Disclaimer

The slides presented here are a combination of the CS251 course notes from previous terms, the work of Xiao-Bo Li, and material from the required textbook "Computer Organization and Design, ARM Edition," by David A. Patterson and John L. Hennessy. It is being used here with explicit permission from the authors.

CS251 course policy requires students to delete all course files after the term. Therefore, please do not post these slides to any website or share them.

CS251 - Computer Organization and Design

Intro to Digital Logic Design - Combinational Logic Design

Instructor: Zille Huma Kamal

University of Waterloo

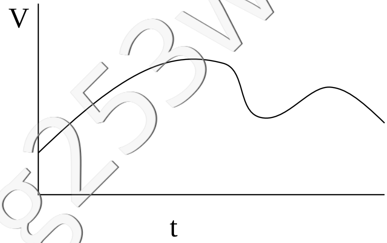
Spring 2023

A Brief Look at Electricity

- Computers work with current/voltage

$$V = IR$$

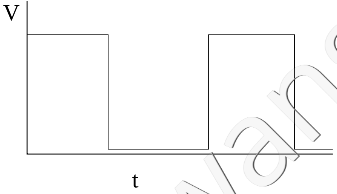
- These quantities are continuous
  Plot: voltage vs time



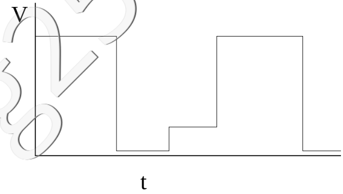
- They can be manipulated, but accuracy is difficult

Digitizing

- Discrete Signal



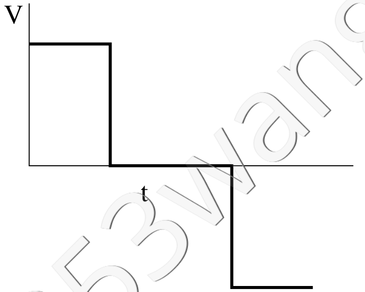
- Signal is either high (1) or low (0)
- Transformation could lead to intermediate values



but these can be “designed out”

Why Binary?

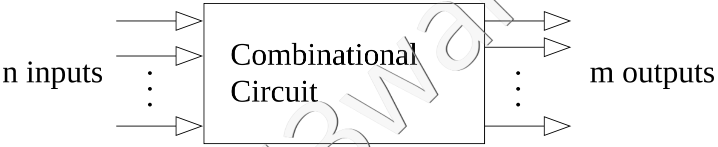
- Could have more levels...



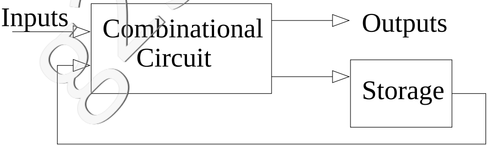
- Plus, Zero, Minus ($+1$, 0, -1 — ternary)
- Two levels are simpler and just as expressive
- Nearly all computers today use binary
- Nearly all computers have similar underlying structure

Logic Blocks

- Inputs and outputs are 1/0
  (High/low voltage, true/false)
- Combinational: without memory



- Sequential: with memory



Specifying input/output behaviour

- Truth table: specifies outputs for each possible input combination

X	Y	Z	F	G
0	0	0	0	1
0	0	1	1	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	0

- Complete description, but big and hard to understand

Compact alternative: Boolean algebra

- Variables (usually A, B, C or X, Y, Z) have values 0 or 1
- OR $(+)$ operator has result 1 iff either operand has value 1
- AND $(\cdot)$ operator has result 1 iff both operands have value 1
  $A \cdot B$ often written AB
- NOT $(\neg)$ operator has result 1 iff operand has value 0
  $\neg A$ usually written $\bar{A}$

OR		
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

AND		
A	B	AB
0	0	0
0	1	0
1	0	0
1	1	1

NOT	
A	¬A
0	1
1	0

Compact alternative: Example using Boolean Algebra

Try this

Write the function F and G in Boolean algebra.

X	Y	Z	F	G
0	0	0	0	1
0	0	1	1	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	0

Solution: Boolean algebra

Try this

Write the function F and G in Boolean algebra.

X	Y	Z	F	G
0	0	0	0	1
0	0	1	1	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	0

- It is clear that $G = \overline{XYZ}$
- However, it is not clear that $F = \bar{X}\bar{Y}Z + X\bar{Y}Z + XY\bar{Z} + XYZ$

Truth Table to Formula Using Minimal Terms

A	B	C	F	$\bar{A}\bar{B}C$	$A\bar{B}C$	ABC	$\bar{A}\bar{B}C + A\bar{B}C + ABC$
0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	1
0	1	0	0	0	0	0	0
0	1	1	0	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	0	1	0	1
1	1	0	0	0	0	0	0
1	1	1	1	0	0	1	1

Example: Derive Formula Using Minimal Terms from Truth Table

Try this

Write the function F in Boolean algebra.

X	Y	Z	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Two-Level Representations

- Any Boolean function can be represented as a sum of products (OR of ANDs) of literals
- Each term in sum corresponds to a single line in truth table with value 1
- This can be simplified by hand or by machine
- An alternative representation is product of sums that may also be useful

Example: Derive formula

Consider the following truth table:

X	Y	F
0	0	1
0	1	0
1	0	1
1	1	1

Try this

Write the function F in Boolean algebra using product of sums.

Try this

Write the function F in Boolean algebra using sum of products.

Don't Cares

- Represented as X instead of 0 or 1
- When used in output, indicates that we don't care what output is for that input
- When used in input, indicates outputs are valid for all inputs created by replacing X by 0 or 1 (useful in compressing truth tables)
- Example: The two rows below gives the same output value for F:

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

The don't care for **input** C simplifies this table

A	B	C	F
0	0	X	0

Becareful with Don't Cares

The following is not a valid truth table.

A	B	C	F
0	0	X	0
0	X	1	1
1	X	X	X

Here is what's wrong:

- 001 appears in two rows with different outputs. A truth table must have exactly one output for each input.
- 010 is missing in the input. Each possible input must be specified exactly once.

Example: Formula for Truth Tables with Don't Cares

Try this

Write the function for F

A	B	C	F
0	0	X	0
0	1	X	1
1	X	X	X

Compressed Truth Tables and Non-Minimal Terms

Try this

Write the function for F

A	B	C	F	$\bar{A}\bar{B}C$	AC	$\bar{A}\bar{B}C + AC$
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	X	0	0	0	0
1	X	0	0	0	0	0
1	X	1	1	0	1	1

Using Overlapping Non-Minimal Terms

Try this

Write the function for F

A	B	C	F	AB	AC	$AB + AC$
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	0	0	0	0
1	0	1	1	0	1	1
1	1	0	1	1	0	1
1	1	1	1	1	1	1

Laws of Boolean Algebra

Rule	Dual Rule	
$\bar{\bar{X}} = X$		
$X + 0 = X$	$X \cdot 1 = X$	(identity)
$X + 1 = 1$	$X \cdot 0 = 0$	(zero/one)
$X + X = X$	$XX = X$	(absorption)
$X + \bar{X} = 1$	$X\bar{X} = 0$	(inverse)
$X + Y = Y + X$	$XY = YX$	(commutative)
$X + (Y + Z) = (X + Y) + Z$	$X(YZ) = (XY)Z$	(associative)
$X(Y + Z) = XY + XZ$	$X + YZ = (X + Y)(X + Z)$	(distributive)
$\overline{X + Y} = \overline{X} \cdot \overline{Y}$	$\overline{XY} = \overline{X} + \overline{Y}$	(DeMorgan)

Formula Simplification Using Laws

- We can use algebraic manipulation (based on laws) to simplify formulas.
- For example:

X	Y	Z	F	G
0	0	0	0	1
0	0	1	1	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	0

$$
\begin{aligned}
F &= \bar{X}\bar{Y}Z + X\bar{Y}Z + XY\bar{Z} + XYZ \\
&= \bar{Y}Z(\bar{X} + X) + XY(\bar{Z} + Z) \\
&= \bar{Y}Z + XY
\end{aligned}
$$

- Difficult even for humans, tricky to automate
- Seems inherently hard to get "simplest" formula
- Is simplest formula the best for implementation?

Example: Simplified Expression for Output Function

Try this

Write a simplified boolean equation for output function F:

A	B	C	F	$\bar{A}\bar{B}C$	$A\bar{B}C$	ABC	$\bar{A}\bar{B}C + A\bar{B}C + ABC$
0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	1
0	1	0	0	0	0	0	0
0	1	1	0	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	0	1	0	1
1	1	0	0	0	0	0	0
1	1	1	1	0	0	1	1

Solution: Using Don't Cares

A	B	C	F	$\bar{A}\bar{B}C$	AC	$\bar{A}\bar{B}C+AC$
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	X	0	0	0	0
1	X	0	0	0	0	0
1	X	1	1	0	1	1

Solution: Further Simplification

The original sum of products expression

$$F = \bar{A}\bar{B}C + A\bar{B}C + ABC$$

was simplified to

$$F = \bar{A}\bar{B}C + AC$$

But this expression looks like it can be simplified further since $\bar{A}$ and A shows up in two minterms.

A	B	C	F	$\bar{A}\bar{B}C$	AC	$\bar{A}\bar{B}C + AC$
0	0	0	0	0	0	0
$\boxed{0}$	0	1	1	1	0	1
0	1	X	0	0	0	0
1	X	0	0	0	0	0
$\boxed{1}$	X	1	1	0	1	1

Solution: Using Laws of Boolean Algebra

Indeed,

$$\begin{aligned} F &= \bar{A}\bar{B}C + A\bar{B}C + ABC \\ &= \bar{A}\bar{B}C + A\bar{B}C + A\bar{B}C + ABC \\ &= (\bar{A} + A)\bar{B}C + A(\bar{B} + \bar{B})C \\ &= \bar{B}C + AC \end{aligned}$$

Thus, if $\bar{X}$ appears in one minterm, and X appears in another, try to simplify further.

Textbook Readings

- Readings: Section A.2 from the textbook