Computer Mathematics

Week 8
Combinational logic circuits

KUAS 京都先端科学大学
KYOTO UNIVERSITY of ADVANCED SCIENCE
Department of Mechanical and Electrical System Engineering

last week

the mathematics of logic circuits

- the foundation of all digital design

Boolean logic

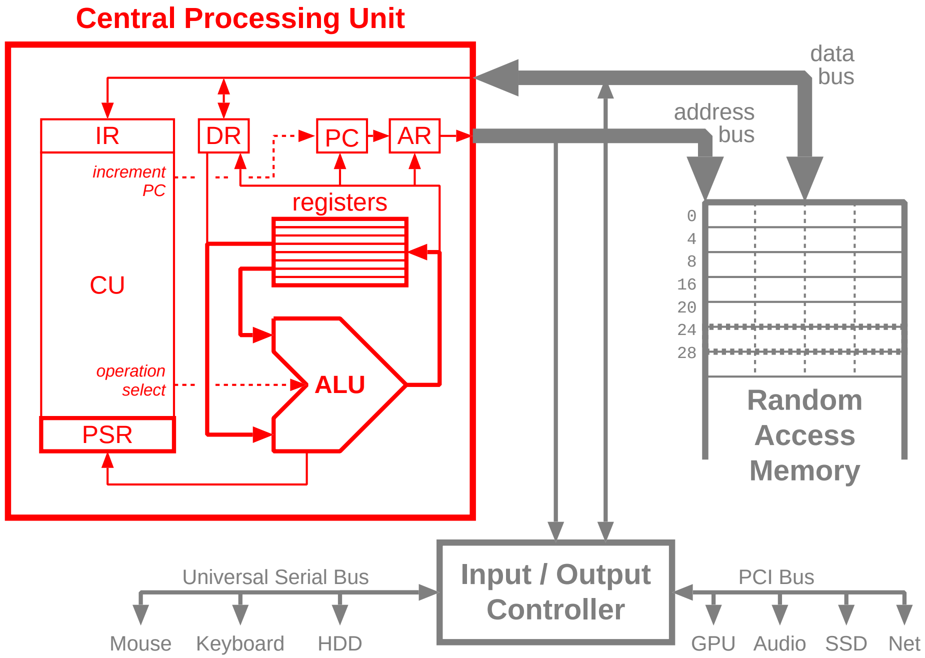
- when `0` and `1` represent true and false

Boolean algebra

- Boolean functions
- canonical forms

simplification of Boolean expressions

- de Morgan's laws



this week

combinational digital circuits

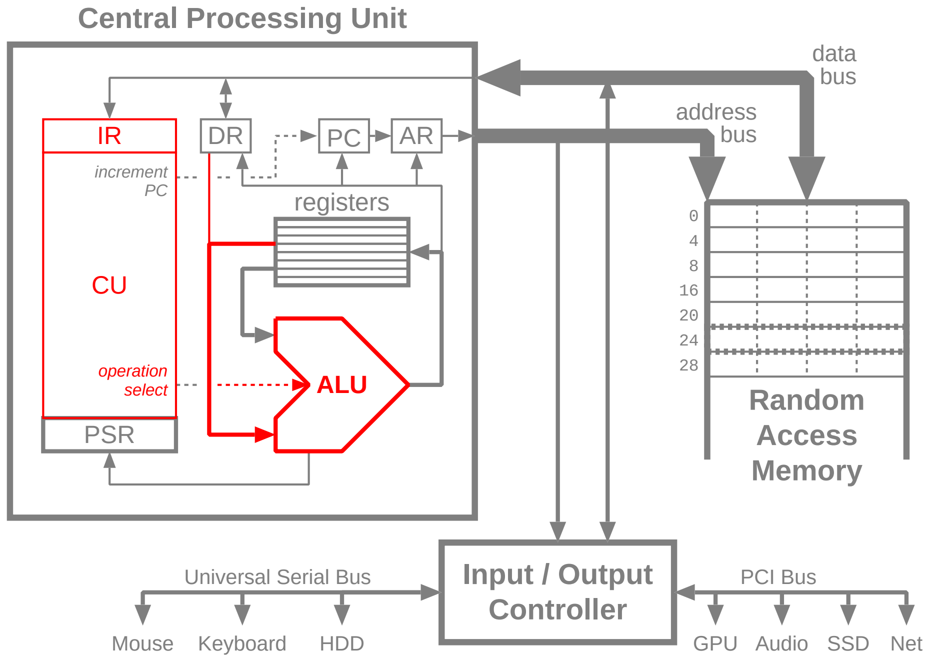
wires, signals and connections

logic gates

- and, or, not
- nand, nor, xor

gate-level arithmetic operations

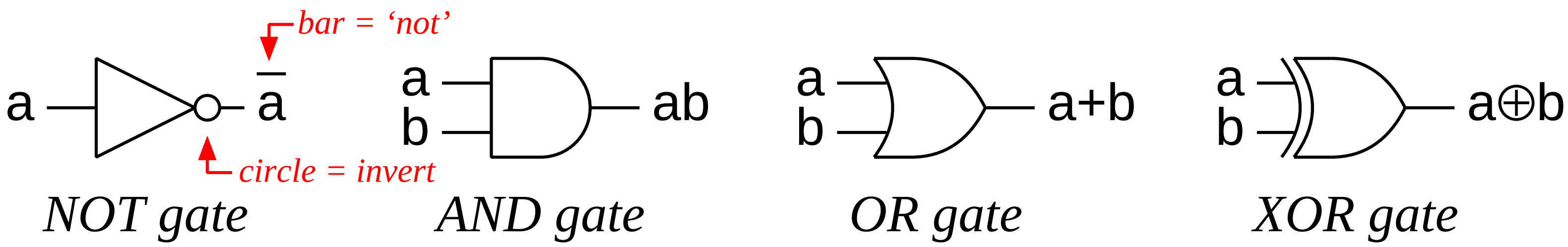
- how logic turns into addition



logic gates

logic *gates* implement Boolean operations

- one or more inputs, one or more outputs
- outputs are a logical function of the inputs
- logic circuits use electrical engineering notation, not mathematical notation



another useful gate: *exclusive-or* (XOR)

- not equivalent, or modulo-2 addition/subtraction
- very useful for arithmetic operations

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

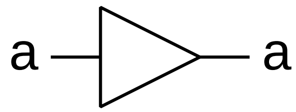
beware: $\overline{ab} = (a \cdot b)'$, but $\overline{a}\,\overline{b} = a' \cdot b' = (a + b)'$

- use an explicit '·' if it helps readability, e.g., $\overline{a} \cdot \overline{b}$

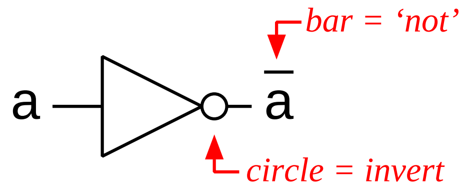
more logic gates

the small circle indicates an inversion

- an *active-low* signal ('not' function)
- written with an $\overline{\text{overbar}}$



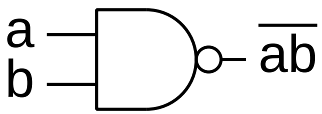
buffer (no change)



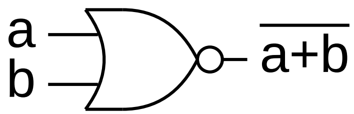
NOT gate (inverter)

it can be placed on any output (or input)

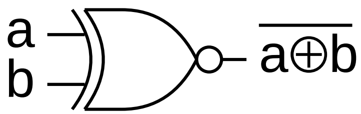
- when 'true' that output (or input) will be 0
- NAND = 'not AND', NOR = 'not OR', XNOR = 'not XOR'



NAND gate



NOR gate



XNOR gate

NAND

a	b	$\overline{ab}$
0	0	1
0	1	1
1	0	1
1	1	0

NOR

a	b	$\overline{a+b}$
0	0	1
0	1	0
1	0	0
1	1	0

XNOR

a	b	$\overline{a \oplus b}$
0	0	1
0	1	0
1	0	0
1	1	1

wires, signals, and connections

signals

- a *signal* is anything that conveys a logic value (or other message)

wires

- a *wire* carries a signal between two or more points in an electrical circuit
- all points connected by the wire have the same logic value

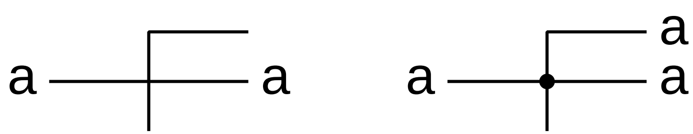
named signals

- any signal can be given a name
- circuit inputs and outputs are usually named
- intermediate signals can be named, to show logical relationships

wire, signal *named signal*

connections

- crossing wires have no connection
- unless an explicit connecting dot is present



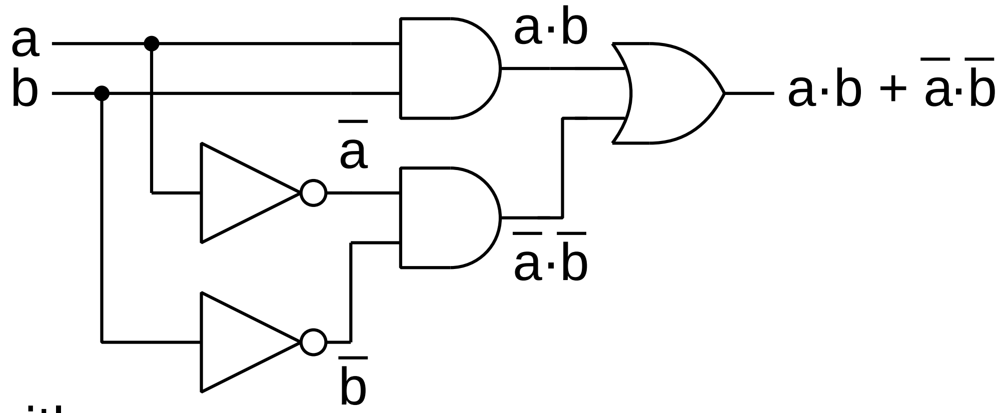
no connection *connection*

logic circuits

Boolean logic and functions

- logical operators perform computation
- operands transmit values implicitly (results of $\cdot$ to input of $+$ below)
- variables transmit values explicitly (e.g., function parameters to expression)

$$\text{equal}(a, b) = a \cdot b + a' \cdot b'$$



logic circuits

- logic gates perform computation
- wires transmit values explicitly
- signal names can transmit values implicitly
  (a and b could be generated elsewhere in the circuit above)
- signals typically flow left-to-right (with frequent exceptions)

abstraction — functional blocks

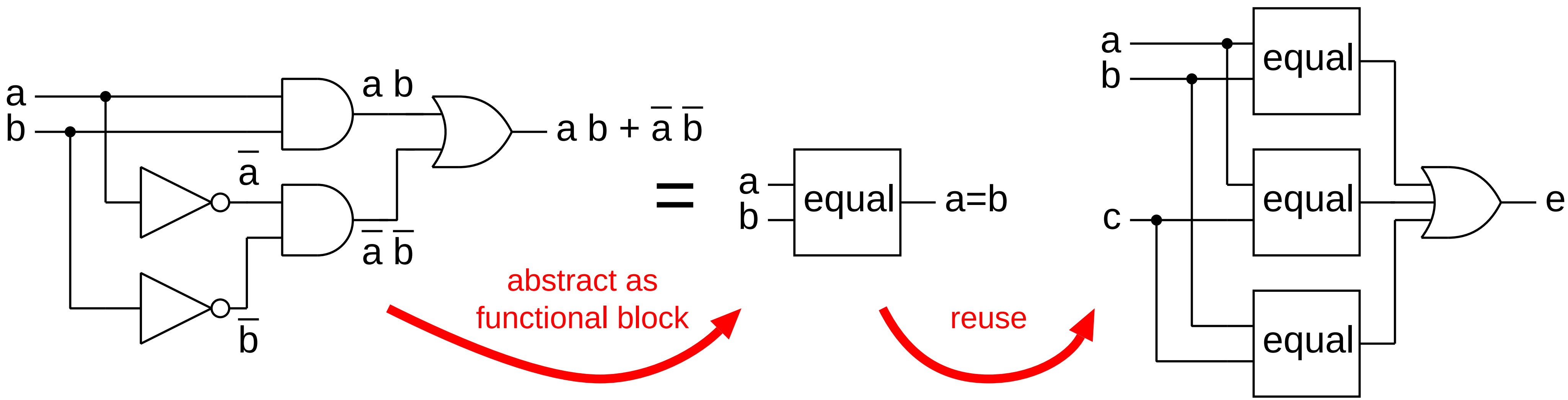
Boolean logic and functions

- functions provide abstraction

$$\begin{aligned} \text{equal}(a,b) &= ab + a'b' \\ \text{any-two}(a,b,c) &= \text{equal}(a,b) + \text{equal}(a,c) + \text{equal}(b,c) \\ e &= \text{any-two}(x,y,z) \end{aligned}$$

logic circuits

- *functional blocks* (components) provide abstraction



Boolean function to logic circuit

e.g., single-bit addition of two inputs

- sum is 1 if exactly one of a and b is 1 (i.e., $a \neq b \Leftrightarrow a \oplus b$)
- carry is 1 if a and b are both 1 (i.e., $a \cdot b$)

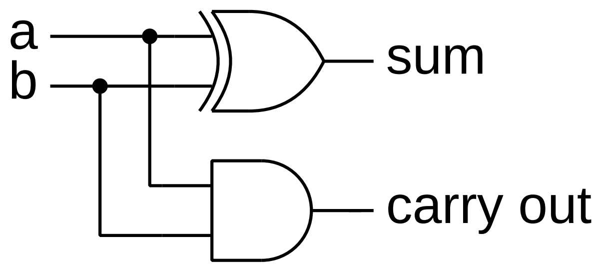
sum

a	b	c_o	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

canonical form of each output

$$s = ab' + a'b = a \oplus b$$
$$c_o = ab$$

translated into gates



this is a *half adder*

- no provision for carry in
- not useful for multi-bit additions

addition (single-bit)

sum

c_i	a	b	c_o	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

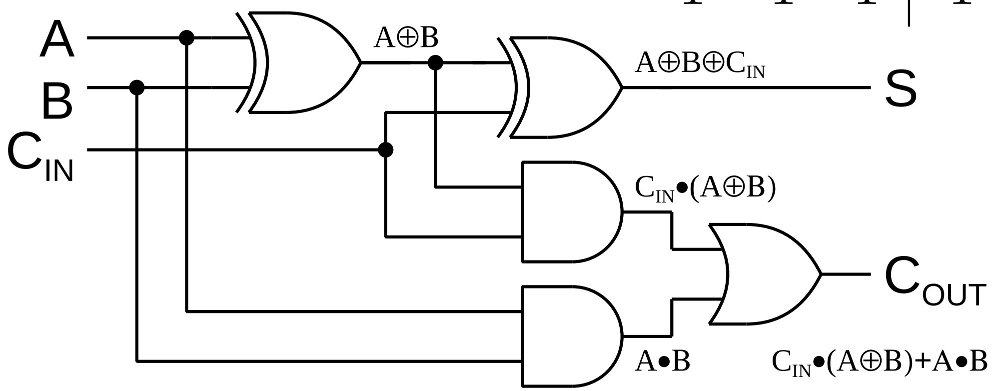
single-bit addition of three inputs

- sum is 1 if an odd number of inputs are 1
- carry is 1 if two or more inputs are 1

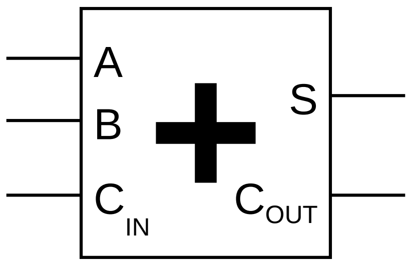
canonical form of each output, simplified, translated into gates

$$
\begin{aligned}
s &= c_i'ab' + c_i'a'b + c_ia'b' + c_iab \\
&= c_i'(ab' + a'b) + c_i(a'b' + ab) \\
&= c_i'(a \oplus b) + c_i(a \oplus b)' \\
&= c_i \oplus a \oplus b
\end{aligned}
$$

$$
\begin{aligned}
c_o &= c_i'ab + c_ia'b + c_iab' + c_iab \\
&= (c_i' + c_i)ab + c_i(a'b + ab') \\
&= ab + c_i(a \oplus b)
\end{aligned}
$$



=



this is a *full adder*

homework

practice drawing logic circuits for Boolean functions

consider some of the gates we did not study in detail

- how many of the logic circuits of this week can you make
  - using only NAND gates?
  - using only NOR gates?

reinforce your understanding

- write a Python program to simulate a multi-bit adder
  - consider the logical operations affecting each signal
  - compute the output of each gate based on its input(s)
  - propagate outputs to inputs at every simulation time step

ask about anything you do not understand

- from any of the classes so far this semester (or the lecture notes)
- it will be too late for you to try to catch up later!
- I am always happy to explain things differently and practice examples with you

next week

signals and busses

gate-level multi-bit logical operations

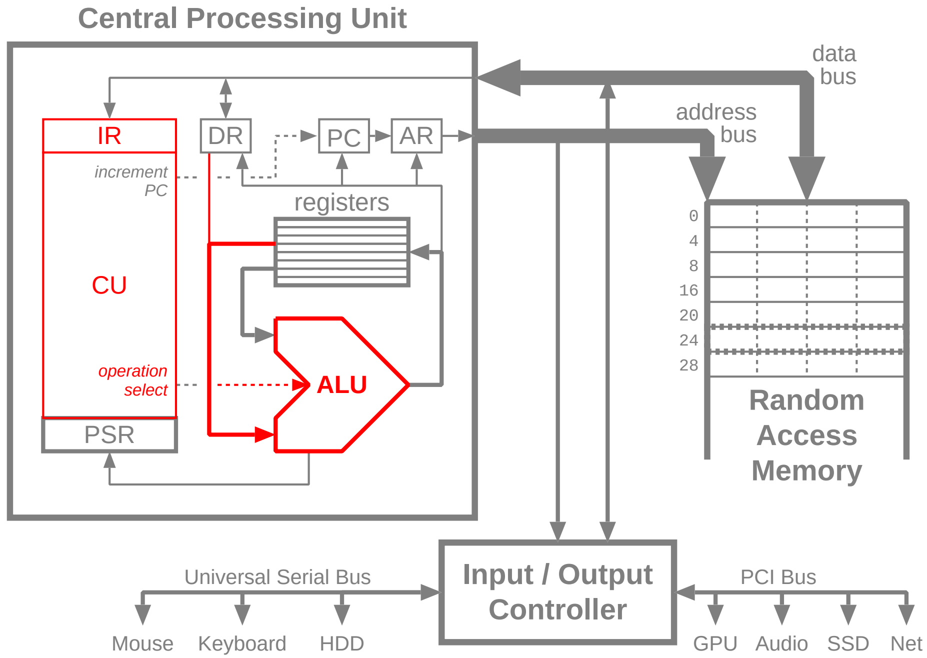
- bitwise: and, or, not

gate-level multi-bit arithmetic operations

- addition, subtraction (unsigned, 2's complement)

1-of-N selection

- multiplexers



glossary

active-low — a signal that is considered ‘true’ when 0.

active-high — a signal that is considered ‘true’ when 1.

adder — a logic circuit that implements 2’s complement addition between two words of data.

carry — a processor status bit indicating that the last arithmetic operation generated an unsigned overflow (a carry out of the MS bit).

exclusive-or — an ‘or’ operation that does not allow both inputs to be the same value.

full adder — an adder that takes three single-bit inputs (two digits and a carry in) and produces two single-bit outputs (a sum and a carry out).

functional block — a high-level abstract component in a digitl circuit that represents a reusable pattern of lower-level components or gates.

gate — an logic circuit component that implements a fundamental Boolean operation.

half adder — an adder that takes two single-digit inputs and produces a sum and carry output.

signal — anything that conveys a logic value from one place to another. In logic circuits, a signal is carried by a wire.

wire — a connection between several points in a circuit that forces them to all have the same logical value.