Computer Mathematics

Week 7
Logic and Boolean algebra

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

last week

coding theory

- channel coding

information theory concept

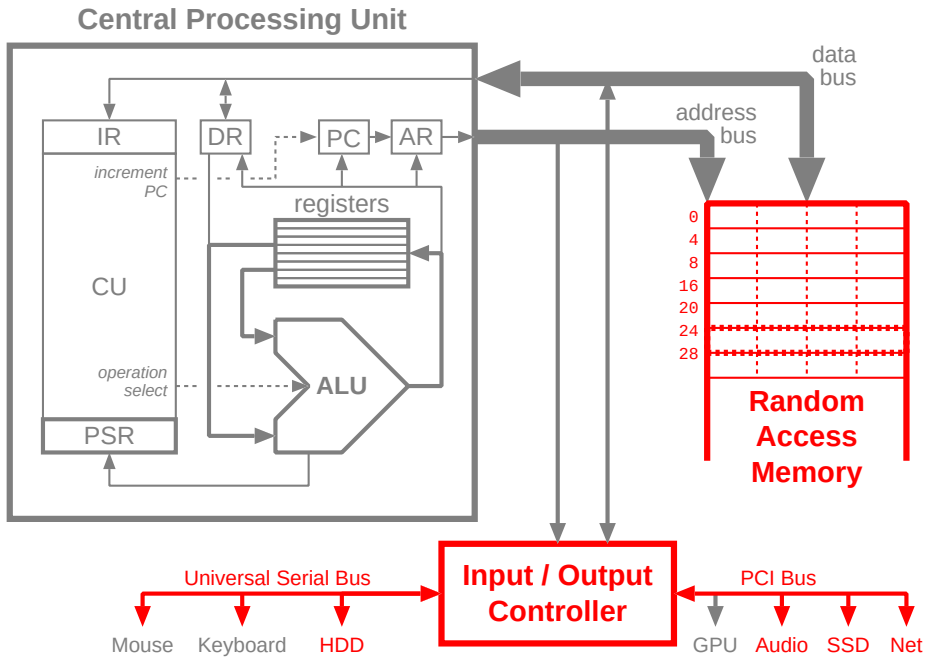
- Hamming distance

error detection

- motivation
- parity
- cyclic redundancy checks

error correction

- motivation
- block parity
- Hamming codes



this week

the mathematics of logic circuits

- the foundation of all digital design

Boolean logic

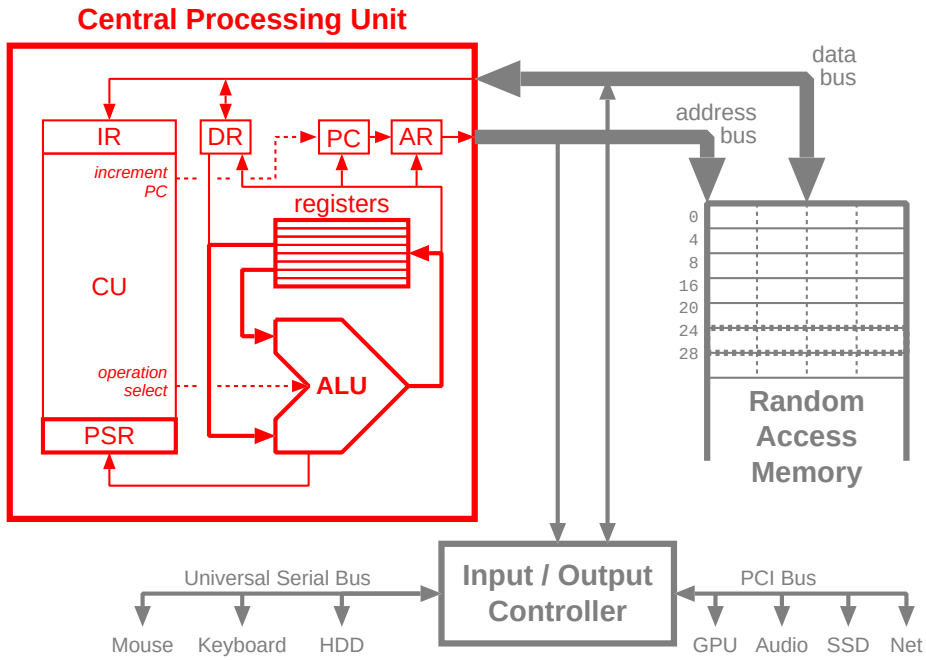
- when `0` and `1` represent true and false

Boolean algebra

- Boolean functions
- canonical forms

simplification of Boolean expressions

- de Morgan's laws



logic circuits

combinational logic (this week, next week)

- similar to mathematical functions
- output depends only on current inputs
- no memory of past inputs or outputs (stateless)
- used to implement arithmetic and logical functions
  - e.g., the ALU, instruction decoding, etc.

sequential logic (week after next)

- memory of past inputs and/or outputs (stateful)
- output depends on current inputs and/or past inputs

synchronous logic (week after next)

- sequential logic with a clock
- change to sequential state occurs whenever the clock ‘ticks’
- used to implement control and iterative functions
  - e.g., the control unit, multiply/divide instructions, registers, etc.

Boolean algebra

only two values: *false* and *true*, usually written 0 and 1 respectively

three fundamental operations

or: $x + y$ is true if either x or y is true

and: $x \cdot y$ is true if both x and y are true

not: x' is true if x is not true

OR	AND	NOT
$0 + 0 = 0$	$0 \cdot 0 = 0$	$0' = 1$
$0 + 1 = 1$	$0 \cdot 1 = 0$	$1' = 0$
$1 + 0 = 1$	$1 \cdot 0 = 0$	
$1 + 1 = 1$	$1 \cdot 1 = 1$	

notation

or	$x + y$	$x \cup y$	$x \vee y$	logical *sum*, *union*, or *disjunction*
and	$x \cdot y$	$x \cap y$	$x \wedge y$	logical *product*, *intersection*, or *conjunction*
not	x'	$\overline{x}$	$\neg x$	logical *complement*, or *negation*

$x \cdot y$ can be written xy (like implicit multiplication in normal arithmetic)

$+$ and $\cdot$ have the same precedence as addition and multiplication, respectively

the unary negation operator x' has highest precedence

truth tables

truth tables list the input(s) and corresponding output of a Boolean operation

OR

x	y	$x+y$
0	0	0
0	1	1
1	0	1
1	1	1

AND

x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

NOT

x	x'
0	1
1	0

or of a Boolean expression, e.g., $x + y \cdot z$

x	y	z	yz	$x+yz$
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

proof by truth table (perfect induction)

to demonstrate the equivalence of two expressions, e.g.,

$$x \cdot (y + z) = x \cdot y + x \cdot z \qquad (\textit{distributivity})$$

a truth table can be constructed

x	y	z	$y+z$	$x(y+z)$	xy	xz	$xy+xz$
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1
				↑			↑

all of the basic properties of Boolean algebra can be proved easily this way

Boolean algebra basic properties

$$x + x = x, \quad x \cdot x = x \qquad \text{(\textit{idempotency})}$$

$$x + 0 = x, \quad x \cdot 1 = x \qquad \text{(\textit{identity})}$$

$$x + 1 = 1, \quad x \cdot 0 = 0 \qquad \text{(\textit{domination})}$$

$$x + y = y + x, \quad x \cdot y = y \cdot x \qquad \text{(\textit{commutativity})}$$

$$(x + y) + z = x + (y + z), \quad (x \cdot y) \cdot z = x \cdot (y \cdot z) \qquad \text{(\textit{associativity})}$$

$$x + x' = 1, \quad x \cdot x' = 0 \qquad \text{(\textit{complementation})}$$

$$x \cdot (y + z) = x \cdot y + x \cdot z, \quad x + y \cdot z = (x + y) \cdot (x + z) \qquad \text{(\textit{distributivity})}$$

algebraic manipulation of Boolean expressions

the basic properties can be used to rewrite Boolean functions

e.g., to demonstrate the property

$$x + (x \cdot y) = x \qquad \textit{(absorption)}$$

$$\begin{aligned} x + (x \cdot y) &= x1 + xy && \textit{(identity)} \\ &= x(1 + y) && \textit{(distributivity)} \\ &= x1 && \textit{(domination)} \\ &= x && \textit{(identity)} \end{aligned}$$

note that Boolean algebra has no inverse operations

- unlike normal algebra, *cancellations are not allowed*

$$x + y = x + z \implies y = z$$

duality and De Morgan's laws

notice what happens if you invert the meaning of 'true' and 'false'

normal logic			inverted logic			
x	y	$x+y$	x'	y'	$x' \cdot y'$	$(x' \cdot y')'$
0	0	0	1	1	1	0
0	1	1	1	0	0	1
1	0	1	0	1	0	1
1	1	1	0	0	0	1

'or' is identical to 'and' when using *inverted logic*

principle of duality:

- replace 0 and 1 by 1 and 0, respectively
- swap the OR and AND operation
- the truth table is the same

this leads to *De Morgan's laws*

$$(x+y)' = x' \cdot y'$$
$$(x \cdot y)' = x' + y'$$

Boolean functions

a function is a mapping

- from each possible combination of input value(s)
- to a unique output value

a Boolean function maps one or more Boolean input values to a Boolean output value

for n Boolean input values, there are 2^n possible combinations

a 3-input function f can be completely specified with an 8-row truth table

x	y	z	$f(x,y,z)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

canonical forms

the truth table for a function f is its *canonical form*

- it completely describes the behaviour of the function

previously we have used truth tables to visualise the behaviour of expressions

if we know the behaviour of a function (its canonical form) we can go the other way

- a truth table lets us reconstruct the expression corresponding to a Boolean function

every row in the table corresponds to a unique combination of inputs

- it is one *term* in the expression: a product of the inputs on which the output depends
- if an input is specified as 1, it appears uncomplemented in the product
- if an input is specified as 0, it appears complemented in the product

the sum of all the terms for which $f = 1$ is the expression for the function

x	y	$f(x,y)$	*term*
0	0	0	
0	1	1	$x' \cdot y$
1	0	1	$x \cdot y'$
1	1	0	

$$f(x,y) = x'y + xy'$$

simplification of Boolean expressions

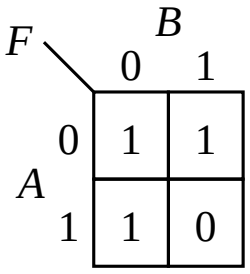
algebraically, by applying the properties described above

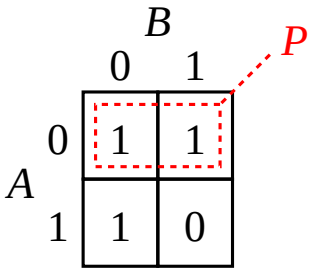
- e.g., simplify $x + x'y$

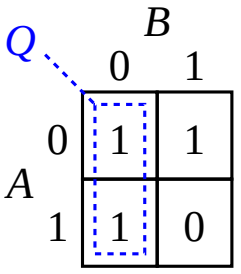
$$
\begin{aligned}
x + x'y &= (x + x')(x + y) && \textit{(distributivity, complementation)} \\
&= 1(x + y) && \textit{(complementation)} \\
&= x + y && \textit{(identity)}
\end{aligned}
$$

visually, using a two-dimensional truth table to find a function's minimal expression

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







$$P = A'$$
$$Q = B'$$
$$F = A' + B'$$

details...

Karnaugh maps

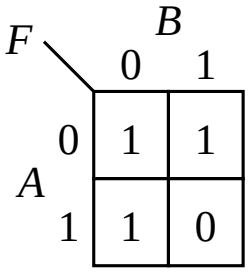
a *Karnaugh map* is a two-dimensinal truth table

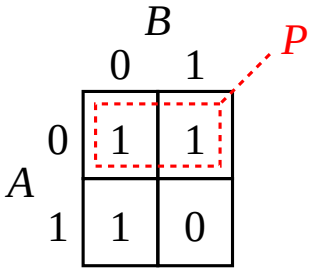
input variables are arranged along the axes

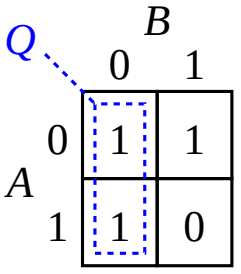
the minimal expression for the map is constructed term-by-term

- pick any '1' which has not been previously considered
- group it with the maximum possible number of adjacent '1's, to form a rectangle
  - the sides of the rectangle ***must*** *be a 'power-of-2' long:* 1, 2, 4, 8, ...
- write a term for this rectangle
  - any input variable completely *covered* (0 and 1) by the rectangle is irrelevant
  - the remaining inputs form the term
    - an input that must be 1 appears uncomplemented
    - an input that must be 0 appears complemented

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







$$P = A'$$
$$Q = B'$$
$$F = A' + B'$$

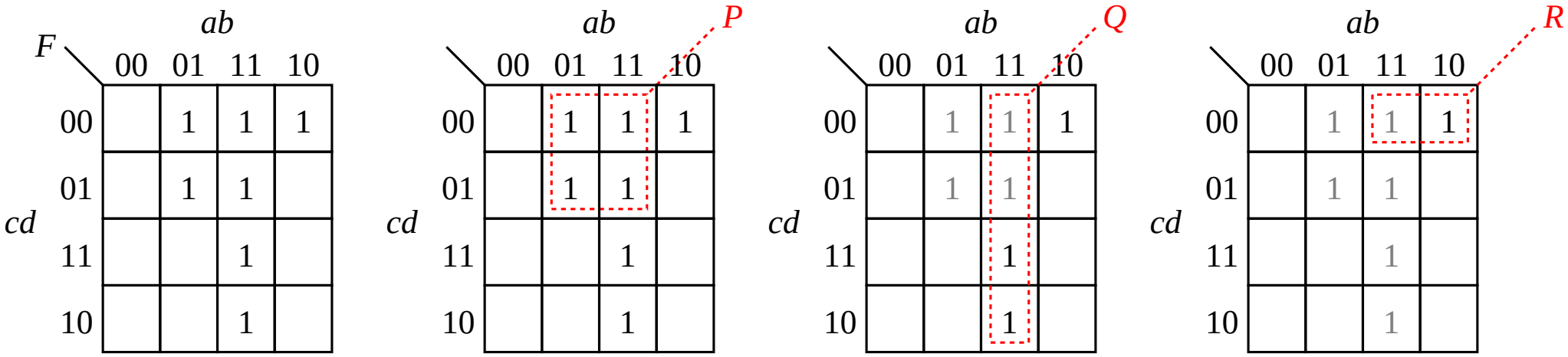
the sum of the resulting terms is the minimal expression for the function

Karnaugh maps

Karnaugh maps are convenient for up to four inputs

along a given axis, adjacent input patterns must differ by only one bit

- i.e., the axes are labelled with a *minimum change code*
- e.g., Gray code



$$P = bc'$$
$$Q = ab$$
$$R = ac'd'$$

$$F = bc' + ab + ac'd'$$

useful and strange Boolean functions

the '*exclusive-or*' (or 'not equivalent') function

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

$$f(x,y) = x'y + xy' \equiv x \neq y \equiv x \not\Leftrightarrow y \equiv x \oplus y$$

- $x \oplus y$ is true if either x or y (but not both) is true
  - i.e., if x and y are *different*
- represents addition, modulo-2
  - very useful in the ALU

the '*implies*' function

x	y	$x \Rightarrow y$
0	0	1
0	1	1
1	0	0
1	1	1

$$f(x,y) = x' + y \equiv x \Rightarrow y$$

- the 'if x then y' function
- note that if the predicate x is false, any consequent y satisfies the relation

next week

combinational digital circuits

signals and busses

logic gates

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

gate-level logical operations

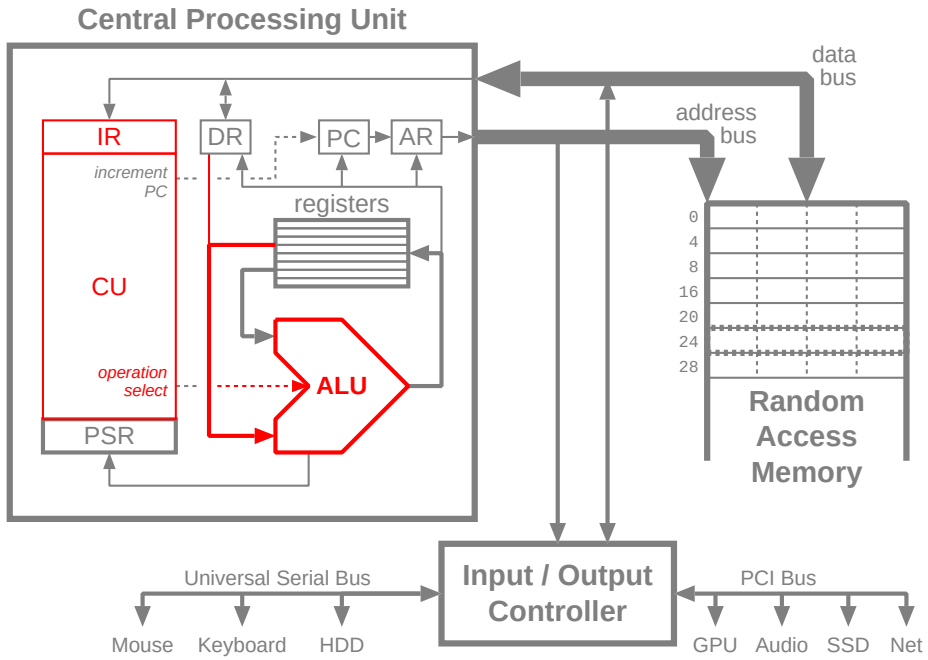
- bitwise: and, or, not

gate-level arithmetic operations

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

1-of-N selection

- multiplexers



homework

practice drawing truth tables for useful functions

- what is the truth table for (three-input) binary sum?
- what is the truth table for (three-input) binary carry?

practice converting truth tables into expressions

- what are the canonical expressions for sum and carry?

practice algebraic simplification

- how simple can you make your sum and carry functions?

practice finding minimal expressions using Karnaugh maps

- what are the minimal expressions for sum and carry?

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

glossary

and — the logical operation '·' in which $x \cdot y$ is true iff x and y are simultaneously true.

associativity — a property of an operation that makes it independent of the order in which repeated applications of it are made. For example, addition is associative because in $2 + 3 + 4$ the result is independent of the order of application of the two additions ($2 + 7$ vs. $5 + 4$).

canonical form — a standard way of writing an expression or function in which every possible combination of inputs are accounted for and have a definite output value.

commutativity — a property of an operation that makes is independent of the order of its operands. For example, addition is commutative because in $3 + 4$ the result is the same as for $4 + 3$.

complementation — the behaviour of an operation when applied to a value and its complement.

complement — the logical opposite of a value. The complement of true is false, and the complement of false is true.

conjunction — when two or more things occur at the same point in time or space.

covered — an input to a function that has no influence on the result.

De Morgan's laws — logical rules arising from the duality of 'and' and 'or' when the logical interpretation of true and false is reversed.

disjunction — when two or more things are distinct (but not necessarily mutually-exclusive) alternatives.

distributivity — a property of two binary functions f and g that ensures $f(x, g(y, z)) = g(f(x, y), f(x, z))$.

domination — a combination of binary function and single operand that yields the same result when applied to any other operand value.

exclusive-or — the ‘not equivalent’ function, true iff its inputs differ.

false — the logical value representing the opposite of true, typically written ‘0’.

idempotency — a value that remains unchanged when operated on by itself.

identity — a value that, when combined with any other value using a specified operation, leaves that value unchanged.

implication — a binary logical operation in which the right hand operand must be true whenever the left hand operand is true.

inclusive-or — the ‘or’ operation, with an explicit indication that it remains true when both inputs are true.

inverted logic — the dual of normal logic, in which 0 represents true and 1 represents false. If all occurences of ‘and’ and ‘or’ are swapped, the rules of inverted logic are identical to the rules of normal logic.

Karnaugh map — a two-dimensional truth table in which adjacent rows or columns differ in exaclt one input. Karnaugh maps immediately identify redundant inputs, facilitating the construction of minimal expressions for any given logical function.

negation — the conversion of a logical value into its complement.

not — the logical operation that negates, or complements, a value.

or — the logical operation '$+$' in which $x + y$ is false iff x and y are simultaneously false.

product — the result of multiplication. In logic, the product of 0 with x is 0, and the product of 1 with x is x.

sum — the result of addition. In logic, the sum of 0 with x is x, and the sum of 1 with x is 1.

term — the expression represented by one line of a truth table, typically written as a product (logical 'and' of input values, complemented as required).

true — the logical value representing the opposite of false, typically written '1'.

truth table — a table listing all possible combinations of input values and their corresponding output value.

xor — the 'exclusive-or' operation, which is true only if exactly one if the inputs is true.