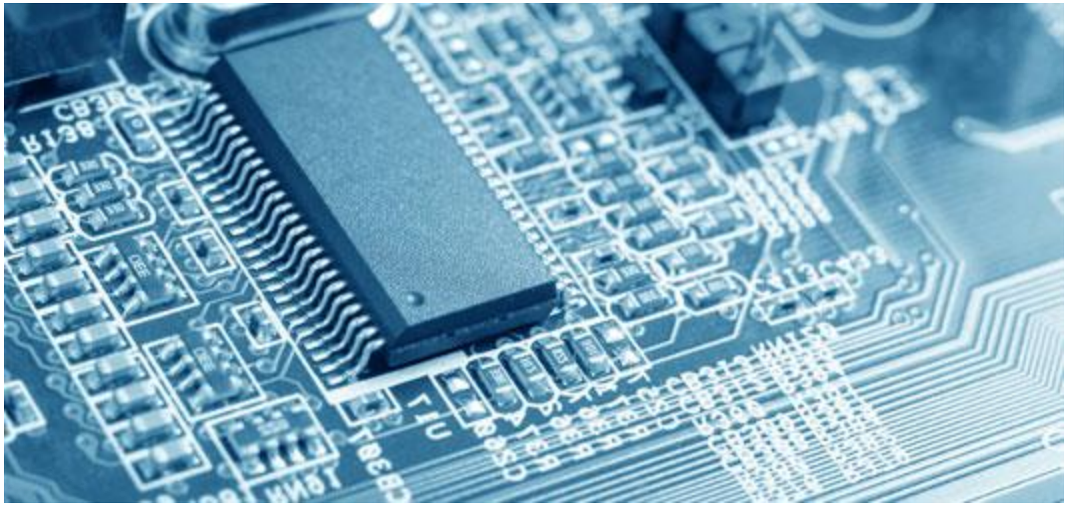
NAZARBAYEV UNIVERSITY
SCHOOL OF SCIENCE AND TECHNOLOGY



CSCI502 – Hardware/Software Co-Design

Self-Study Lecture 1– Number Systems, Binary Logic and Gates

9 January, 2019

Course Logistics

Reference Reading (available in Moodle and Library):
Logic and Computer Design Fundamentals. 5th edition: Chapters 1 – 2 (slide relevant material)
Exploring BeagleBone. 2nd edition: Chapter 4

Number Systems: Common Bases

Name	Base	Digits
Binary	2	0,1
Octal	8	0,1,2,3,4,5,6,7
Decimal	10	0,1,2,3,4,5,6,7,8,9
Hexadecimal	16	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Numbers in Different Bases

- Good idea to memorize!

Decimal (Base 10)	Binary (Base 2)	Octal (Base 8)	Hexadecimal (Base 16)
00	00000	00	00
01	00001	01	01
02	00010	02	02
03	00011	03	03
04	00100	04	04
05	00101	05	05
06	00110	06	06
07	00111	07	07
08	01000	10	08
09	01001	11	09
10	01010	12	0A
11	01011	13	0B
12	01100	14	0C
13	01101	15	0D
14	01110	16	0E
15	01111	17	0F
16	10000	20	10

Converting Binary to Decimal

- To convert to decimal, use decimal arithmetic to form Σ (digit × respective power of 2).

• example: $1011.1 = (1011.1)_2$

$$
\begin{array}{ccccccccc}
1 & & 0 & & 1 & & 1 & . & 1 \\
\times 2^{3} & & \times 2^{2} & & \times 2^{1} & & \times 2^{0} & & \times 2^{-1} \\
\hline
8 & + & 0 & + & 2 & + & 1 & + & .5
\end{array}
= (11.5)_{10}
$$

- Convert 11010_2 to N_{10}:

Converting Hexadecimal to Decimal

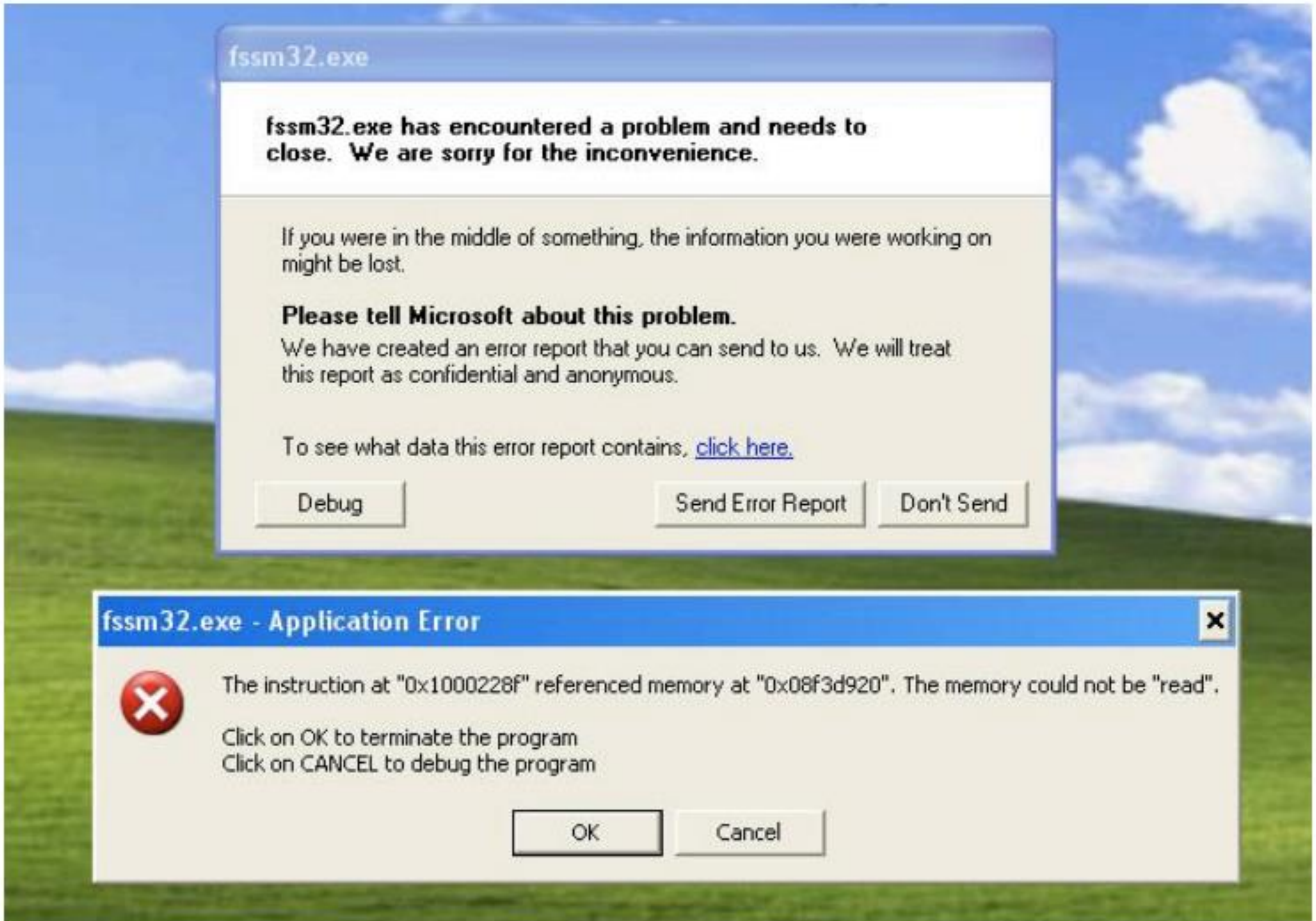
- 16 digits = {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}

- example: $(26BA)_{16}$ [alternate notation for hex: 0x26BA]

$$
\begin{array}{ccccccccc}
2 & & 6 & & B & & A & & \\
\times\ 16^{3} & & \times\ 16^{2} & & \times\ 16^{1} & & \times\ 16^{0} & & \\
\hline
8192 & + & 1536 & + & 176 & + & 10 & = & (9914)_{10}
\end{array}
$$

Why Important: More concise than binary, but related (a power of 2)

Hexadecimal (or hex) is often used for addressing



Conversion Between Bases

To convert from one base to another:

1) **Convert the Integer Part**

2) **Convert the Fraction Part**

3) **Join the two results with a base point**

Conversion Details

- To Convert the Integral Part:
  - Repeatedly divide the number by the new base and save the remainders.
  - The digits for the new base are the remainders **in *reverse order*** of their computation.
  - If the new base is > 10, then convert all remainders > 10 to digits A, B, …

- To Convert the Fractional Part:
  - Repeatedly multiply the fraction by the new base and save the integer digits that result.
  - The digits for the new base are the integer digits **in *order*** of their computation.
  - If the new base is > 10, then convert all integers > 10 to digits A, B, …

Example: Convert 46.6875_{10} To Base 2

- Convert 46 to Base 2

- Convert 0.6875 to Base 2:

- Join the results together with the base point:

Checking the Conversion

- To convert back, sum the digits times their respective powers of r.
- From the prior conversion of 46.6875_{10}

$$101110_2 = 1 \cdot 32 + 0 \cdot 16 + 1 \cdot 8 + 1 \cdot 4 + 1 \cdot 2 + 0 \cdot 1$$
$$= 32 + 8 + 4 + 2$$
$$= 46$$

$$0.1011_2 = 1/2 + 1/8 + 1/16$$
$$= 0.5000 + 0.1250 + 0.0625$$
$$= 0.6875$$

Hexadecimal (Octal) to Binary and Back

- Hexadecimal (Octal) to Binary:
  - Restate the hexadecimal (octal) as four (three) binary digits starting at the base point and going both ways.
- Binary to Hexadecimal (Octal):
  - Group the binary digits into four (three) bit groups starting at the base point and going both ways, padding with zeros as needed in the fractional part.
  - Convert each group of three bits to an hexadecimal (octal) digit.

Base Conversion - Positive Powers of 2

- Useful for Base Conversion

Exponent	Value
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024

Exponent	Value
11	2,048
12	4,096
13	8,192
14	16,384
15	32,768
16	65,536
17	131,072
18	262,144
19	524,288
20	1,048,576
21	2,097,152

Computer from Digital Perspective

- Information: just sequences of binary (0's and 1's)
  - True = 1, False = 0
- Numbers: converted into binary form when "viewed" by computer
  - e.g., 19 = 10011 (16 (1) + 8 (0) + 4 (0) + 2 (1) + 1 (1)) in binary
- Characters: assigned a specific numerical value (ASCII standard)
  - e.g., 'A' = 65 = 1000001, 'a' = 97 = 1100001
- Text is a sequence of characters:
  - "Hi there" = 72, 105, 32, 116, 104, 101, 114, 101

    = 1001000, 1101001, ...

Terminology: Bit, Byte, Word

- bit = a binary digit e.g., 1 or 0
- byte = 8 bits e.g., 01100100
- word = a group of bits that is **architecture dependent**

(the number of bits that an architecture can process at once)

a 16-bit word = 2 bytes e.g., 1001110111000101

a 32-bit word = 4 bytes e.g., 10011101110001011001110111000101

OBSERVATION: computers have bounds on how much input they can handle at once → limits on the sizes of numbers they can deal with

Terminology: MSB, LSB

- Bit at the left is highest order, or most significant bit (MSB)
- Bit at the right is lowest order, or least significant bit (LSB)

MSB LSB

e.g., 10011101110001 01

- Common reference notation for k-bit value: $b_{k-1}b_{k-2}b_{k-3}\ldots b_1b_0$

Number of Bits Required

- Given M elements to be represented by a binary code, the minimum number of bits, *n*, needed, satisfies the following relationships:

$$2^n \geq M > 2^{(n-1)}$$

$n = \log_2 M$ where x , called the *ceiling function,* is the integer greater than or equal to *x*.

- Example: How many bits are required to represent <u>decimal digits</u> with a binary code?

Number of Elements Represented

- Given n digits in base r, there are r^n distinct elements that can be represented.
- But, you can represent m elements, m < r^n
- Examples:
  - You can represent 4 elements in base $r = 2$ with $n = 2$ digits: (00, 01, 10, 11).

Alphanumeric codes – ASCII character codes

- American Standard Code for Information Interchange
- This code is a popular code used to represent information sent as character-based data.
  It uses 7-bits to represent:
  - 94 Graphic printing characters.
  - 34 Non-printing characters
- Some non-printing characters are used for text format (e.g. BS = Backspace, CR = carriage return)
- Other non-printing characters are used for record marking and flow control (e.g., STX and ETX start and end text areas).

ASCII Properties

ASCII has some interesting properties:

Digits 0 to 9 span Hexadecimal values 30 to 39.

Upper case A - Z span 41 to 5A.

Lower case a - z span 61 to 7A.

- Lower to upper case translation (and vice versa) occurs by flipping bit 6.

Warning: Conversion or Coding?

- Do <u>NOT</u> mix up <u>conversion</u> of a decimal number to a binary number with <u>coding</u> a decimal number with a BINARY CODE.
- $13_{10} = 1101_2$ (This is <u>conversion</u>)
- 13 $\Leftrightarrow$ 0001|0011 (This is <u>coding</u>)

Binary Logic and Gates

- **Binary variables** take on one of two values.
- **Logical operators** operate on binary values and binary variables.
- Basic logical operators are the **logic functions** AND, OR and NOT.
- **Logic gates** implement logic functions.
- **Boolean Algebra**: a useful mathematical system for specifying and transforming logic functions. Is a foundation for designing and analyzing digital systems!

Binary Variables

- Recall that the two binary values have different names:
  - True/False
  - On/Off
  - Yes/No
  - 1/0
- We use 1 and 0 to denote the two values.

Logical Operations

- The three basic logical operations are:
  - AND
  - OR
  - NOT
- AND is denoted by a dot ($\cdot$).
- OR is denoted by a plus (+).
- NOT is denoted by an overbar ($\bar{\ }$), a single quote mark (') after, or (~) before the variable.

Notation Examples

- Examples:
  - $Y = A \cdot B$ is read "Y is equal to A AND B."
  - $z = x + y$ is read "z is equal to x OR y."
  - $X = \overline{A}$ is read "X is equal to NOT A."

- Note: The statement:

  1 + 1 = 2 (read "one plus one equals two")

  is not the same as

  1 + 1 = 1 (read "1 or 1 equals 1").

Operator Definitions

- Operations are defined on the values "0" and "1" for each operator:

AND	OR	NOT
$0 \cdot 0 = 0$	$0 + 0 = 0$	$\overline{0} = 1$
$0 \cdot 1 = 0$	$0 + 1 = 1$	$\overline{1} = 0$
$1 \cdot 0 = 0$	$1 + 0 = 1$	
$1 \cdot 1 = 1$	$1 + 1 = 1$	

Truth Tables

- *Truth table* – a tabular listing of the values of a function for all possible combinations of values on its arguments

- Example: Truth tables for the basic logic operations:

AND		
X	Y	$Z = X \cdot Y$
0	0	0
0	1	0
1	0	0
1	1	1

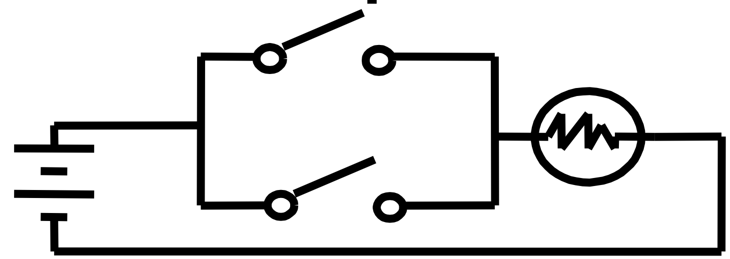
OR		
X	Y	$Z = X + Y$
0	0	0
0	1	1
1	0	1
1	1	1

NOT	
X	$Z = \overline{X}$
0	1
1	0

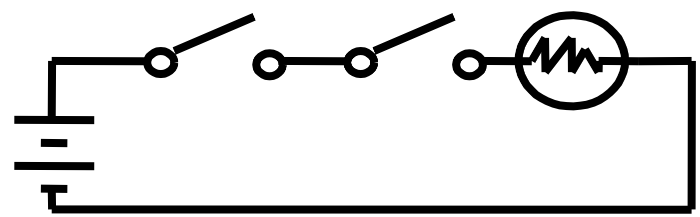
Logic Function Implementation

- Using Switches
  - For inputs:
    - logic 1 is switch closed
    - logic 0 is switch open
  - For outputs:
    - logic 1 is light on
    - logic 0 is light off.
  - NOT uses a switch such that:
    - logic 1 is switch open
    - logic 0 is switch closed

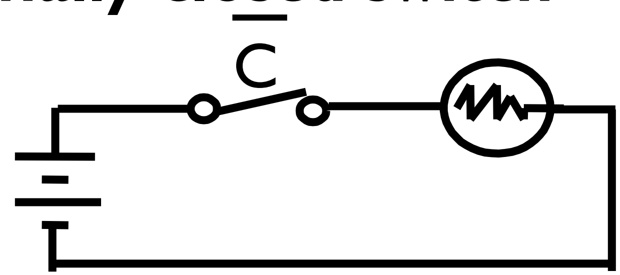
Switches in parallel => OR



Switches in series => AND

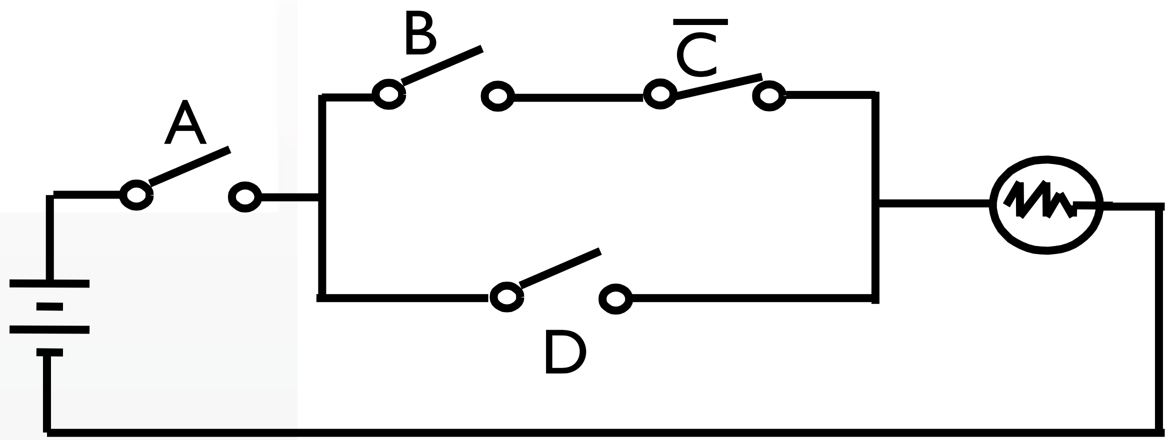


Normally-closed switch => NOT



Logic Function Implementation

- Example: Logic Using Switches



- Light is on (L = 1) for

  L(A, B, C, D) =

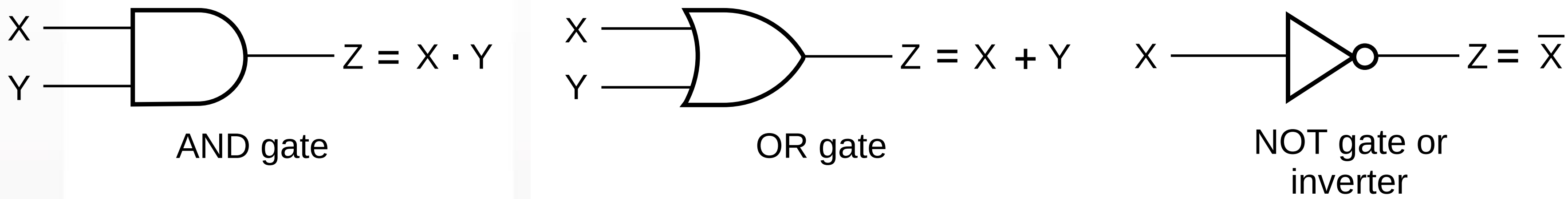
  and off (L = 0), otherwise.
- Useful model for relay circuits and for CMOS gate circuits, the foundation of current digital logic technology

Logic Gates

- In the earliest computers, switches were opened and closed by magnetic fields produced by energizing coils in ***relays***. The switches in turn opened and closed the current paths.
- Later, ***vacuum tubes*** that open and close current paths electronically replaced relays.
- Today, ***transistors*** are used as electronic switches that open and close current paths.

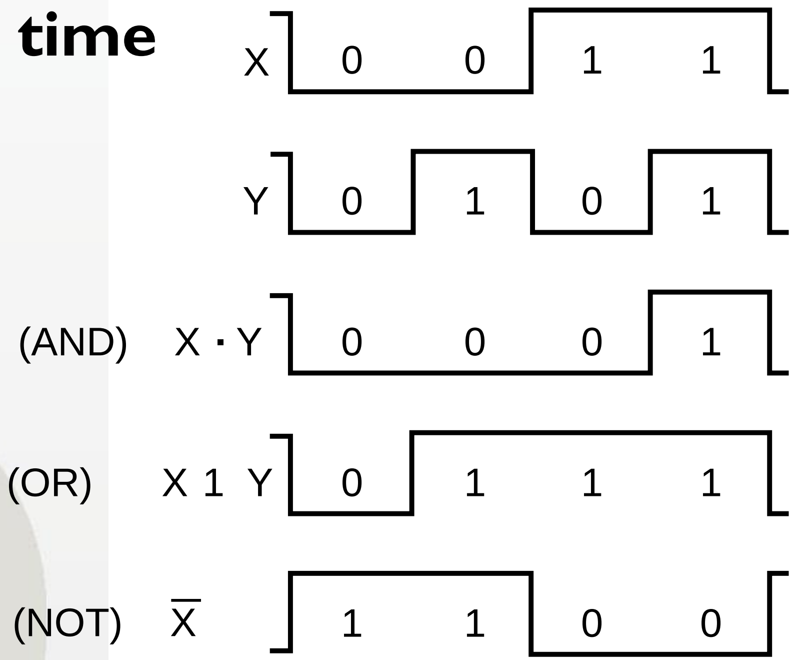
Logic Gate Symbols and Behavior

- **Logic gates have special symbols:**



(a) Graphic symbols

- **And waveform behavior in time as follows:**



(b) Timing diagram

Logic Diagrams and Expressions

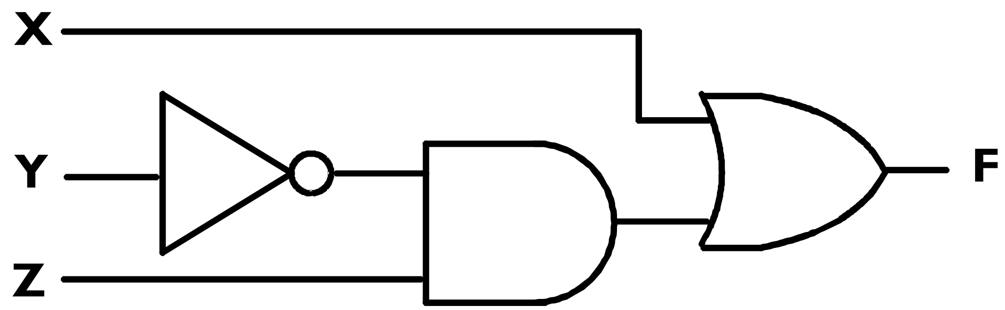
Truth Table

X Y Z	$F = X + \bar{Y} \cdot Z$
0 0 0	0
0 0 1	1
0 1 0	0
0 1 1	0
1 0 0	1
1 0 1	1
1 1 0	1
1 1 1	1

Equation

$$F = X + \bar{Y} Z$$

Logic Diagram



- Boolean equations, truth tables and logic diagrams describe the same function!
- Truth tables are unique; expressions and logic diagrams are not. This gives flexibility in implementing functions.

Boolean Algebra

- An algebraic structure defined on a set of at least two elements, B, together with three binary operators (denoted +, $\cdot$ and $\overline{\ }$) that satisfies the following basic identities:

1.	$X + 0 = X$	2.	$X \cdot 1 = X$	
3.	$X + 1 = 1$	4.	$X \cdot 0 = 0$	
5.	$X + X = X$	6.	$X \cdot X = X$	
7.	$X + \overline{X} = 1$	8.	$X \cdot \overline{X} = 0$	
9.	$\overline{\overline{X}} = X$			
10.	$X + Y = Y + X$	11.	$XY = YX$	**Commutative**
12.	$(X + Y) + Z = X + (Y + Z)$	13.	$(XY) Z = X(Y Z)$	**Associative**
14.	$X(Y + Z) = XY + XZ$	15.	$X + YZ = (X + Y) (X + Z)$	**Distributive**
16.	$\overline{X + Y} = \overline{X} \cdot \overline{Y}$	17.	$\overline{X \cdot Y} = \overline{X} + \overline{Y}$	**DeMorgan's**

Boolean Operator Precedence

- The order of evaluation in a Boolean expression is:

1. Parentheses
2. NOT
3. AND
4. OR

Example: Boolean Algebraic Proof

- $A + A \cdot B = A$ (Absorption Theorem)

Proof Steps	Justification using Identities
$A + A \cdot B$	
$= A \cdot 1 + A \cdot B$	$X = X \cdot 1$
$= A \cdot (1 + B)$	$X \cdot Y + X \cdot Z = X \cdot (Y + Z)$ (Distributive Law)
$= A \cdot 1$	$1 + X = 1$
$= A$	$X \cdot 1 = X$

Expression Simplification

- An application of Boolean algebra
- Simplify to contain the smallest number of **<u>literals</u>** (complemented and uncomplemented variables):

$$AB + \overline{A}CD + \overline{A}BD + \overline{A}C\overline{D} + ABCD$$

$$= AB + ABCD + \overline{A}\,C\,D + \overline{A}\,C\,\overline{D} + \overline{A}\,B\,D$$

$$= AB + AB(CD) + \overline{A}\,C\,(D + \overline{D}) + \overline{A}\,B\,D$$

$$= AB + \overline{A}\,C + \overline{A}\,B\,D = B(A + \overline{A}D) + \overline{A}C$$

$$= B\,(A + D) + \overline{A}\,C \quad \rightarrow \quad 5 \text{ literals}$$

Any Questions?