

CS 110

Computer Architecture

Digital Circuits and Systems

Instructors:

Chundong Wang, Yuan Xiao & Siting Liu

Course website: [https://toast-](https://toast-lab.sist.shanghaitech.edu.cn/courses/CS110@ShanghaiTech/Spring-2025/index.html)

School of Information Science and Technology (SIST)

ShanghaiTech University

2025/3/18

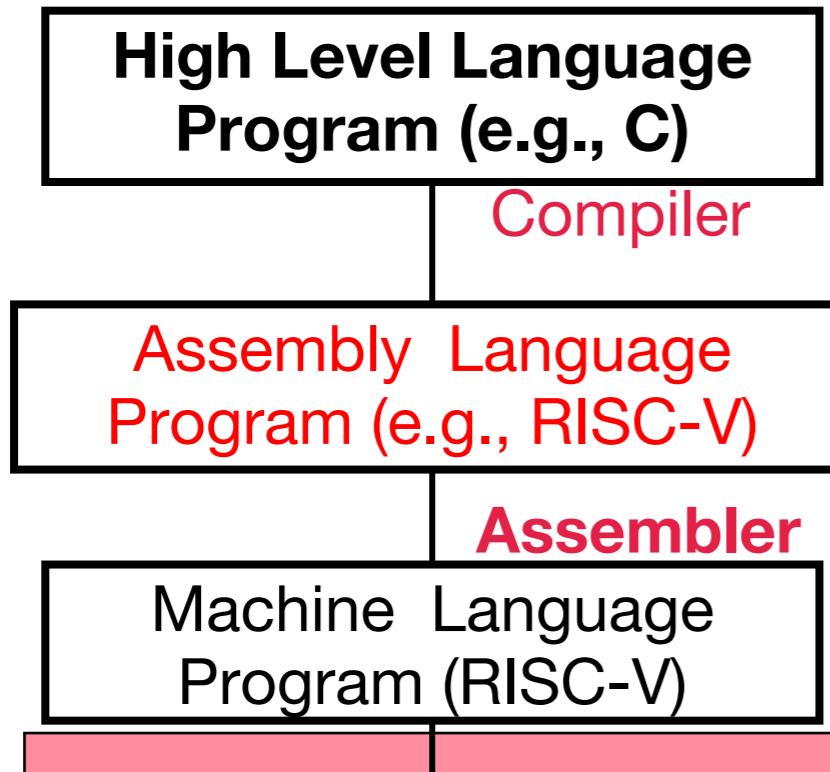
Administratives

- HW3 to release TODAY!
 - Be on time! Only those submissions before ddl will receive marks, otherwise you got 0. So **START EARLY!**
 - Make sure you submit the correct activated version!
- Lab 5 available, please prepare in advance, check next week!
Lab 4 to check this week.
- Proj1.1 released, ddl Mar. 27th.
- Discussion this week on calling convention & proj. 1.1 Q&A, covered by TA Kunchang Guo at teaching center 301.

Outline

- **Digital system**
- **Combinational logics**
 - From transistors to basic logic gates
 - From logic gates to combinational circuits
 - Boolean algebra
 - Boolean expression
 - Truth table
- State elements
- Useful building blocks

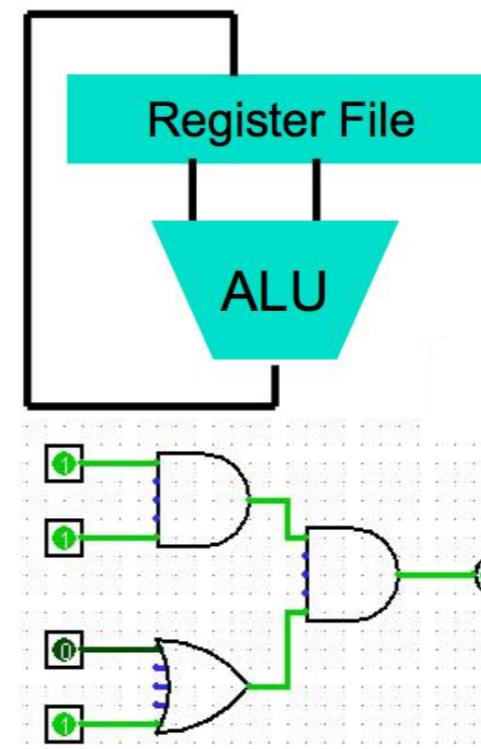
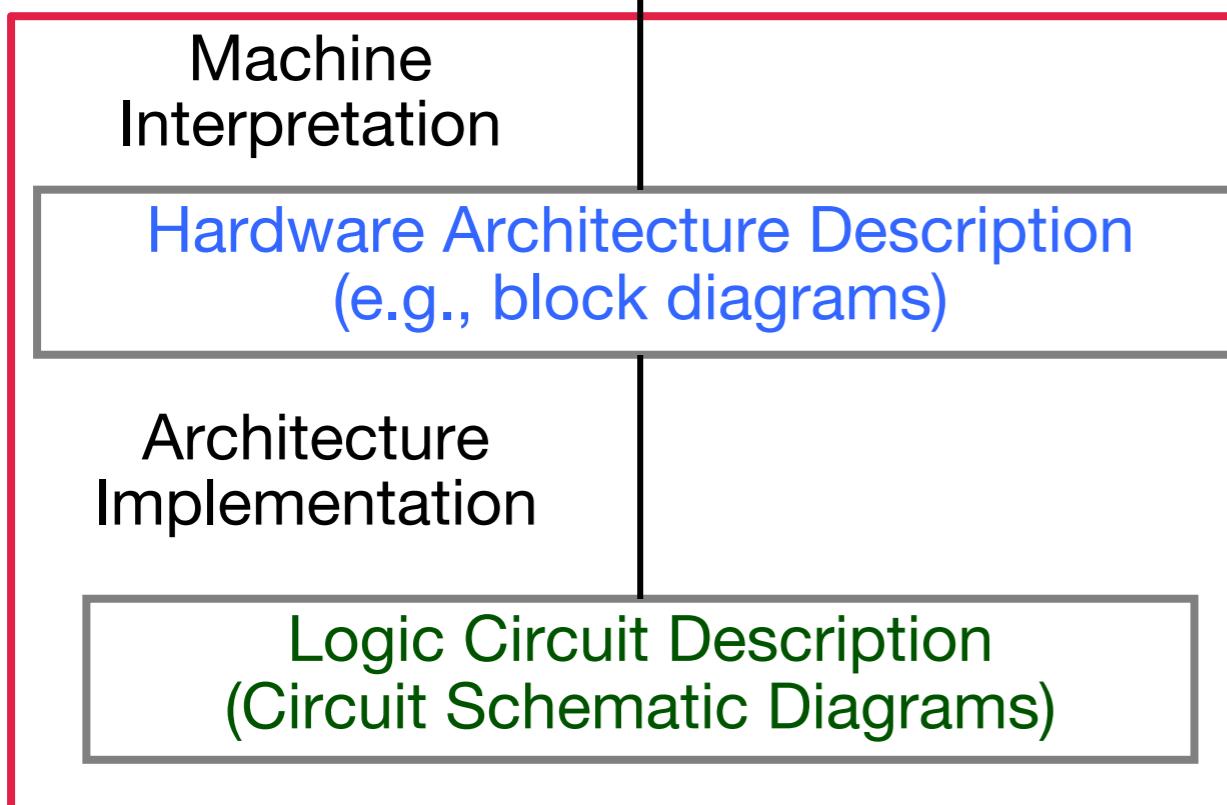
Where are we?



```
temp = v[k];  
v[k] = v[k+1];  
v[k+1] = temp;
```

```
lw t0, 0($2)  
lw t1, 4($2)  
sw t1, 0($2)  
sw t0, 4($2)
```

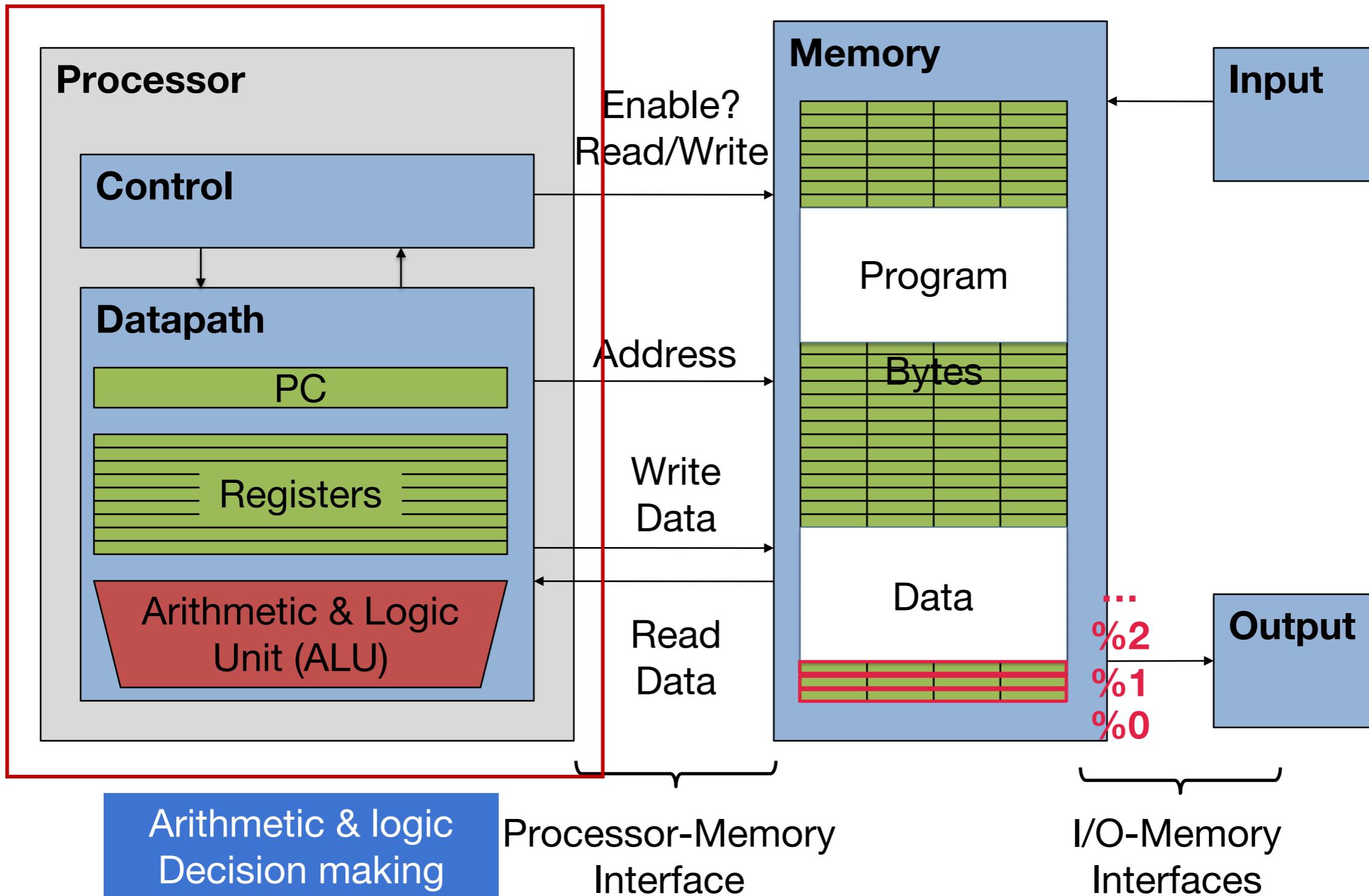
```
0000 1001 1100 0110 1010 1111 0101 1000  
1010 1111 0101 1000 0000 1001 1100 0110  
1100 0110 1010 1111 0101 1000 0000 1001  
0101 1000 0000 1001 1100 0110 1010 1111
```



Hardware (HW) Design

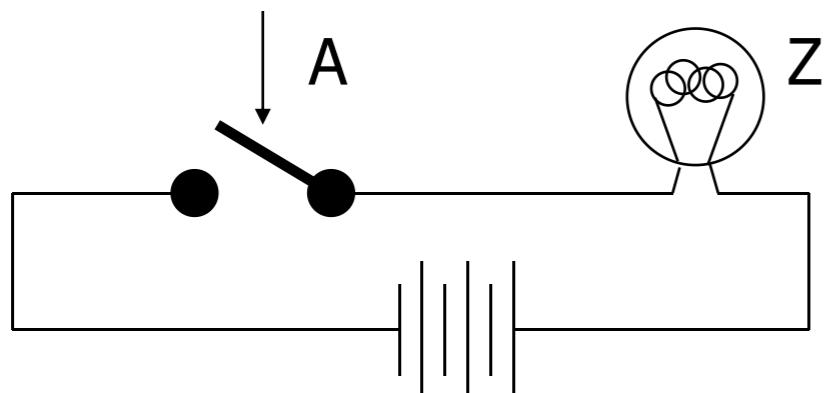
- Next several weeks: how a modern processor is built, starting with basic elements (transistors) as building blocks
- Why study hardware design?
 - Understand capabilities and limitations of HW in general and processors in particular
 - What processors can do fast and what they can't do fast (avoid slow things if you want your code to run fast!)
 - Background for more in-depth HW courses (Digital circuit/VLSI/AI computing system, etc.)
 - There is only so much you can do with standard processors: you may need to design own custom HW for extra performance
 - Even some commercial processors today have customizable hardware!
 - E.g. Google Tensor Processing Unit (TPU)

Components of Computers

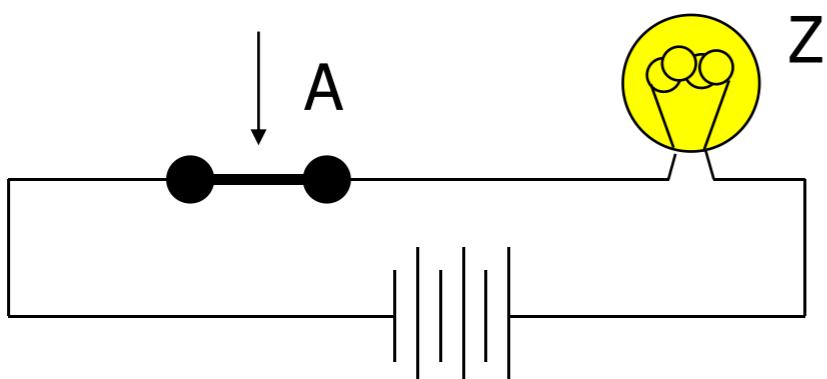


Switches: Basic Element of Physical Implementations

- Implementing a simple circuit (arrow shows action if wire changes to “1” or is *asserted*):



Off-switch (if A is “0” or unasserted)
turns-off light bulb (Z)



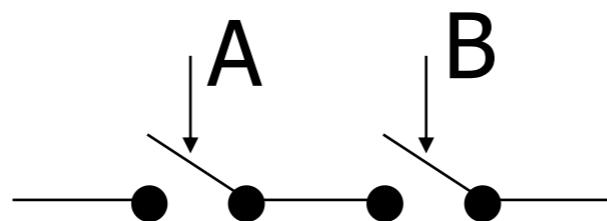
On-switch (if A is “1” or asserted)
turns-on light bulb (Z)

$$Z \equiv A$$

Switches

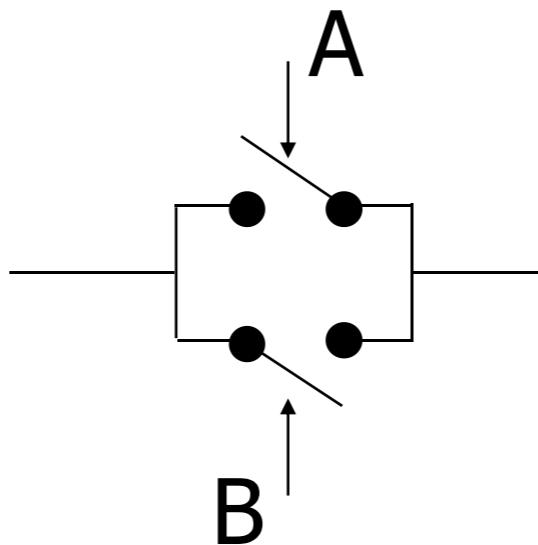
- Compose switches into more complex ones (Boolean functions):

AND



$Z \equiv A \text{ and } B$

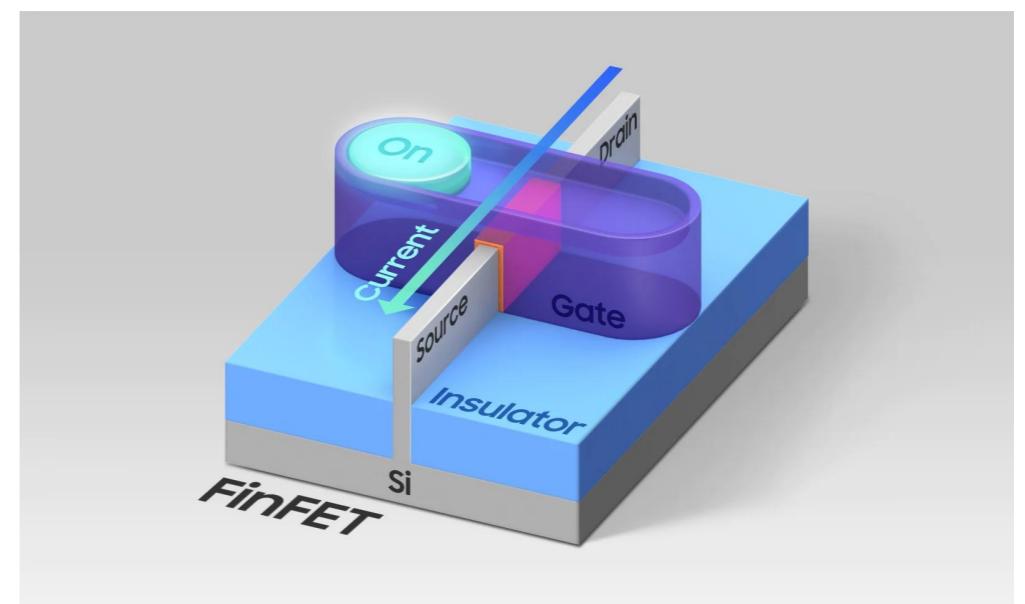
OR



$Z \equiv A \text{ or } B$

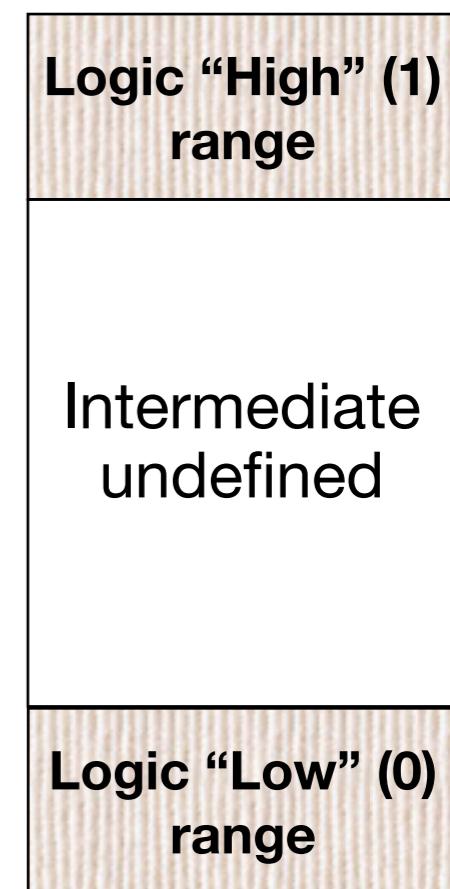
Revisit: Binary System

- 0 and 1 (binary digit or bit, unit of information entropy)
- Decided by the characteristic of semiconductor devices (bi-stable states)
 - **They can also be considered as voltage-controlled switches**
- Resilient to noise (threshold)
- Supported by Boolean algebra theory (George Boole, 1854)
- Basic operations: \wedge , \vee , \sim (Universal set)



Binary Representation of Signals

- High voltage (V_{dd}) represents 1, or true
 - In modern microprocessors, core $V_{dd} \sim 1.0$ Volt
- Low voltage (0 Volt or Ground) represents 0, or false
- **Digital**: discretize signal/voltage to a 0 or a 1
 - This removes noise as signals propagate – a big advantage of digital systems over **analog** systems
 - Circuits to discriminate between two possible inputs are simple to implement and have scaled well with Moore's Law.
- If one switch can control another switch with digital signal, we can build a computer!
- Our switches: CMOS transistors

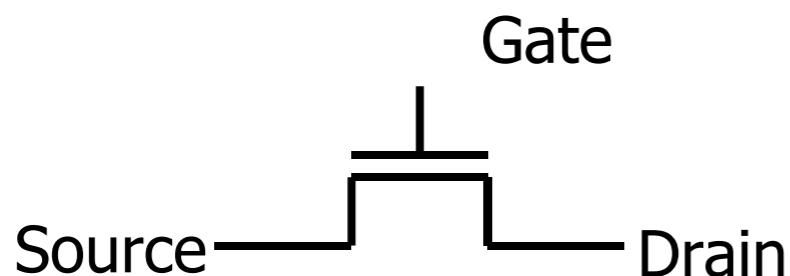
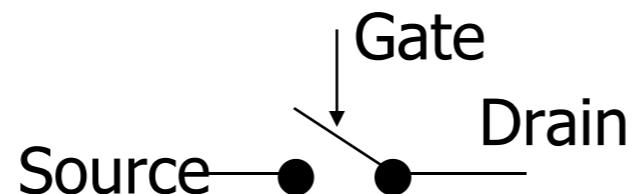


1 V

0 V

NMOS & PMOS Transistors

- Three terminals: source, gate, and drain
 - Basic model



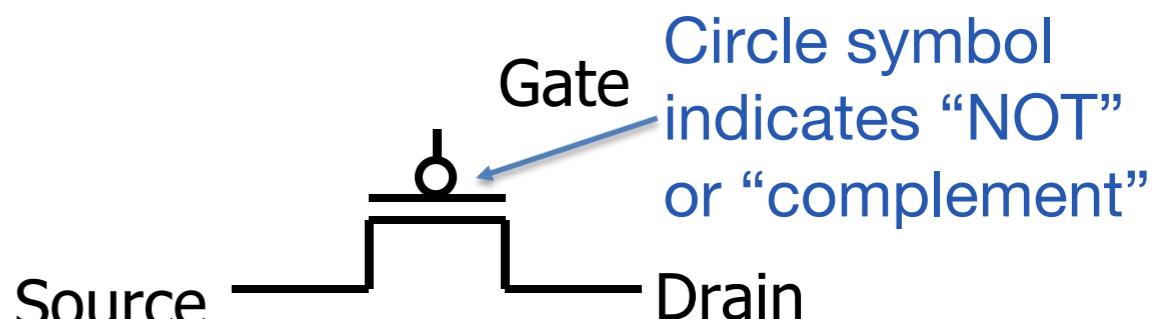
n-channel transistor

off when voltage at Gate is low

on when:

voltage (Gate) > voltage (Threshold)

(**High** resistance when gate voltage **Low**,
Low resistance when gate voltage **High**)



p-channel transistor

on when voltage at Gate is low

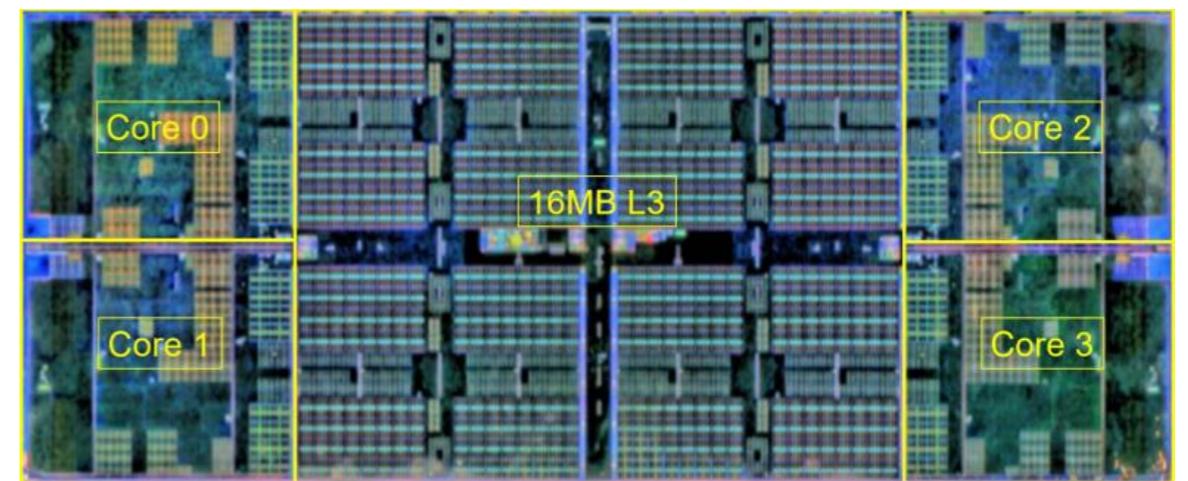
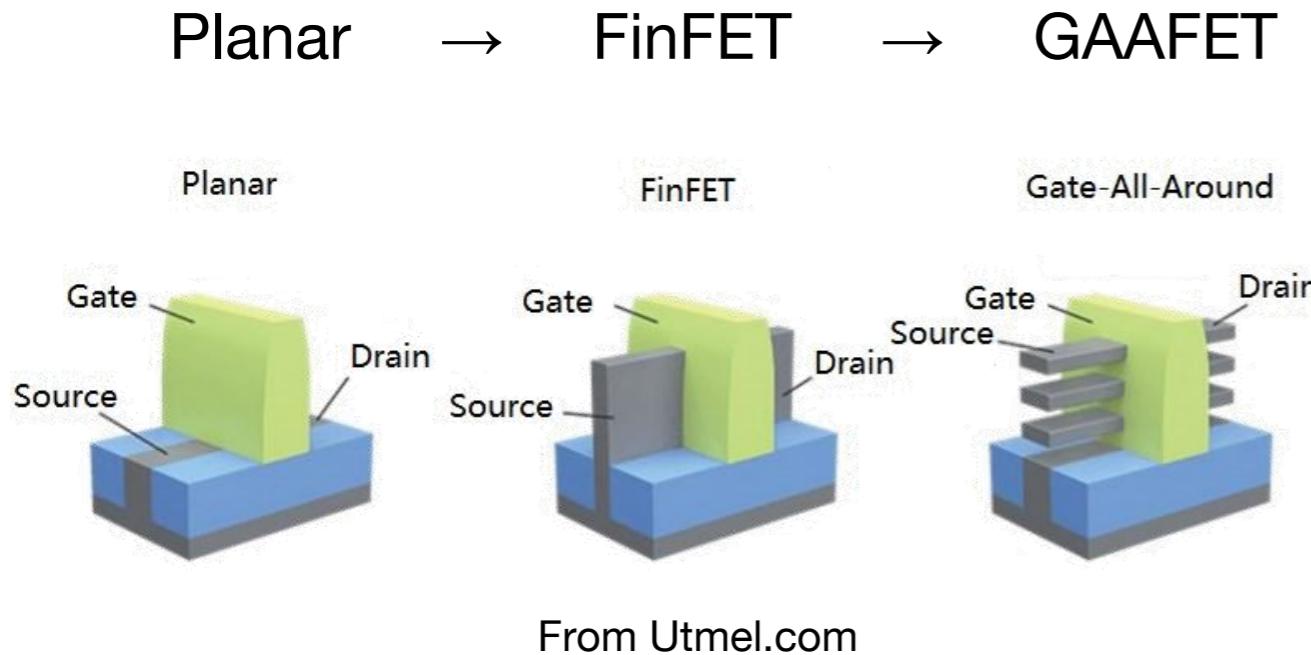
off when:

voltage (Gate) > voltage (Threshold)

(Low resistance when gate voltage **Low**,
High resistance when gate voltage **High**)

NMOS & PMOS Transistors: Clarifications

- Transistors can be modeled by resistors and capacitors, i.e., they can have non-ideal effects such as **leakage** and **delay**
- Recent trend of transistors
- Real stuff: AMD Zen 2

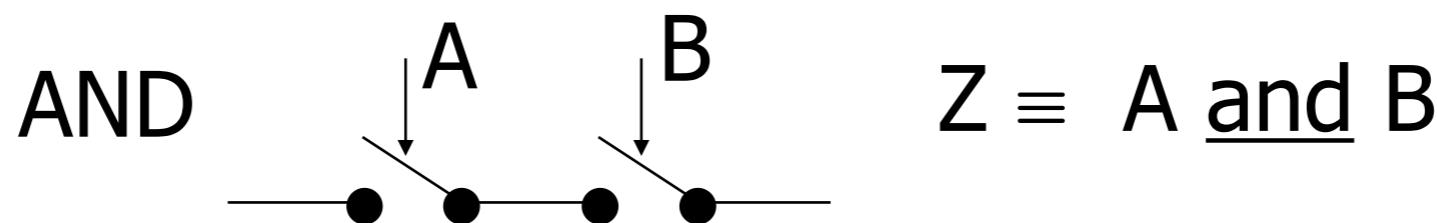


Synchronous Digital System (SDS)

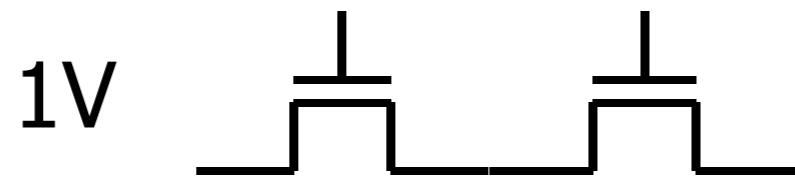
- A system that processes digital signals (0s and 1s)
- Synchronous digital systems consist of two basic types of circuits.
 - Combinational logic circuits (**this lecture**)
 - The outputs solely depend on the input
 - No way to store information
 - State Elements (next lecture)
 - Circuits that store information
 - E.g., registers and memory
- CPU cores are SDS's
 - Our Goal: Implement a RISC-V processor as a synchronous digital system.
 - This SDS should have the capabilities to execute RISC-V instructions.

From Transistors to Logic Gates

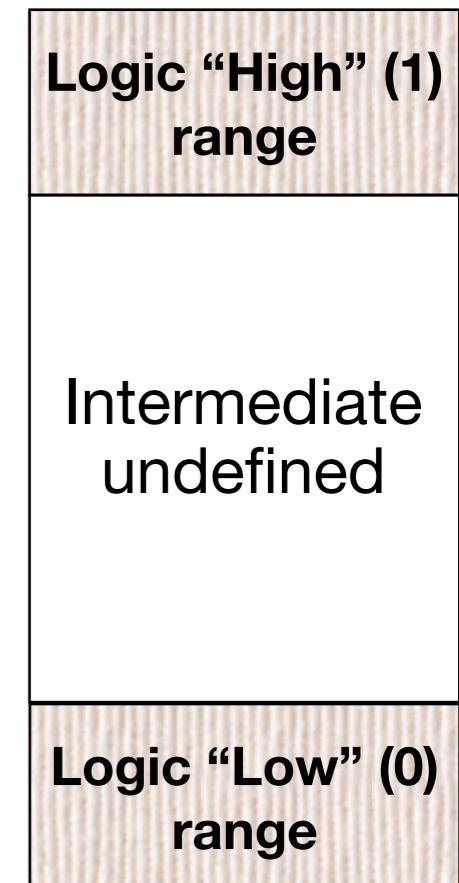
- Complementary MOS (CMOS)



Similarly

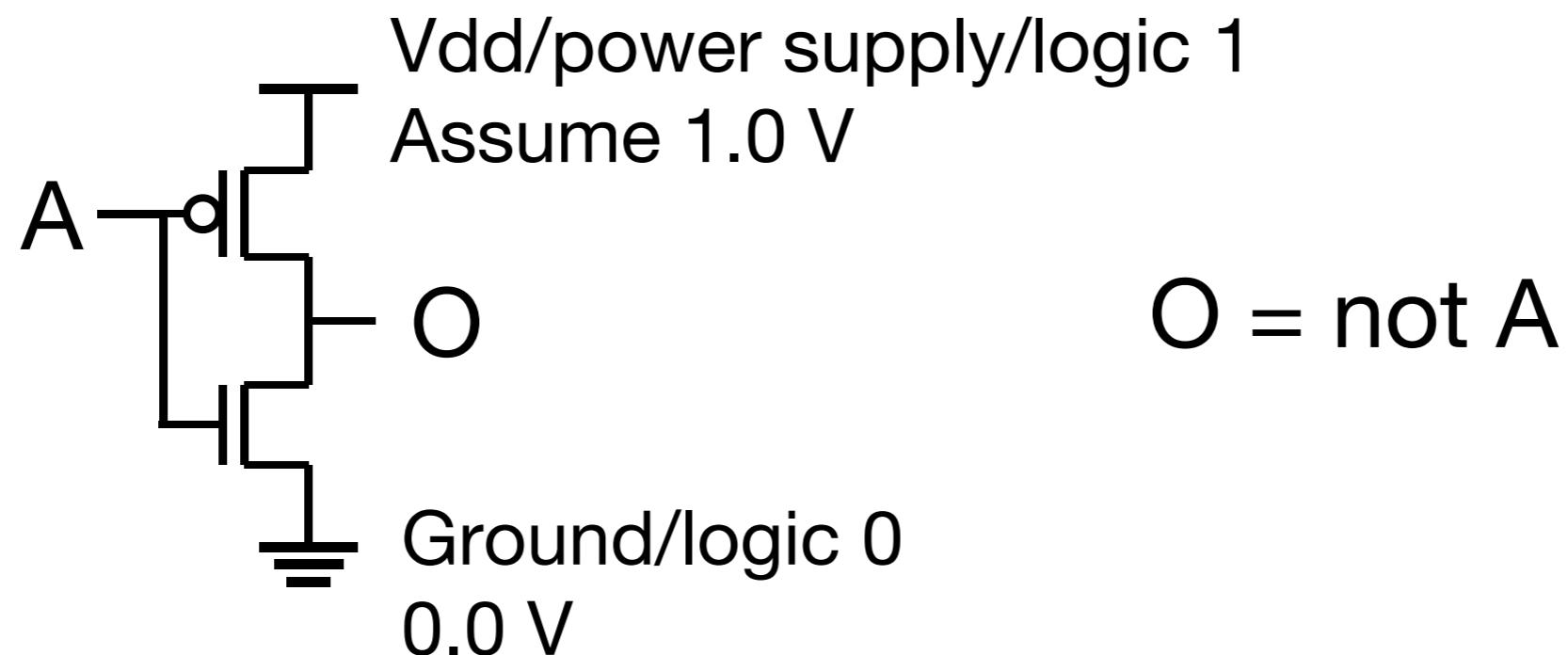


- N-type transistors (NMOS) pass weak 1 ($V_{dd} - V_{th}$) and strong 0
- P-type transistors (PMOS) pass weak 0 (V_{th}) and strong 1
- Pairs of N/P-type transistors to pass strong 0 and strong 1



The Simplest CMOS Circuits

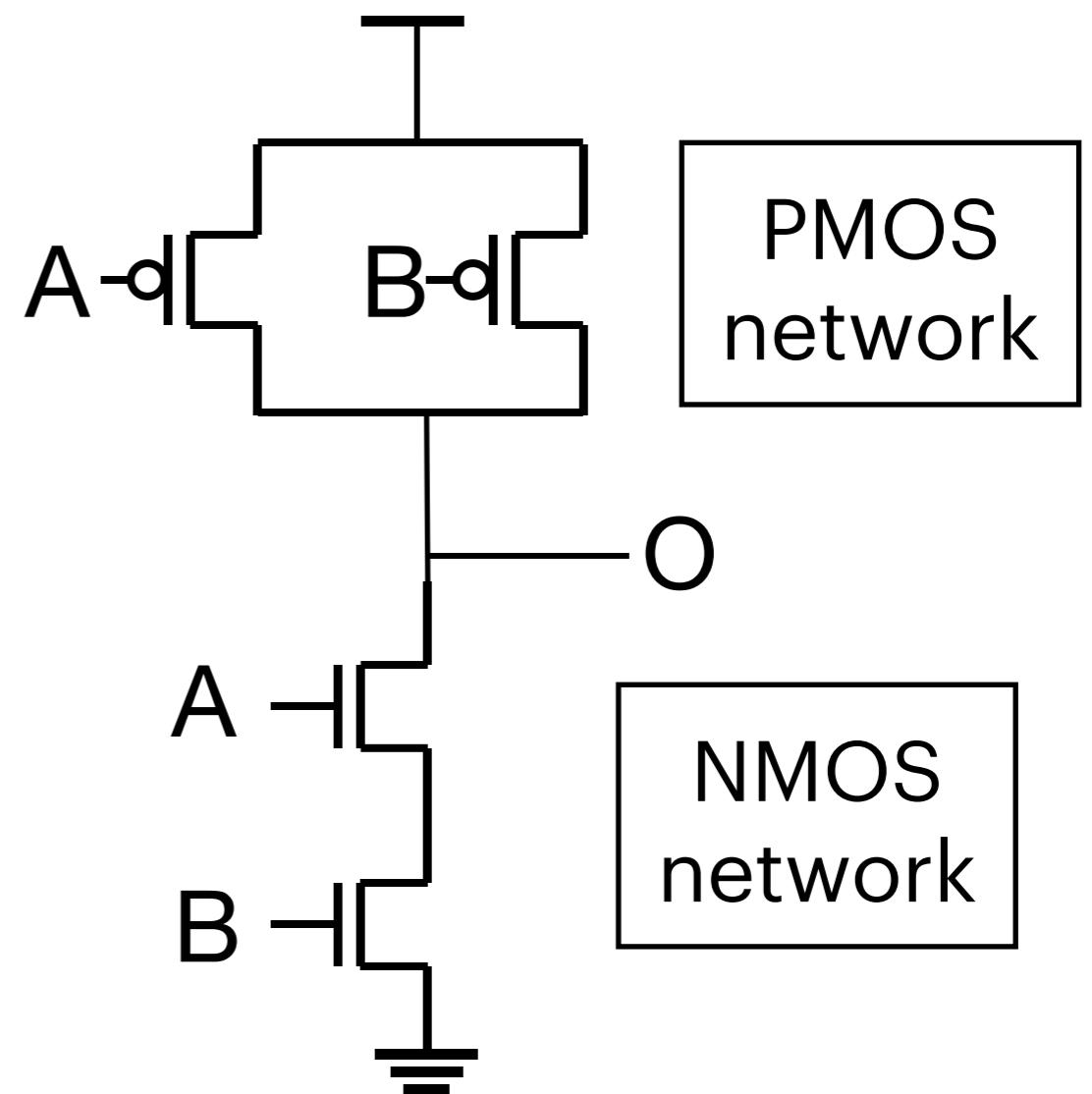
- Inverter/Not gate



NAND Gate

Truth table

A	B	O
0	0	1
0	1	1
1	0	1
1	1	0

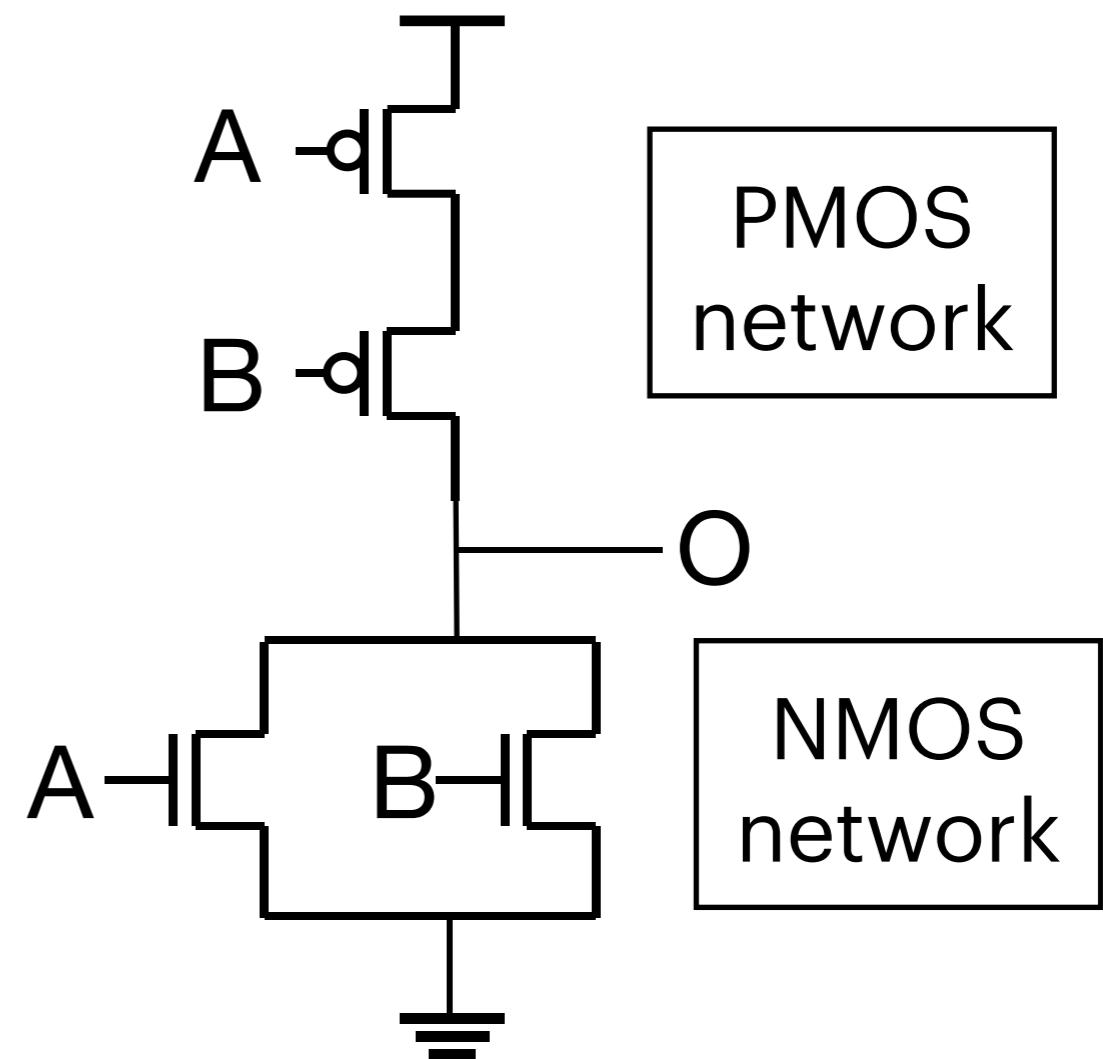


What about 3-input NAND?
What about 2-input AND?

NOR Gate

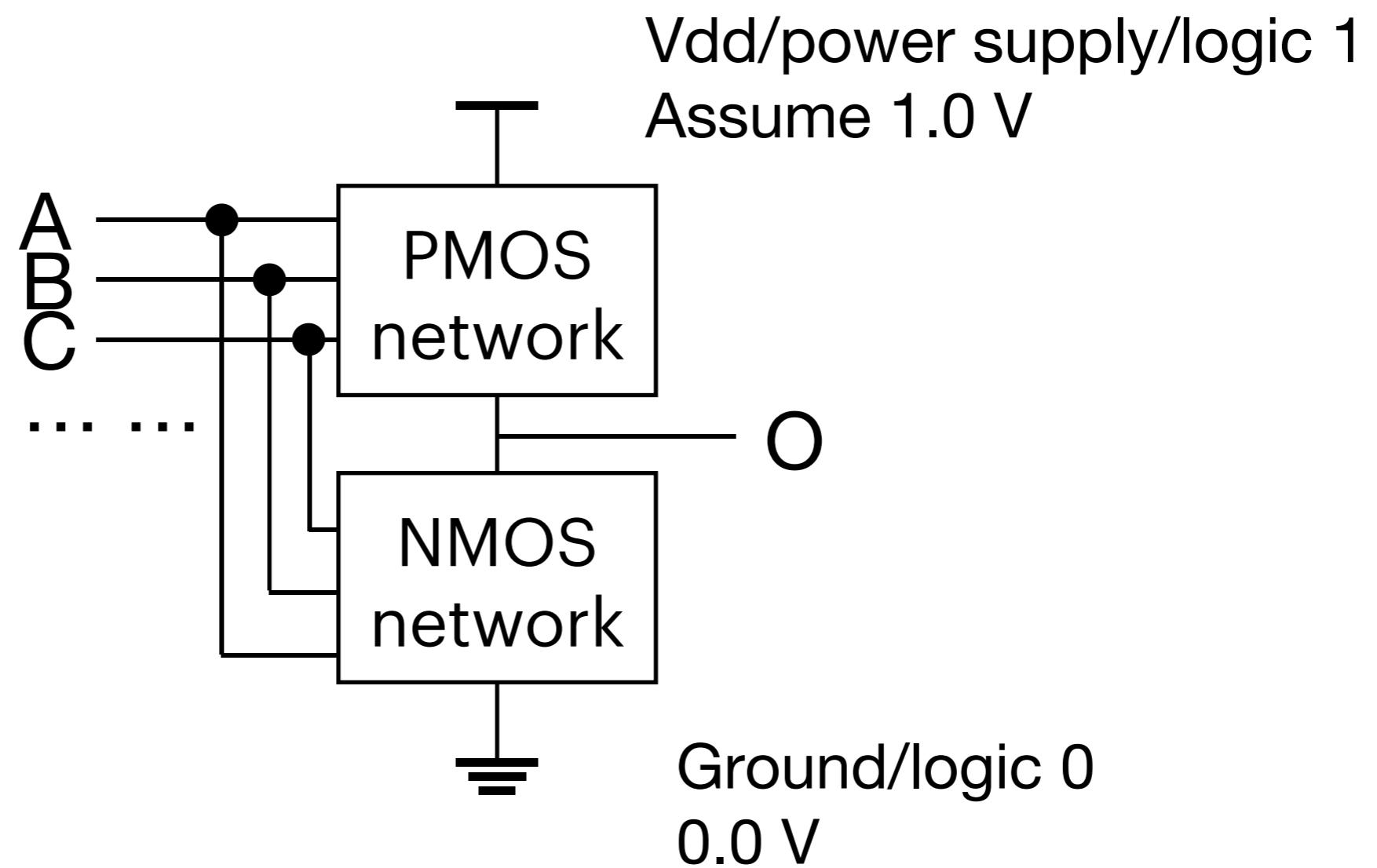
Truth table

A	B	O
0	0	1
0	1	0
1	0	0
1	1	0



What about 3-input NOR?

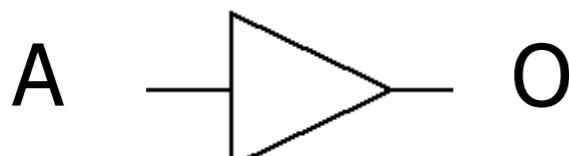
General CMOS Logic Gates



Basic Symbols

- Standard symbols for logic gates

– Buffer, NOT



- Universal sets

– NOT, AND, OR

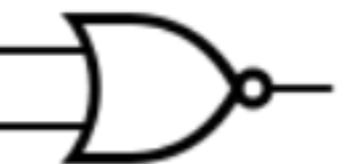
Can be combined to implement any logics

– AND, NAND



– NAND

– OR, NOR



– NOR

Through Boolean algebra!

From Logic Gates to Building Blocks

- Method 1: through boolean expressions (sum-of-minterm)
- Method 2: Karnaugh Map

Boolean Algebra

- Use plus “ $+$ ” for OR
 - “logical sum” $1+0=0+1=1$ (True); $1+1=2$ (True); $0+0=0$ (False)
- Use product for AND ($a \cdot b$ or implied via ab)
 - “logical product” $0 \cdot 0 = 0 \cdot 1 = 1 \cdot 0 = 0$ (False); $1 \cdot 1 = 1$ (True)
- “Bar” to mean complement (NOT)
- Thus

$$\begin{aligned} & ab + a + \bar{c} \\ & = a \cdot b + a + \bar{c} \\ & = (a \text{ AND } b) \text{ OR } a \text{ OR } (\text{NOT } c) \end{aligned}$$



Build Combinational Circuits with Basic Logic Gates

- Combinational circuits: the ones that the output of the digital circuits depends solely on its inputs; usually built with logic gates without feedback
 - **Step 1:** Write down truth table of the desired logic

For example build an XOR
with AND/OR/NOT

A	B	O
0	0	0
0	1	1
1	0	1
1	1	0

Build Combinational Circuits with Basic Logic Gates

- Combinational circuits: the ones that the output of the digital circuits depends solely on its inputs; usually built with logic gates without feedback
 - **Step 2:** Pick the lines with 1 as the output; write them down in *Sum of Minterms (Product)* form;

For example build an XOR
with AND/OR/NOT

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

Minterms

\overline{AB}	m_0
$\overline{A}B$	m_1
$A\overline{B}$	m_2
AB	m_3

Build Combinational Circuits with Basic Logic Gates

- Combinational circuits: the ones that the output of the digital circuits depends solely on its inputs; usually built with logic gates without feedback
 - **Step 3:** Simplify using Laws of Boolean algebra;

For example build an XOR
with AND/OR/NOT

A	B	<i>o</i>
0	0	0
0	1	1
1	0	1
1	1	0

$$o = m_1 + m_2$$

Minterms

$\overline{A}\overline{B}$	m_0
$\overline{A}B$	m_1
$A\overline{B}$	m_2
AB	m_3

Laws of Boolean Algebra

AND form

$$X\bar{X} = 0$$

$$X0 = 0$$

$$X1 = X$$

$$XX = X$$

$$XY = YX$$

$$(XY)Z = X(YZ)$$

$$X(Y+Z) = XY+XZ$$

$$XY+X = X$$

$$\overline{XY} = \overline{X}+\overline{Y}$$

OR form

$$X+\bar{X} = 1$$

$$X+1 = 1$$

$$X+0 = X$$

$$X+X = X$$

$$X+Y = Y+X$$

$$(X+Y)+Z = X+(Y+Z)$$

$$X+YZ = (X+Y)(X+Z)$$

$$(X+Y)X = X$$

$$\overline{X+Y} = \overline{X}\overline{Y}$$

Complementarity

Laws of 0's and 1's

Identities

Idempotent Laws

Commutativity

Associativity

Distribution

Absorption

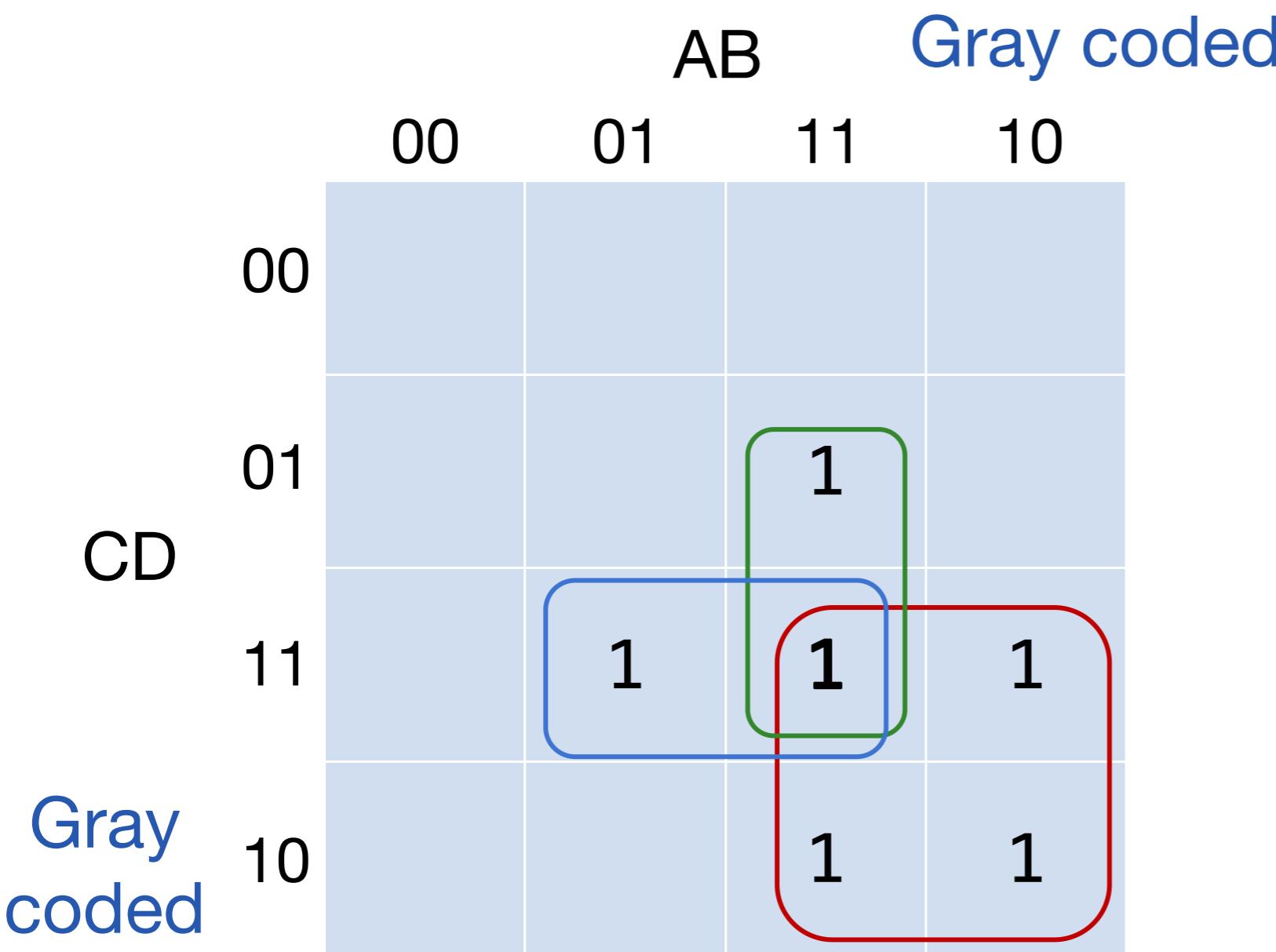
DeMorgan's Law

Your turn!

- Build a half adder:
 - Sum Carry
 - $0 + 0 = 0$ 0
 - $0 + 1 = 1$ 0
 - $1 + 0 = 1$ 0
 - $1 + 1 = 0$ 1
- Build a 2-bit adder:

	Sum	Carry	Sum	Carry
• $00 + 00 = 00$	0	10 + 00 = 10	0	0
• $00 + 01 = 01$	0	10 + 01 = 11	0	0
• $00 + 10 = 10$	0	10 + 10 = 00	1	
• $00 + 11 = 11$	0	10 + 11 = 01	1	
• $01 + 00 = 01$	0	11 + 00 = 11	0	0
• $01 + 01 = 10$	0	11 + 01 = 00	1	
• $01 + 10 = 11$	0	11 + 10 = 01	1	
• $01 + 11 = 00$	1	11 + 11 = 10	1	
	AB	CD		

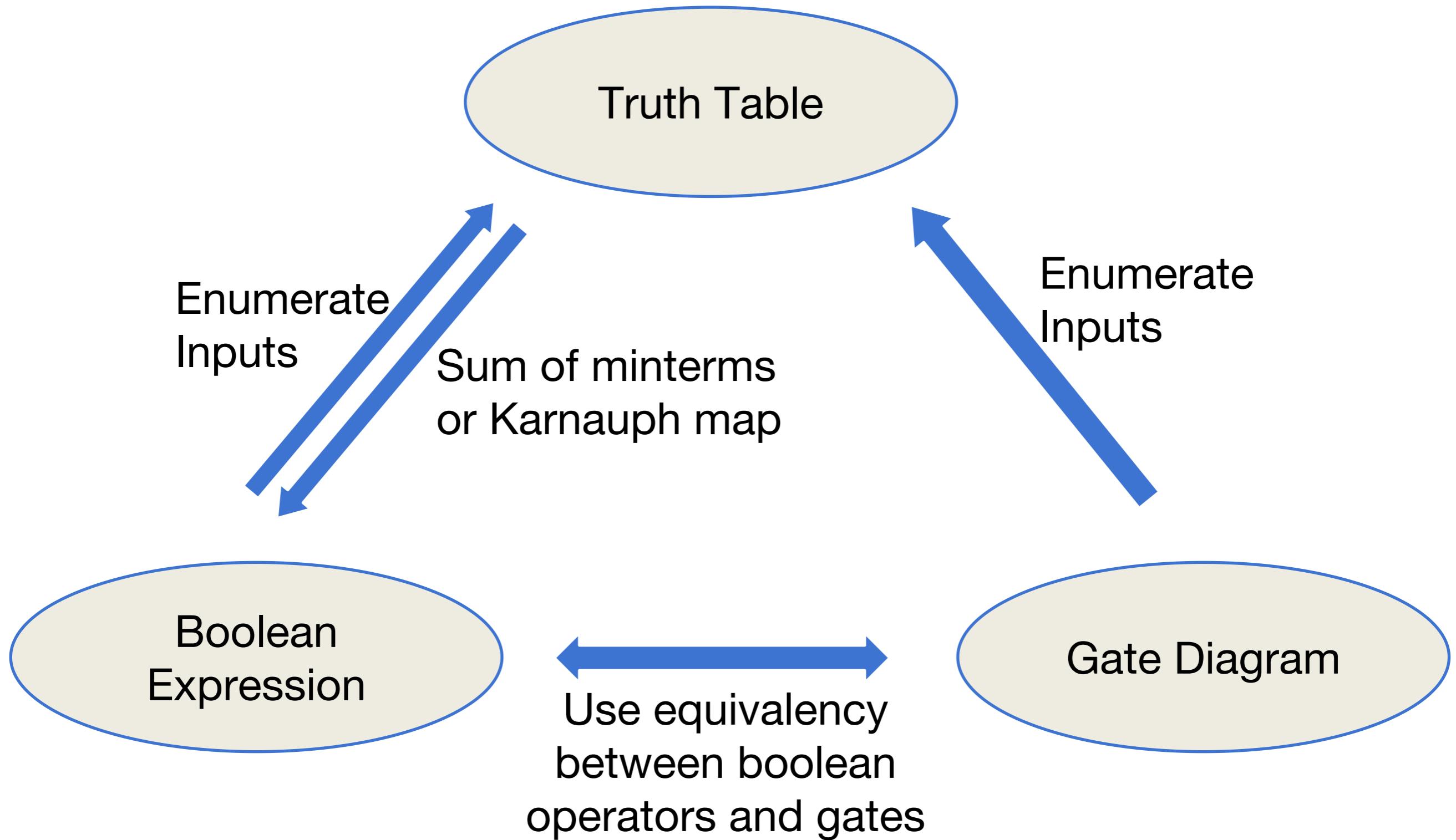
Another Method—Karnaugh Map (optional)



Each cell corresponds to a minterm

Online Karnaugh map solver: <http://www.32x8.com/index.html>

Representations of Combinational Logic

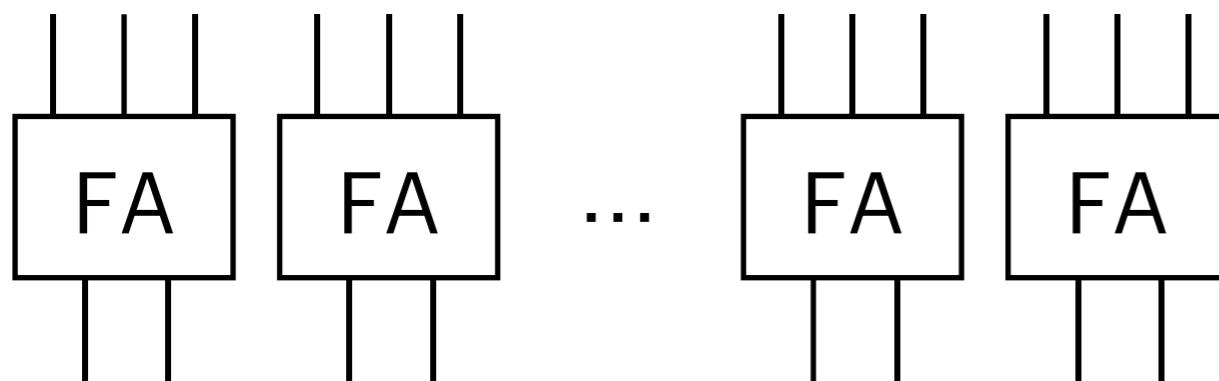


Build Larger Blocks—like LEGO®

$$\begin{array}{r}
 01010101 \\
 + 01110011 \\
 \hline
 \end{array}$$

Build a full adder (FA): truth table

Carry in	A	B	Sum	Carry out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

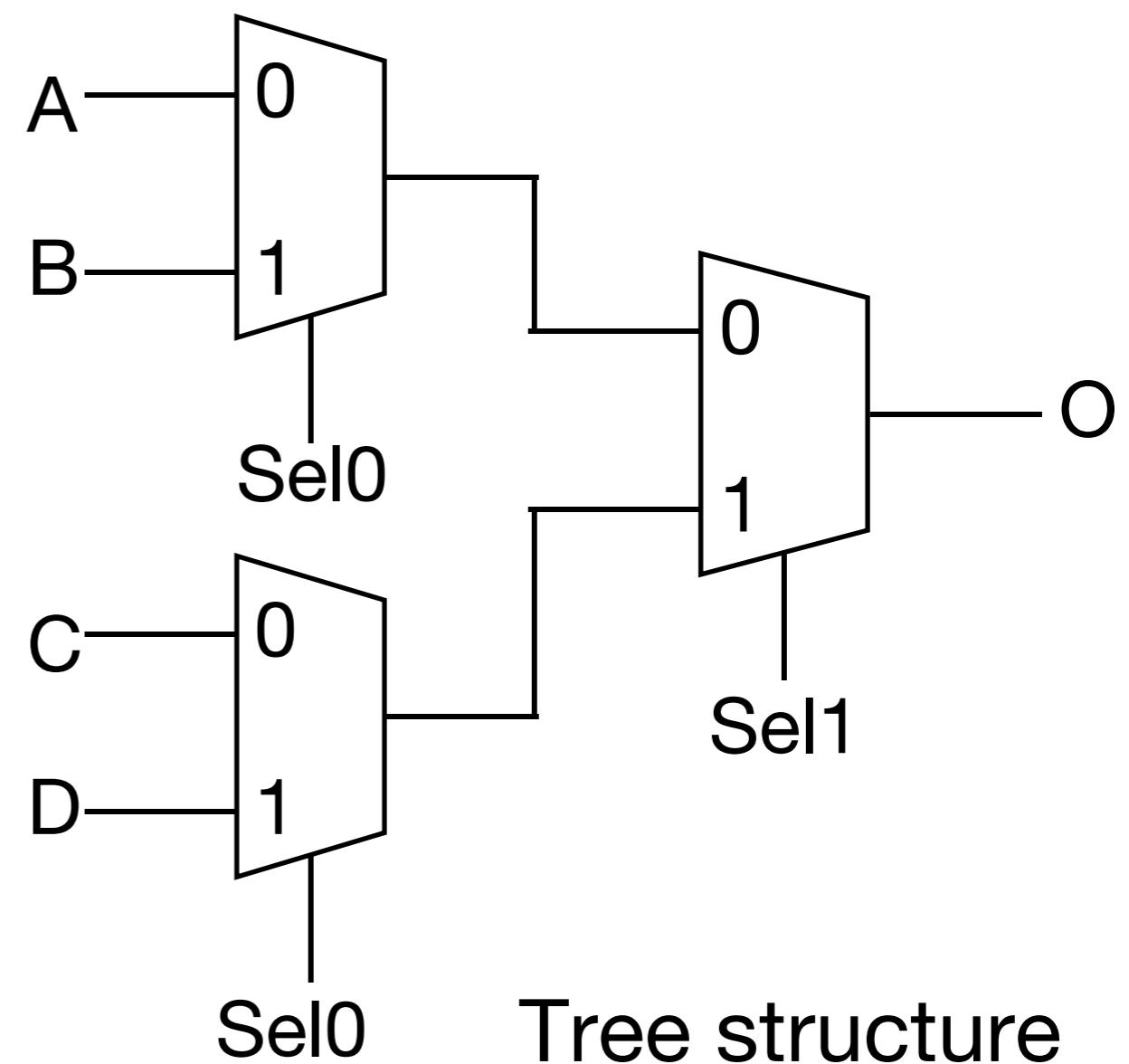
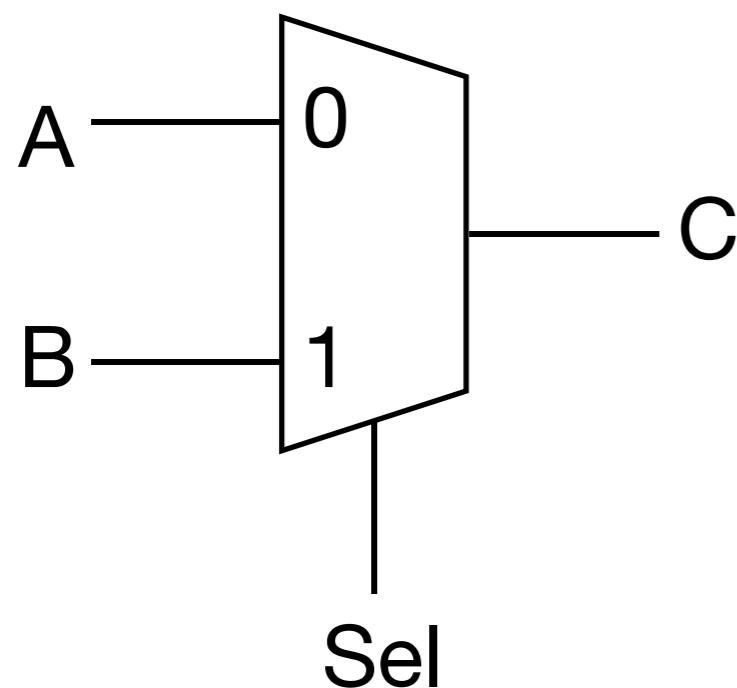


Exercise

- Recall beq instruction. Build a comparator that makes the decision. 1 indicates “equal”, 0 indicates “not equal”

Other Useful Combinational Circuits

- Multiplexer (2^n -to-1)



Tree structure