WHY DO COMPUTER SYSTEMS MATTER?

- Implement component-level feedback control
- Implement system-level supervisory control
- Signal processing
- Communication with other systems
- Easily modified through software

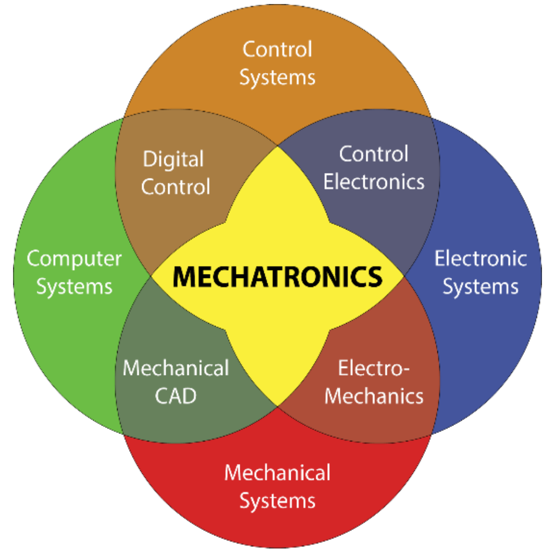
We cover basics of computer system operation in UNIT 3 and UNIT 4

UNIT 3: COMBINATIONAL LOGIC

TOPICS

- Part A: Boolean Algebra
- Part B: Boolean Simplification (K-map)
- Part C: Implementing Boolean Logic
- Part D: Digital Interfacing

COMBINATIONAL LOGIC



At the end of this section, students should be able to:

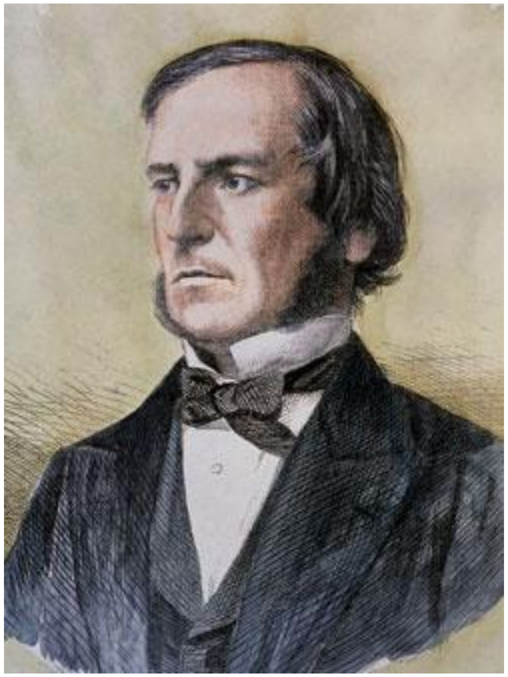
- Explain the value of Computer Systems to Mechatronics
- Write simple logical statements using Boolean algebra
- Perform graphical minimization on combinational logic
- Implement combinational logic using ICs

UNIT 3:
COMBINATIONAL LOGIC

PART A:
BOOLEAN ALGEBRA

BOOLEAN ALGEBRA INVENTED BY GEORGE BOOLE

George Boole – English mathematician (1815-1864)



BOOLEAN VARIABLES MAY ASSUME TWO (AND ONLY TWO) VALUES

Typical symbol pairs to represent Boolean values:

- 0/1
- TRUE/FALSE
- ON/OFF
- OPEN/CLOSE
- HI/LO

GENERAL SYSTEMS MAY BE CONSIDERED 'STATIC' OR 'DYNAMIC'

Static Systems

- Outputs depends on the current values of the inputs
- Described by ***algebraic*** equations
- Can be implemented as a look-up table (LUT)

Dynamic Systems

- Outputs depends on the current and past values of the inputs
- Described by differential equations (calculus)

BOOLEAN BEHAVIOR MAY ALSO BE ‘STATIC’ OR ‘DYNAMIC’

Static Behavior

- Described by **combinational logic** (**Unit 3**)
- Subject to mathematical tools of Boolean algebra

Dynamic Behavior

- Described by **sequential logic** (**Unit 4**)
- Developed *ad hoc* (no formal calculus for binary systems)

BINARY REPRESENTATION USES ONES AND ZEROS

Expression of numerical values using a base-2 system (0/1) is called a ***binary*** representation.

$$5_{10} = 101_2$$

Boolean logic ***is not*** dependent upon using binary values to represent Boolean values. However, it is quite convenient for use in digital computing devices.

DIGITAL DATA

Digital:

1s and 0s, $(1001\ 1011)_2$

Advantages

- Less susceptible to noise
- Easy to manipulate using a computer

Disadvantages

- Finite precision
- Sample loses information
- Time lag associated with sample and hold

Analog:

3.141592687..., 1/3

Advantages

- Exact – infinite resolution
- Instantaneous

Disadvantages

- Noise
- Repeatability
- Difficult to manipulate

BOOLEAN ALGEBRA

NOT ($\bar{x}$ or x' or $/x$)

- Truth Table:

$\boldsymbol{x}$	$\boldsymbol{\bar{x}}$
0	1
1	0

- Map:

	$\boldsymbol{x}$ = 0	1
$\boldsymbol{\bar{x}}$	1	0

BOOLEAN ALGEBRA

AND (•)

- Truth Table:

x	y	$x \bullet y$
0	0	0
0	1	0
1	0	0
1	1	1

- Map:

x AND y

		x	
		0	1
y	0	0	0
	1	0	1

BOOLEAN ALGEBRA

OR (+)

- Truth Table:

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

- Map:

x **OR** y

		x = 0	x = 1
y	0	0	1
	1	1	1

BOOLEAN ALGEBRA

XOR ($\oplus$) "Exclusive OR"

- Truth Table:

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

- Map:

x **XOR** y

		x = 0	x = 1
y	0	0	1
	1	1	0

BOOLEAN ALGEBRA EQUATIONS

- Write equations in the "normal" way
- Hierarchy and parentheses usage borrowed from "ordinary" algebra
- Unlike "ordinary" algebra, the AND and OR operators are dual operators – if one substitutes 0 for 1, and also substitutes AND for OR, the result is unchanged

BOOLEAN ALGEBRA AXIOMS

(Axioms define domain characteristics, from which other 'truths' can be derived.)

- Commutativity: $x + y = y + x$
  $x \cdot y = y \cdot x$
- Associativity: $(x + y) + z = x + (y + z)$
  $(x \cdot y) \cdot z = x \cdot (y \cdot z)$
- Distributivity: $x \cdot (y + z) = x \cdot y + x \cdot z$
  $x + (y \cdot z) = (x + y) \cdot (x + z)$
- Existence of 0 and 1: $x + 0 = x$
  $x \cdot 1 = x$
- Existence of complements: $x \cdot \bar{x} = 0$
  $x + \bar{x} = 1$

BOOLEAN ALGEBRA THEOREMS

(Theorems are proven through algebraic manipulation or exhaustive substitution.)

- Idempotent Operators:
$$x + x = x$$
$$x \cdot x = x$$

- Absorption:
$$x + x \cdot y = x$$
$$x \cdot (x + y) = x$$

- Simplification:
$$x + \bar{x} \cdot y = x + y$$
$$x \cdot (\bar{x} + y) = x \cdot y$$

- DeMorgan's Law:
$$\overline{(x + y)} = \bar{x} \cdot \bar{y}$$
$$\overline{(x \cdot y)} = \bar{x} + \bar{y}$$

COMBINATIONAL LOGIC DESIGN

Use Boolean algebra to go from design specification to implementable logic function

Boolean algebra can be used to design systems that have:

- Binary inputs
- Binary outputs
- No history dependence or memory

EXAMPLE: ELECTRONIC DOOR LOCK

- The lock has a set of buttons. To enter (open the lock), one must simultaneously press the correct combination of buttons.
- Inputs: buttons B1 and B2, Button-IN = 1, Button-OUT = 0
- Output: Lock actuator (L), 0 = Lock, 1 = Unlock
- Truth Table: (two button lock)

B1	*B2*	*L*
0	0	**0**
0	1	**0**
1	0	**1**
1	1	**0**

SUM OF PRODUCTS

$B1$	$B2$	L
0	0	**0**
0	1	**0**
1	0	**1**
1	1	**0**

Generate an algebraic expression for desired function from a truth table by creating a **sum-of-products** expression:

- Form a sum-of-products function:

  $$y = (\boldsymbol{C}_1) + (\boldsymbol{C}_2) + \cdots + (\boldsymbol{C}_{n-1}) + (\boldsymbol{C}_n)$$

  where each $\boldsymbol{C}_i$ term expresses one of n possible combinations of the input variables, or their complements, as Boolean products. Treat ones in the truth table as uncomplimented variables, and zeros as complimented variables.
- Drop terms corresponding to an output of zero in the truth table.

SUM OF PRODUCTS

All Boolean design flows from this procedure. It is the Boolean equivalent of regression, but it is exact!

EXAMPLE: TWO BUTTON DOOR LOCK

$B1$	$B2$	L
0	0	**0**
0	1	**0**
1	0	**1**
1	1	**0**

- Form a sum-of-products function:

$$y = (\overline{B1} \cdot \overline{B2}) + (\overline{B1} \cdot B2) + (B1 \cdot \overline{B2}) + (B1 \cdot B2)$$

- Drop terms associated with an output of zero in the truth table:

$$y = \cancel{(\overline{B1} \cdot \overline{B2})} + \cancel{(\overline{B1} \cdot B2)} + (B1 \cdot \overline{B2}) + \cancel{(B1 \cdot B2)}$$

$$y = (B1 \cdot \overline{B2})$$

PRODUCT OF SUMS

$B1$	$B2$	L
0	0	**0**
0	1	**0**
1	0	**1**
1	1	**0**

Complementary to Sum-of-Products

- All rules are dual

Example: Two button door lock

- Form product of sums function:
  $$y = (\overline{B1} + \overline{B2}) \cdot (\overline{B1} + B2) \cdot (B1 + \overline{B2}) \cdot (B1 + B2)$$
- Drop terms with truth table output of 1:

$$y = (\overline{B1} + \overline{B2}) \cdot (\overline{B1} + B2) \cdot \cancel{(B1 + \overline{B2})} \cdot (B1 + B2)$$

$$y = (\overline{B1} + \overline{B2}) \cdot (B1 + \overline{B2}) \cdot (B1 + B2)$$

In this example, the product-of-sums expression is more complex than the sum-of-products.

- Complexity is problem dependent
- Function minimization can be done algebraically and graphically.

COMING UP…

Combinational Logic

- Minimizing combinational logic
- Implementing combinational logic

Then, we investigate sequential logic…