# COMPUTER ORGANISATION
## (TỔ CHỨC MÁY TÍNH)

# Boolean Algebra

# Acknowledgement


- The contents of these slides have origin from School of Computing, National University of Singapore.
- We greatly appreciate support from Mr. Aaron Tan Tuck Choy for kindly sharing these materials.

# Policies for students

- These contents are only used for students PERSONALLY.
- Students are NOT allowed to modify or deliver these contents to anywhere or anyone for any purpose.

# Recording of modifications

- Minor change in slide 13 (replace new picture)
- Currently, there are no modification on these contents.

# WHERE ARE WE NOW?

- Number systems and codes
- Boolean algebra
- Logic gates and circuits
- Simplification
- Combinational circuits
- Sequential circuits
- Performance
- Assembly language
- The processor: Datapath and control
- Pipelining
- Memory hierarchy: Cache
- Input/output

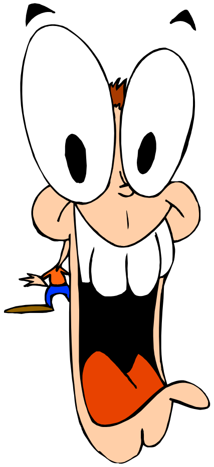
Preparation: 2 weeks

Logic Design: 3 weeks

Computer organisation


# CHECK LIST

- Have you done the *Quick Review Questions* for Chapter 2 Number Systems and Code?
- Have you attempted the *Self-Assessment Exercise #1* on IVLE Assessment?
- Have you clarified your doubts on *IVLE forum*?
- Ready to do a *pop quiz*?



# BOOLEAN ALGEBRA

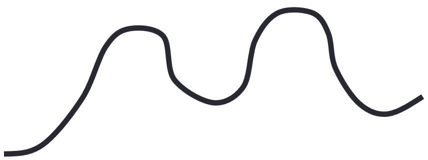
- Boolean Algebra
- Precedence of Operators
- Truth Table
- Duality
- Basic Theorems
- Complement of Functions
- Standard Forms
- Minterms and Maxterms
- Canonical Forms

Read up DLD for details!

# DIGITAL CIRCUITS (1/2)

- Two voltage levels
  - High, true, 1, asserted
  - Low, false, 0, deasserted


*Signals in digital circuit*



*Signals in analog circuit*


*A digital watch*

# DIGITAL CIRCUITS (2/2)

- Advantages of digital circuits over analog circuits
  - More reliable (simpler circuits, less noise-prone)
  - Specified accuracy (determinable)
  - Abstraction can be applied using simple mathematical model – Boolean Algebra
  - Ease design, analysis and simplification of digital circuit – Digital Logic Design

# TYPES OF LOGIC BLOCKS

- Combinational: no memory, output depends solely on the input
  - Gates
  - Decoders, multiplexers
  - Adders, multipliers
- Sequential: with memory, output depends on both input and current state
  - Counters, registers
  - Memories

# BOOLEAN ALGEBRA

- Boolean values:
  - True (1)
  - False (0)
- Connectives
  - Conjunction (AND)
    - A · B; A ∧ B
  - Disjunction (OR)
    - A + B; A ∨ B
  - Negation (NOT)
    - $\overline{A}$; ¬A; A'

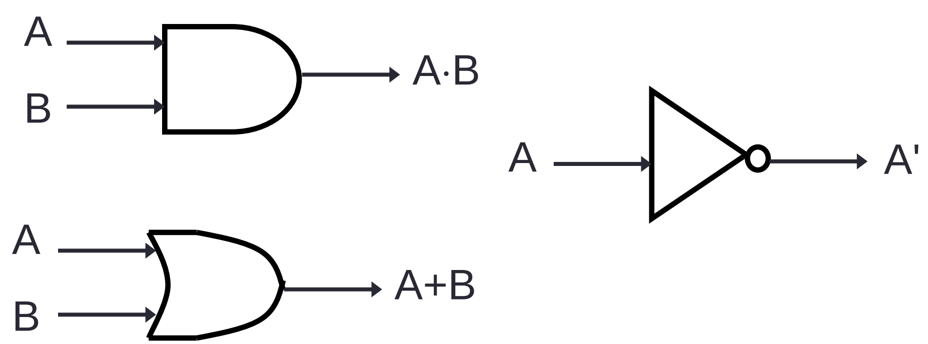
## Truth tables

| A | B | A · B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

| A | B | A + B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

| A | A' |
|---|---|
| 0 | 1 |
| 1 | 0 |

## Logic gates



# AND (·)

- Do write the AND operator · instead of omitting it.
  - Example: Write ***a·b*** instead of ***ab***
  - Why? Writing *ab* could mean it is a 2-bit value.

# LAWS OF BOOLEAN ALGEBRA

- Identity laws

  $A + 0 = 0 + A = A$ ; $A \cdot 1 = 1 \cdot A = A$

- Inverse/complement laws

  $A + A' = 1$ ; $A \cdot A' = 0$

- Commutative laws

  $A + B = B + A$ ; $A \cdot B = B \cdot A$

- Associative laws

  $A + (B + C) = (A + B) + C$ ; $A \cdot (B \cdot C) = (A \cdot B) \cdot C$

- Distributive laws

  $A \cdot (B + C) = (A \cdot B) + (A \cdot C)$ ; $A + (B \cdot C) = (A + B) \cdot (A + C)$

# PRECEDENCE OF OPERATORS

- Precedence from highest to lowest
  - Not
  - And
  - Or
- Examples:
  - $A \cdot B + C = (A \cdot B) + C$
  - $X + Y' = X + (Y')$
  - $P + Q' \cdot R = P + ((Q') \cdot R)$
- Use parenthesis to overwrite precedence. Examples:
  - $A \cdot (B + C)$
  - $(P + Q)' \cdot R$

# TRUTH TABLE

- Provide a listing of every possible combination of inputs and its corresponding outputs.
  - Inputs are usually listed in binary sequence.
- Example
  - Truth table with 3 inputs and 2 outputs

| x | y | z | y + z | x · (y + z) |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |

# PROOF USING TRUTH TABLE

- Prove: $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$
  - Construct truth table for LHS and RHS

| x | y | z | y + z | x · (y + z) | x · y | x · z | (x · y) + (x · z) |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | | | | | |
| 0 | 0 | 1 | | | | | |
| 0 | 1 | 0 | | | | | |
| 0 | 1 | 1 | | | | | |
| 1 | 0 | 0 | | | | | |
| 1 | 0 | 1 | | | | | |
| 1 | 1 | 0 | | | | | |
| 1 | 1 | 1 | | | | | |

- Check that column for LHS = column for RHS

# DUALITY

- If the AND/OR operators and identity elements 0/1 in a Boolean equation are interchanged, it remains valid
- Example:
  - The dual equation of $a+(b\cdot c)=(a+b)\cdot(a+c)$ is $a\cdot(b+c)=(a\cdot b)+(a\cdot c)$
- Duality gives free theorems – “two for the price of one”. You prove one theorem and the other comes for free!
- Examples:
  - If $(x+y+z)' = x'\cdot y'\cdot z'$ is valid, then its dual is also valid: $(x\cdot y\cdot z)' = x'+y'+z'$
  - If $x+1 = 1$ is valid, then its dual is also valid: $x\cdot 0 = 0$

# BASIC THEOREMS (1/2)

1. Idempotency

   $X + X = X$ ; $X \cdot X = X$

2. Zero and One elements

   $X + 1 = 1$ ; $X \cdot 0 = 0$

3. Involution

   $( X' )' = X$

4. Absorption

   $X + X \cdot Y = X$ ; $X \cdot (X + Y) = X$

5. Absorption (variant)

   $X + X' \cdot Y = X + Y$ ; $X \cdot (X' + Y) = X \cdot Y$

# BASIC THEOREMS (2/2)

6. **DeMorgan's**

   $(X + Y)' = X' \cdot Y'$ ; $(X \cdot Y)' = X' + Y'$

   DeMorgan's Theorem can be generalised to more than two variables, example: $(A + B + \ldots + Z)' = A' \cdot B' \cdot \ldots \cdot Z'$

7. **Consensus**

   $X \cdot Y + X' \cdot Z + Y \cdot Z = X \cdot Y + X' \cdot Z$

   $(X+Y) \cdot (X'+Z) \cdot (Y+Z) = (X+Y) \cdot (X'+Z)$

# PROVING A THEOREM

- Theorems can be proved using truth table, or by algebraic manipulation using other theorems/laws.
- Example: Prove absorption theorem $X + X \cdot Y = X$

  $X + X \cdot Y = X \cdot 1 + X \cdot Y$ (by identity) $= X \cdot (1+Y)$ (by distributivity)
  $= X \cdot (Y+1)$ (by commutativity) $= X \cdot 1$ (by one element)
  $= X$ (by identity)

- By duality, we have also proved $X \cdot (X+Y) = X$

# BOOLEAN FUNCTIONS

- Examples of Boolean functions (logic equations):

  $F1(x,y,z) = x \cdot y \cdot z'$

  $F2(x,y,z) = x + y' \cdot z$

  $F3(x,y,z) = x' \cdot y' \cdot z + x' \cdot y \cdot z + x \cdot y'$

  $F4(x,y,z) = x \cdot y' + x' \cdot z$

| x | y | z | F1 | F2 | F3 | F4 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | | | |
| 0 | 0 | 1 | 0 | | | |
| 0 | 1 | 0 | 0 | | | |
| 0 | 1 | 1 | 0 | | | |
| 1 | 0 | 0 | 0 | | | |
| 1 | 0 | 1 | 0 | | | |
| 1 | 1 | 0 | 1 | | | |
| 1 | 1 | 1 | 0 | | | |

# COMPLEMENT

- Given a Boolean function F, the complement of F, denoted as F', is obtained by interchanging 1 with 0 in the function's output values.
- Example: $F1 = x \cdot y \cdot z'$
- What is F1' ?

| x | y | z | F1 | F1' |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | |
| 0 | 0 | 1 | 0 | |
| 0 | 1 | 0 | 0 | |
| 0 | 1 | 1 | 0 | |
| 1 | 0 | 0 | 0 | |
| 1 | 0 | 1 | 0 | |
| 1 | 1 | 0 | 1 | |
| 1 | 1 | 1 | 0 | |

# STANDARD FORMS (1/2)

- Certain types of Boolean expressions lead to circuits that are desirable from implementation viewpoint.
- Two standard forms:
  - Sum-of-Products
  - Product-of-Sums
- Literals
  - A Boolean variable on its own or in its complemented form
  - Examples: $x$, $x'$, $y$, $y'$
- Product term
  - A single literal or a logical product (AND) of several literals
  - Examples: $x$, $x \cdot y \cdot z'$, $A' \cdot B$, $A \cdot B$, $d \cdot g' \cdot v \cdot w$

# STANDARD FORMS (2/2)

- Sum term
  - A single literal or a logical sum (OR) of several literals
  - Examples: x, x+y+z', A'+B, A+B, c+d+h'+j
- Sum-of-Products (SOP) expression
  - A product term or a logical sum (OR) of several product terms
  - Examples: x, x + y·z', x·y' + x'·y·z, A·B + A'·B',
    A + B'·C + A·C' + C·D
- Product-of-Sums (POS) expression
  - A sum term or a logical product (AND) of several sum terms
  - Examples: x, x·(y+z'), (x+y')·(x'+y+z),
    (A+B)·(A'+B'), (A+B+C)·D'·(B'+D+E')
- Every Boolean expression can be expressed in SOP or POS.

# DO IT YOURSELF

**SOP** expr: A product term or a logical sum (OR) of several product terms.
**POS** expr: A sum term or a logical product (AND) of several sum terms.

- Put the right ticks in the following table.

| ***Expression*** | ***SOP?*** | ***POS?*** |
|---|---|---|
| X'·Y + X·Y' + X·Y·Z | | |
| (X+Y')·(X'+Y)·(X'+Z') | | |
| X' + Y + Z | | |
| X·(W' + Y·Z) | | |
| X·Y·Z' | | |
| W·X'·Y + V·(X·Z + W') | | |

# MINTERMS & MAXTERMS (1/2)

- A minterm of *n* variables is a product term that contains *n* literals from all the variables.
  - Example: On 2 variables x and y, the minterms are: $x' \cdot y'$, $x' \cdot y$, $x \cdot y'$ and $x \cdot y$
- A maxterm of *n* variables is a sum term that contains *n* literals from all the variables.
  - Example: On 2 variables x and y, the maxterms are: $x'+y'$, $x'+y$, $x+y'$ and $x+y$
- In general, with *n* variables we have $2^n$ minterms and $2^n$ maxterms.

# MINTERMS & MAXTERMS (2/2)

- The minterms and maxterms on 2 variables are denoted by m0 to m3 and M0 to M3 respectively.

| x | y | Minterms | | Maxterms | |
|---|---|---|---|---|---|
| | | Term | Notation | Term | Notation |
| 0 | 0 | x'·y' | m0 | x+y | M0 |
| 0 | 1 | x'·y | m1 | x+y' | M1 |
| 1 | 0 | x·y' | m2 | x'+y | M2 |
| 1 | 1 | x·y | m3 | x'+y' | M3 |

- Each minterm is the complement of the corresponding maxterm
  - Example: m2 = x·y'
    m2' = ( x·y' )' = x' + ( y' )' = x' + y = M2

# CANONICAL FORMS

- Canonical/normal form: a unique form of representation.
  - Sum-of-minterms = Canonical sum-of-products
  - Product-of-maxterms = Canonical product-of-sums

# SUM-OF-MINTERMS

- Given a truth table, example:

| x | y | z | F1 | F2 | F3 |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 |

- Obtain sum-of-minterms expression by gathering the minterms of the function (where output is 1).

F1 = x·y·z' = m6

F2 =

F3 =

# PRODUCT-OF-MAXTERMS

- Given a truth table, example:

- Obtain **product-of-maxterms** expression by gathering the maxterms of the function (where output is 0).

| x | y | z | F1 | F2 | F3 |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 |

$F2 = (x+y+z) \cdot (x+y'+z) \cdot (x+y'+z')$
$\quad = M0 \cdot M2 \cdot M3 = \Pi M(0,2,3)$

F3 =

# CONVERSION

- We can convert between sum-of-minterms and product-of-maxterms easily
- Example: F2 = $\Sigma$m(1,4,5,6,7) = $\Pi$M(0,2,3)
- Why? See F2' in truth table.

| x | y | z | F2 | F2' |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 |

- F2' = m0 + m2 + m3
  Therefore,
  F2 = (m0 + m2 + m3)'
  = m0' · m2' · m3' (by DeMorgan's)
  = M0 · M2 · M3 (m*x*' =M*x*)

# READING ASSIGNMENT

- Conversion of Standard Forms
  - Read up DLD section 3.4, pg 57 – 58.

# Q&A