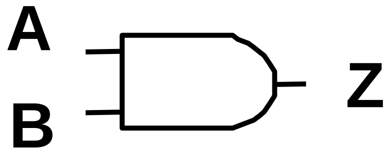

open switch (if A is "0")
and turn off light bulb (Z)

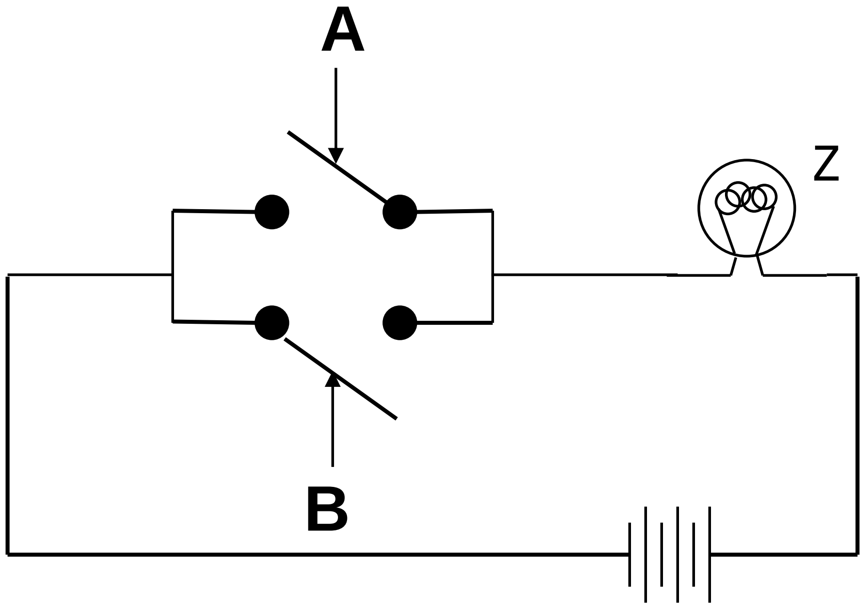
# Switches (cont'd)

- Compose switches into more complex ones (Boolean functions):

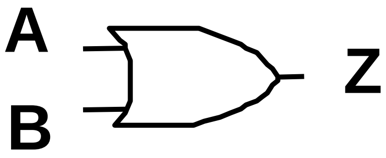



Z = A AND B

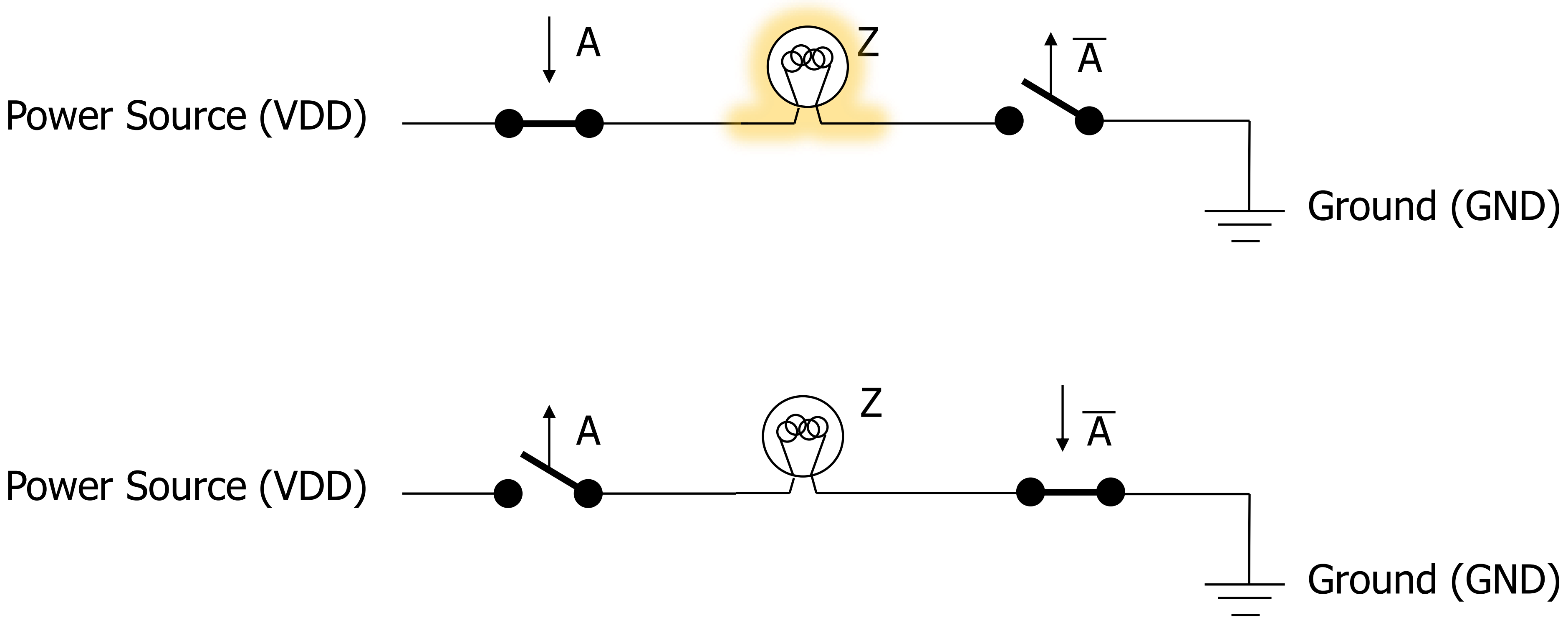



Z = A OR B

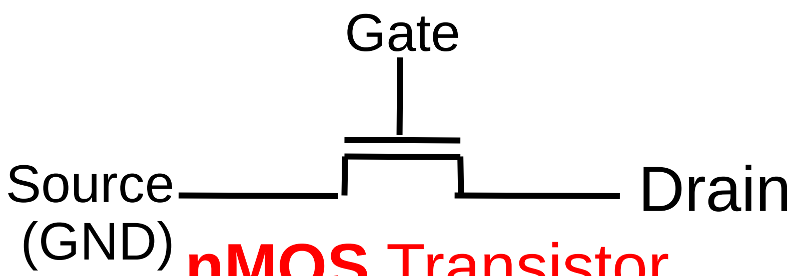


# Semiconductor Switches



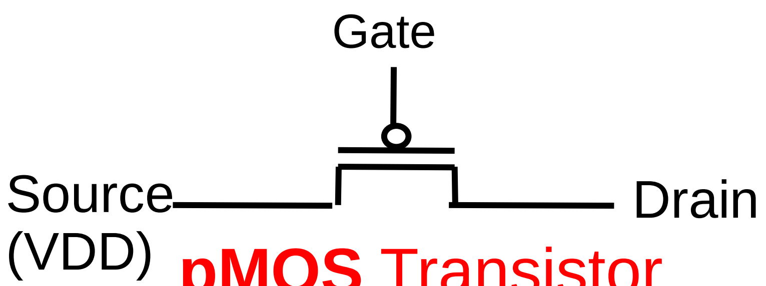
# MOS transistors

- MOS transistors have three terminals: drain, gate, and source




**nMOS** Transistor

Logic 1 on gate:
Source and Drain connected

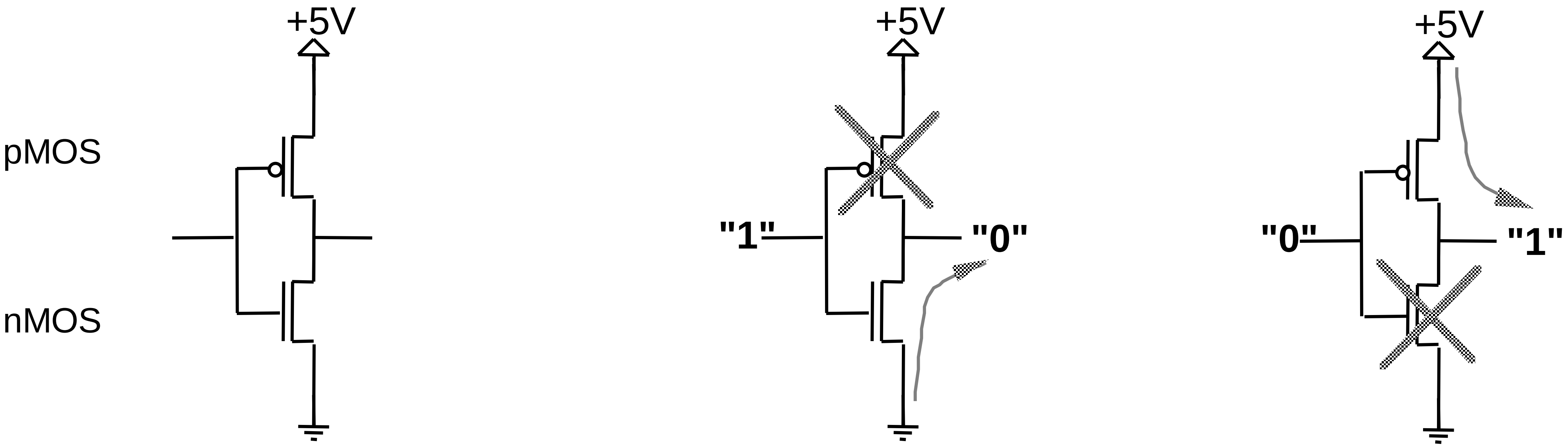



**pMOS** Transistor

Logic 0 on gate
Source and Drain connected

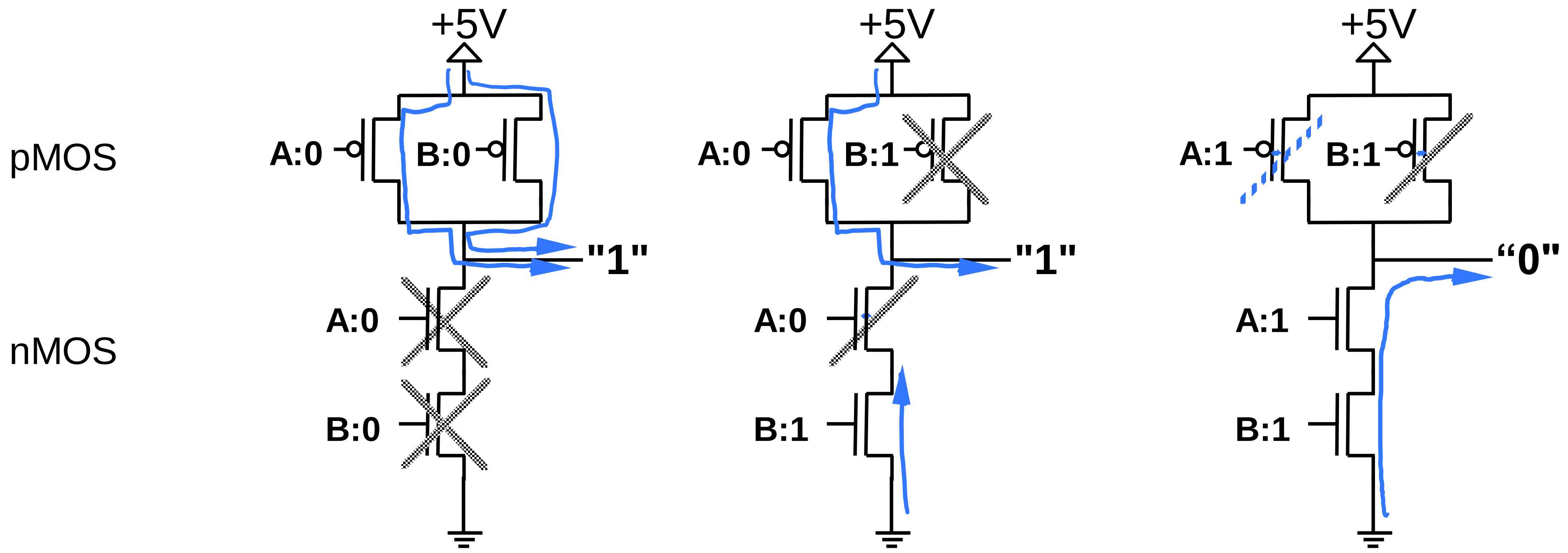
- MOS transistors act as voltage-controlled switches

# Inverter (NOT Gate) Operation



# NAND Operation

| X | Y | X NAND Y |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |



# Number Systems - Binary Number

- **Binary** numbers
  - Bit represents one of 2 values: 0 or 1
  - Each digit of a binary number has 2x the weight of the previous digit
    - ex) $10110_2 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 22_{10}$
  - N-bit binary number represents one of $2^N$ possibilities

    ex) 3-bit binary number represents one of 8 possibilities: 0 ~ 7

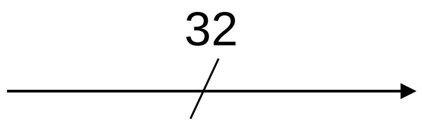
# 32-Bit Binary Number

- 32bit positive integer

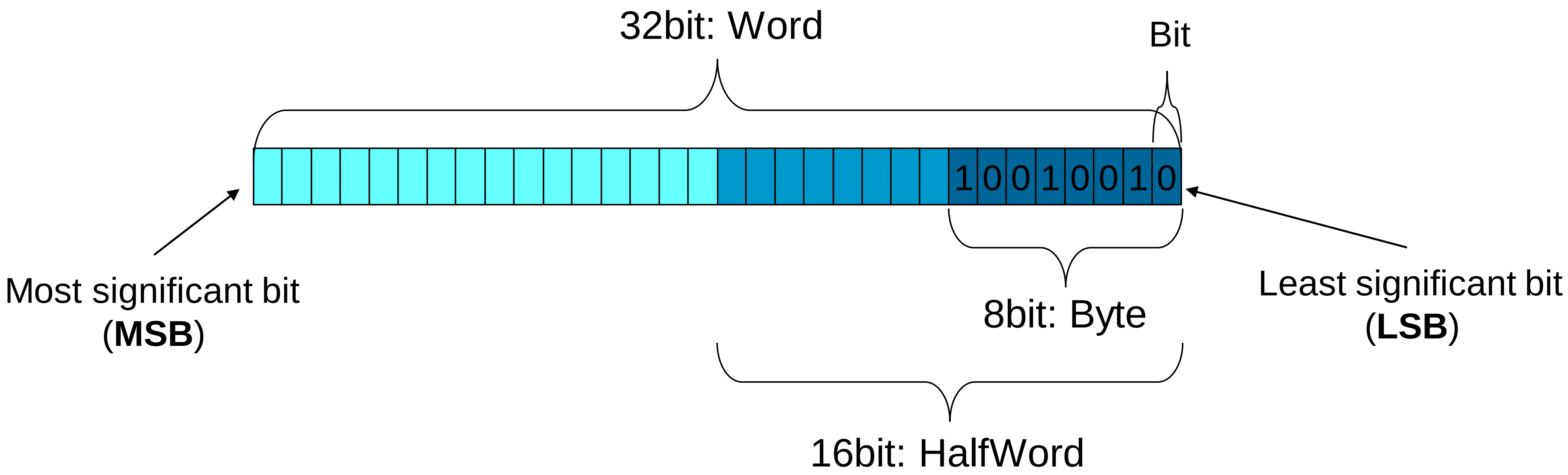
| | 31 | 30 | 29 | 28 | … | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A: | **1** | 0 | **1** | 0 | … | 0 | 0 | 0 | 0 | **1** | 0 | **1** | 0 |

$$2^{31} + 2^{29} + 2^{3} + 2^{1}$$
$$= 2{,}147{,}483{,}648 + 536{,}870{,}912 + 8 + 2 = 2{,}684{,}354{,}570$$

- Notation for a 32-bit wide data: A[31:0]



# Bits, Bytes


32bit: Word
Bit
1 0 0 1 0 0 1 0
Most significant bit (MSB)
8bit: Byte
Least significant bit (LSB)
16bit: HalfWord

# Hexadecimal

- Base 16
  - Compact representation of bit strings
  - 4 bits per hex digit

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0000 | 4 | 0100 | 8 | 1000 | C | 1100 |
| 1 | 0001 | 5 | 0101 | 9 | 1001 | D | 1101 |
| 2 | 0010 | 6 | 0110 | A | 1010 | E | 1110 |
| 3 | 0011 | 7 | 0111 | B | 1011 | F | 1111 |

- Example: $\text{ECA86420}_{16}$
  - $\text{1110 1100 1010 1000 0110 0100 0010 0000}_{2}$

- Starts with “0x” prefix: 0xECA96420

# Addition & Overflow

$$
\begin{array}{cccccc}
 & & & 1 & & \\
 & 1 & 0 & 0 & 1 & _2 \\
+ & 0 & 1 & 0 & 1 & _2 \\
\hline
 & 1 & 1 & 1 & 0 & _2
\end{array}
$$

$$
\begin{array}{cc|ccccccccc}
 & 1 & 1 & 1 & & 1 & 1 & & & & \\
 & & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & _2 \\
+ & & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & _2 \\
\hline
 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & _{two}
\end{array}
$$

- 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

# Singed Binary Numbers

- How represent negative integer numbers?

  - Sign / Magnitude
  - Two’s Complement

# Sign / Magnitude Numbers

- 1 sign bit, N-1 magnitude bits (absolute number)
- Sign bit is the MSB (left-most bit)
  - Positive : sign bit is 0
  - Negative : sign bit is 1

- Example: 4-bit representations of +5 and -5:

$$+5 = 0101_2$$

$$-5 = 1101_2$$

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

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

# Sign / Magnitude Number Problems

- Addition doesn't work naturally
- Example: 5 + (-5) = 0 ?

$$
\begin{array}{r|l}
 & 0\ 1\ 0\ 1_2 \\
+ & 1\ 1\ 0\ 1_2 \\
\hline
1 & 0\ 0\ 1\ 0_2
\end{array}
$$

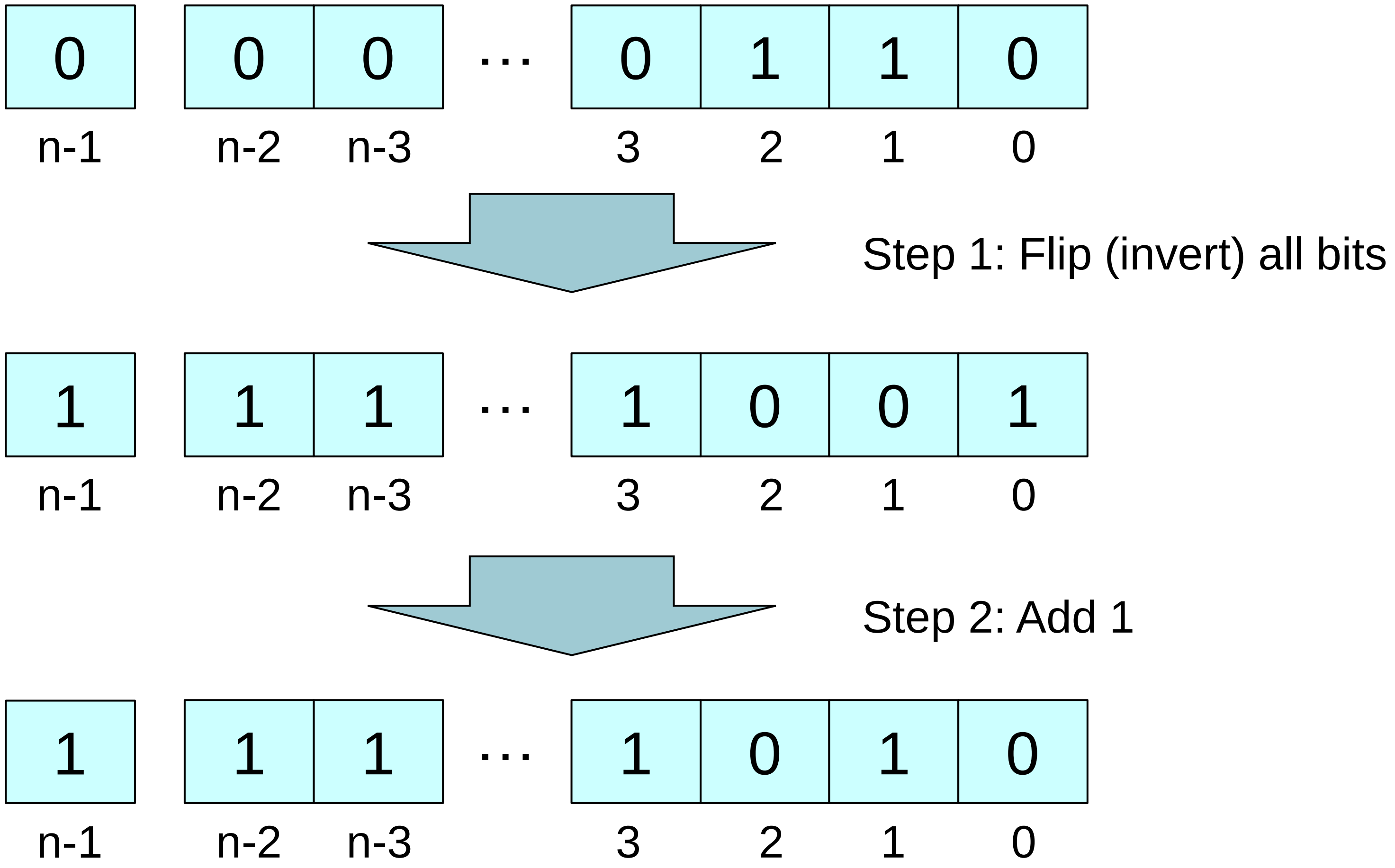
- Two representations of 0 (+0 and -0)

  0000 (+0)

  1000 (-0)

# Two's Complement

- To negate a positive integer value,



# Two's Complement Addition

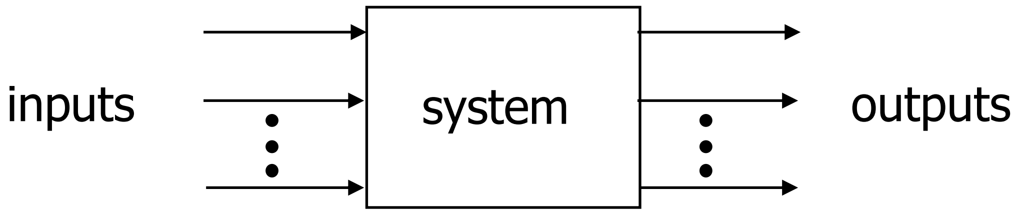
- Example: 5 + (-5) = 0 ?

```
    +5:  0 1 0 1 ₂
Step 1:  1 0 1 0 ₂
Step 2:  1 0 1 1 ₂
```

```
     0 1 0 1 ₂      +5
 +   1 0 1 1 ₂      -5
 ---------------
   1 0 0 0 0 ₂
```

# Combinational vs. sequential circuits

- A simple model of a digital system is a unit with inputs and outputs:

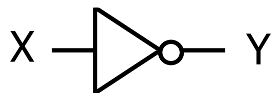



- Combinational means “memory-less”
  - Output values are determined by its current input values

- Sequential means “with memory”
  - Output values depend on the past history of input values

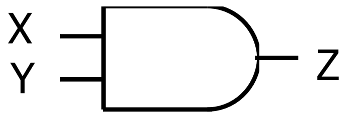
# Combinational Logic - Logic Gates

- NOT: X'



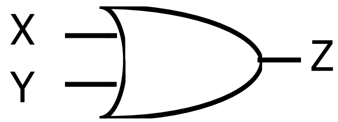

| X | Y |
|---|---|
| 0 | 1 |
| 1 | 0 |

- AND: X • Y




| X | Y | Z |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

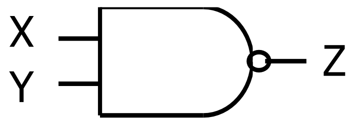
- OR: X + Y




| X | Y | Z |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

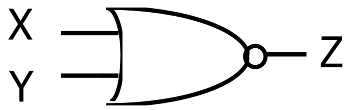
# Combinational Logic - Logic Gates

- NAND




| X | Y | Z |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

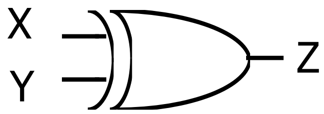
- NOR




| X | Y | Z |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

- XOR

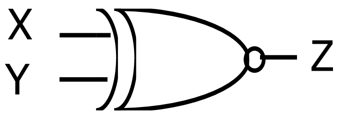
  $X \oplus Y$




| X | Y | Z |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

- XNOR

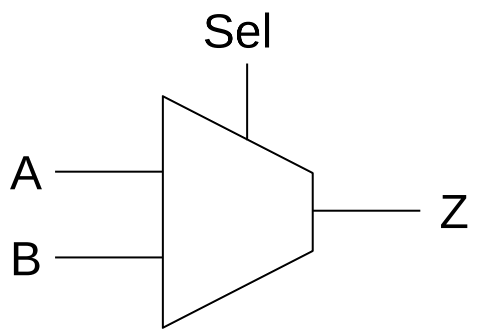
  $X = Y$



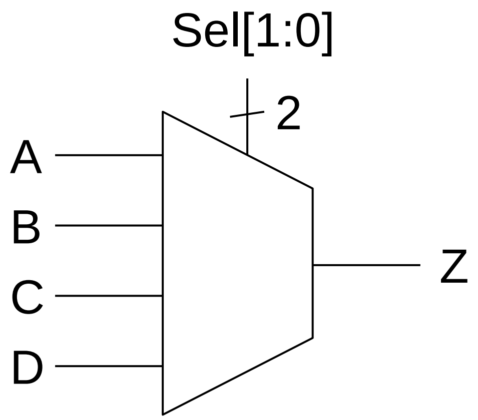

| X | Y | Z |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

# Combinational Logic – Mux / Demux

- Mux



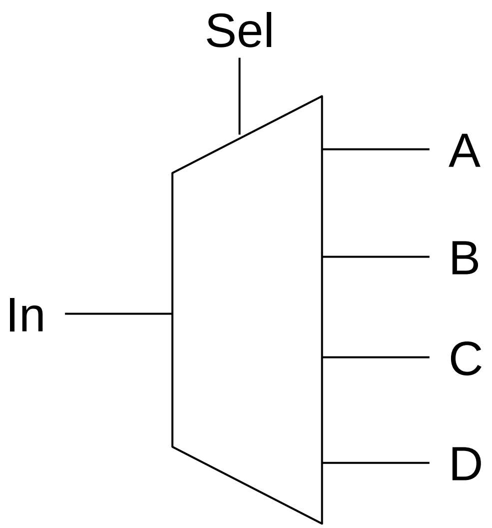

| Sel | Z |
|---|---|
| 0 | A |
| 1 | B |




| Sel | Z |
|---|---|
| 00 | A |
| 01 | B |
| 10 | C |
| 11 | D |

# Combinational Logic – Mux / Demux

- Demux

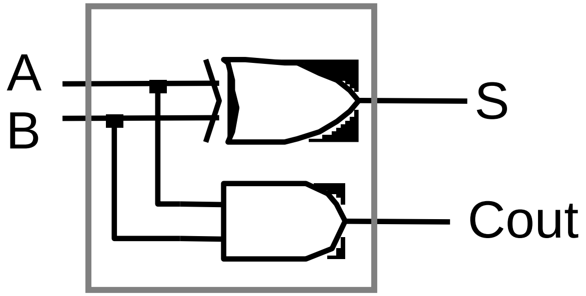



| Sel | A | B | C | D |
|---|---|---|---|---|
| 00 | In | 0 | 0 | 0 |
| 01 | 0 | In | 0 | 0 |
| 10 | 0 | 0 | In | 0 |
| 11 | 0 | 0 | 0 | In |

# Combinational Logic: 1-bit Binary Adder

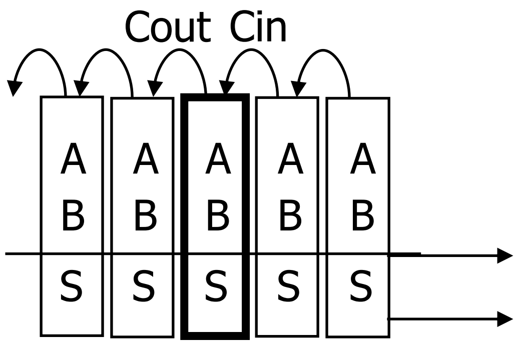
- Inputs: A, B
- Outputs: Sum (S), Carry-out (Cout)

| A | B | Cout | S |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

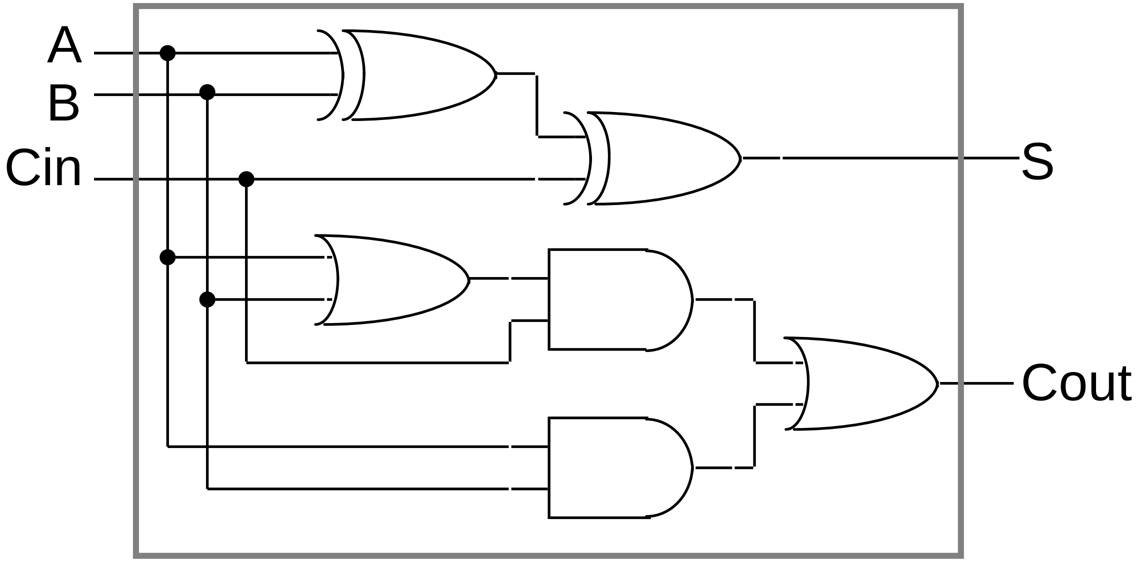


# Combinational Logic: 1-bit Binary Full Adder

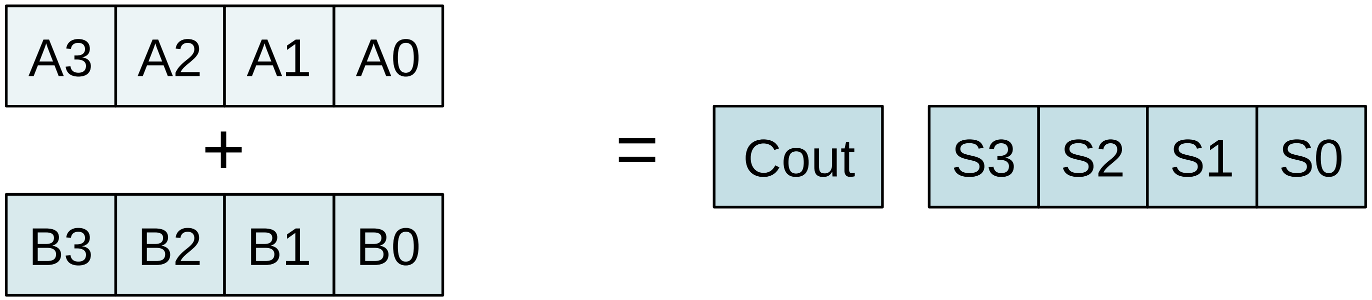
- Inputs: A, B, Carry-in
- Outputs: Sum, Carry-out

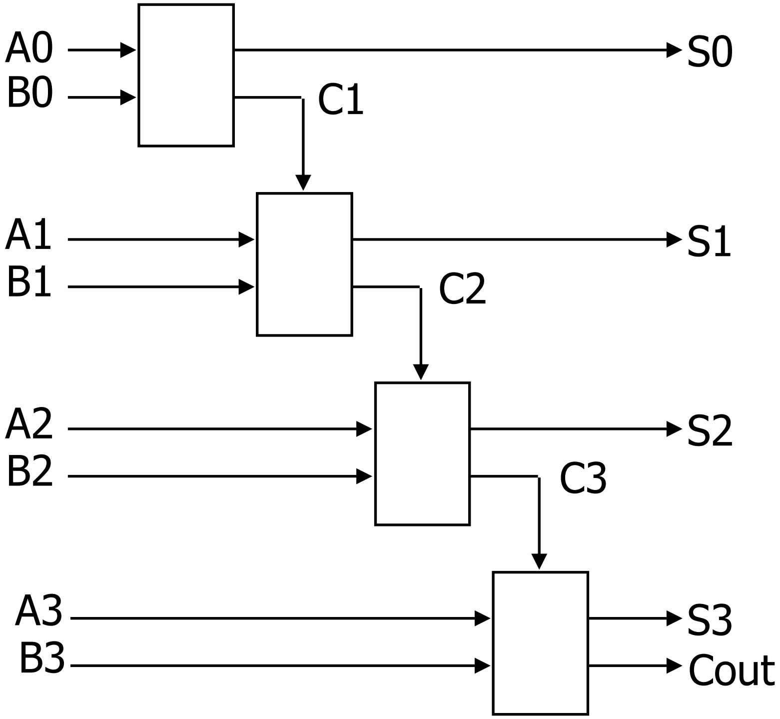



| A | B | Cin | Cout | S |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

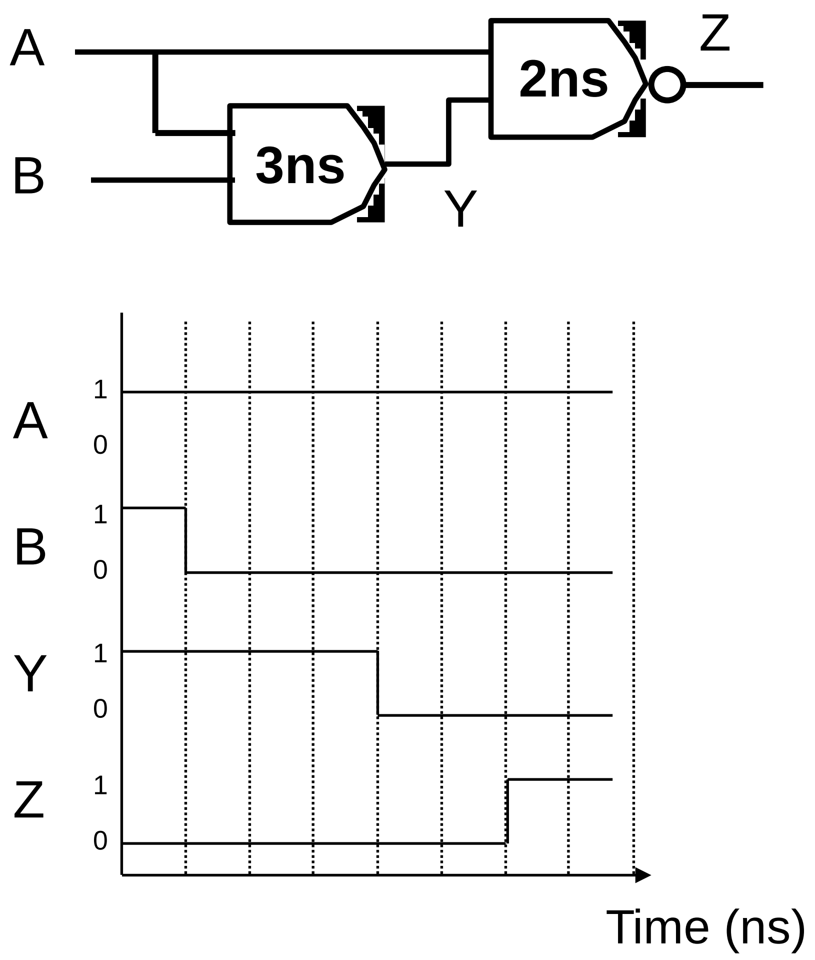


# Ripple Carry Adder



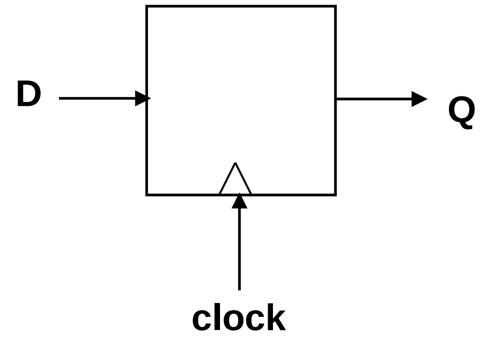
# Pulse Propagation

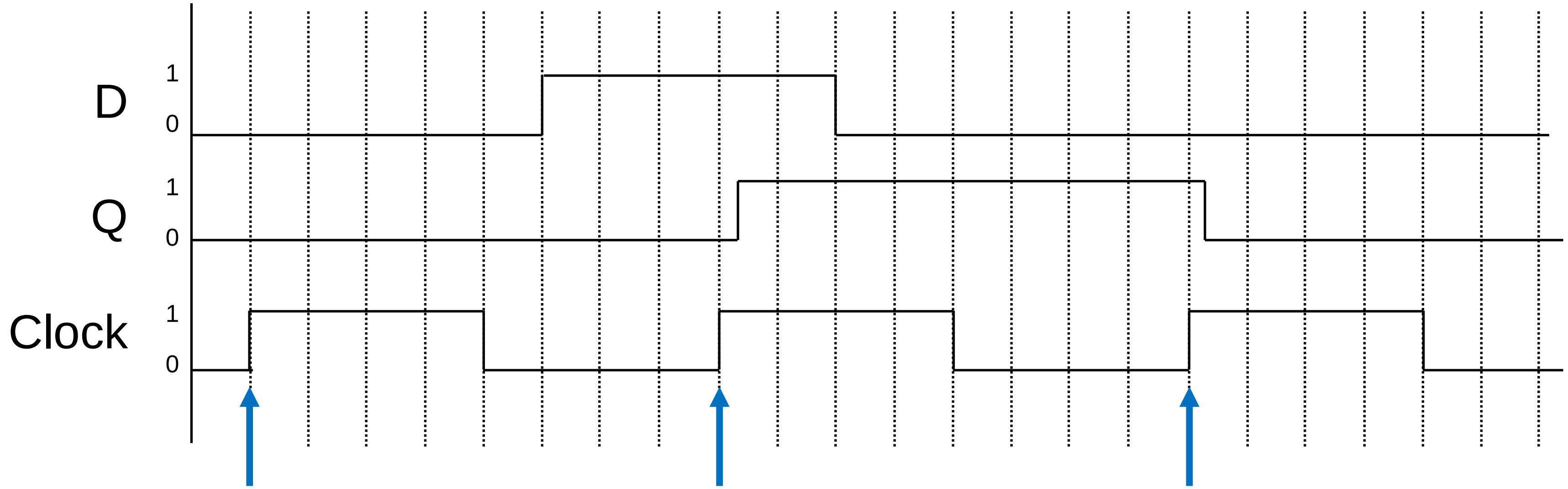



Propagation Delay

# Sequential logic

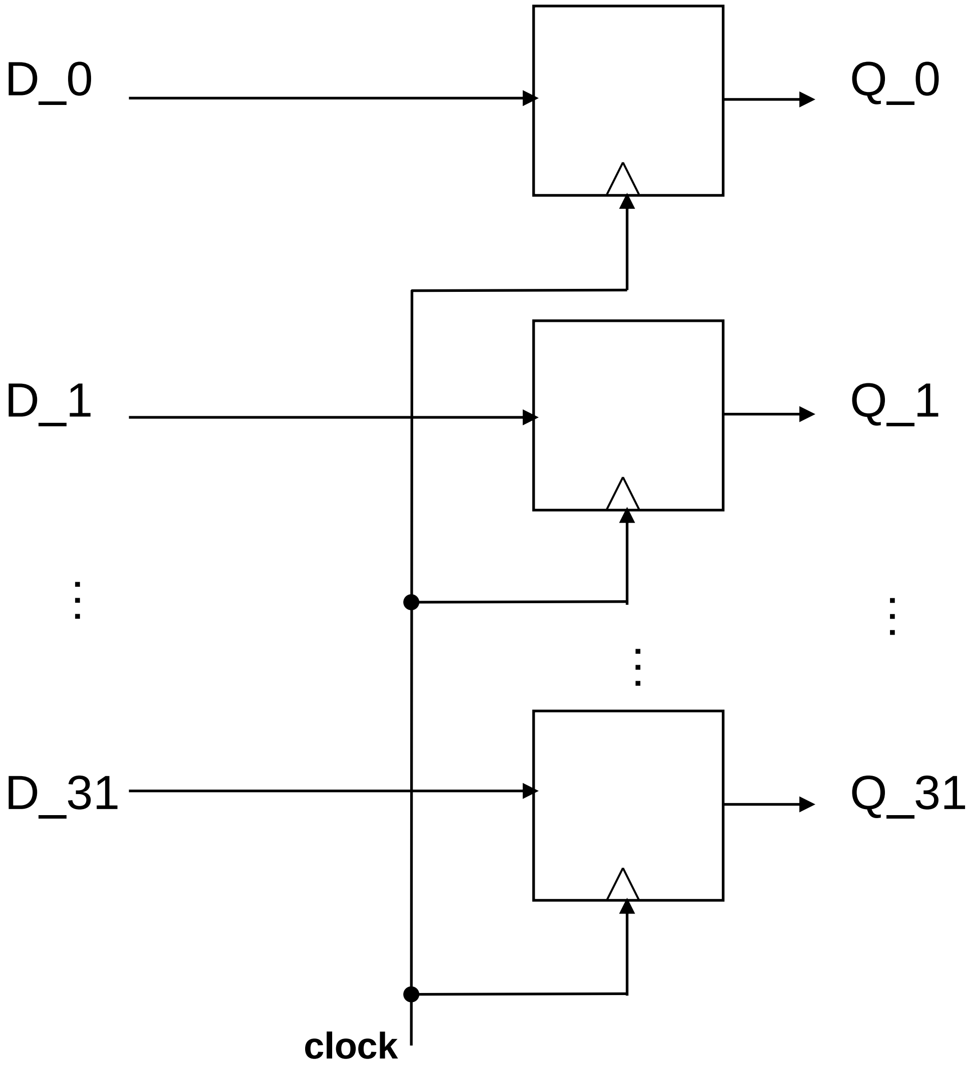
- Storage elements to remember the current state
- Common sequential logic elements are called flip-flops
  - Sense input value on rising clock edge



# Sequential logic

- Multiple flip-flops can used together to represent a multi-bit data



# Sequential logic

- In all real circuits, outputs depend on inputs and the entire history of execution!