Computer Arithmetic

Logic Gates

- The electronics inside a modern computer are digital.
- Digital electronics operate with only two voltage levels of interest: a high voltage and a low voltage.
- *A logic gate is a simple switching circuit that determines whether an input pulse can pass through to the output in digital circuits.*
- Logic gates use Boolean algebra to execute logical processes.
- Logic gates are found in nearly every digital gadget we use on a regular basis. Logic gates are used in the architecture of our telephones, laptops, tablets, and memory devices.

- Logic blocks are categorized as one of two types, depending on whether they contain memory.
- Blocks without memory are called combinational; the output of a combinational block depends only on the current input.
- In blocks with memory called the Sequential, the outputs can depend on both the inputs and the value stored in memory, which is called the state of the logic block.
- Combinational logic: A logic system whose blocks do not contain memory and hence compute the same output given the same input.
- Sequential logic: A group of logic elements that contain memory and hence whose value depends on the inputs as well as the current contents of the memory

Boolean Algebra

- *Boolean algebra is a type of logical algebra in which symbols represent logic levels.*
- *The digits(or symbols) 1 and 0 are related to the logic levels in this algebra; in electrical circuits, logic 1 will represent a closed switch, a high voltage, or a device's "on" state. An open switch, low voltage, or "off" state of the device will be represented by logic 0.*
- A light bulb can be used to demonstrate the operation of a logic gate. When logic 0 is supplied to the switch, it is turned off, and the bulb does not light up. The switch is in an ON state when logic 1 is applied, and the bulb would light up. In integrated circuits (IC), logic gates are widely employed.

Truth table

- *The outputs for all conceivable combinations of inputs that may be applied to a logic gate or circuit are listed in a truth table. When we enter values into a truth table, we usually express them as 1 or 0, with 1 denoting True logic and 0 denoting False logic.*

Types of Logic gates

- A logic gate is a digital gate that allows data to be transferred. Logic gates, use logic to determine whether or not to pass a signal. Logic gates, on the other hand, govern the flow of information based on a set of rules. The following types of logic gates are commonly used:
    - AND
    - OR
    - NOT
    - NOR
    - NAND
    - XOR
    - XNOR

Digital Logic Gate Symbols

Gate	Symbol	Notation	Truth Table
AND		$A \cdot B$	INPUT / OUTPUT A | B | A AND B 0 | 0 | 0 0 | 1 | 0 1 | 0 | 0 1 | 1 | 1
OR		$A + B$	INPUT / OUTPUT A | B | A OR B 0 | 0 | 0 0 | 1 | 1 1 | 0 | 1 1 | 1 | 1
NOT		$\overline{A}$	INPUT / OUTPUT A | NOT A 0 | 1 1 | 0

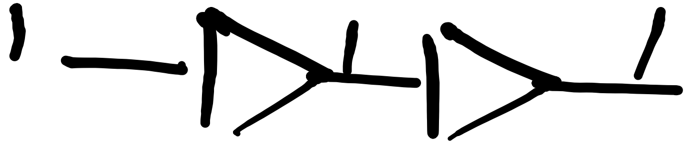
inverter

<table>
<tr>
<td>NAND</td>
<td></td>
<td>$\overline{A \cdot B}$</td>
<td>
<table>
<tr><th colspan="2">INPUT</th><th>OUTPUT</th></tr>
<tr><th>A</th><th>B</th><th>A NAND B</th></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table>
</td>
</tr>
<tr>
<td>NOR</td>
<td></td>
<td>$\overline{A + B}$</td>
<td>
<table>
<tr><th colspan="2">INPUT</th><th>OUTPUT</th></tr>
<tr><th>A</th><th>B</th><th>A NOR B</th></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table>
</td>
</tr>
<tr>
<td>XOR</td>
<td></td>
<td>$A \oplus B$</td>
<td>
<table>
<tr><th colspan="2">INPUT</th><th>OUTPUT</th></tr>
<tr><th>A</th><th>B</th><th>A XOR B</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table>
</td>
</tr>
</table>

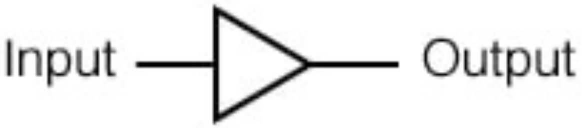
$\bar{A}B + A\bar{B}$

- Buffer gate :

  A buffer, is **a basic logic gate that passes its input, unchanged, to its output**. Its behavior is the opposite of a NOT gate. The main purpose of a buffer is to regenerate the input, usually using a strong high and a strong low



"Buffer" gate



Input	Output
0	0
1	1

Representing Boolean expressions as circuits:

- Minimizing the expression
- Representing using gates

PROPERTIES OF BOOLEAN ALGEBRA:

- Identity law: $A + 0 = A$ and $A \cdot 1 = A$
- Zero and One laws: $A + 1 = 1$ and $A \cdot 0 = 0$
- Inverse laws: $\underline{A + \overline{A} = 1}$ and $A \cdot \overline{A} = 0$
- Commutative laws: $A + B = B + A$ and $A \cdot B = B \cdot A$
- Associative laws: $A + (B + C) = (A + B) + C$ and $A \cdot (B \cdot C) = (A \cdot B) \cdot C$
- Distributive laws: $A \cdot (B + C) = (A \cdot B) + (A \cdot C)$ and
  $A + (B \cdot C) = (A + B) \cdot (A + C)$

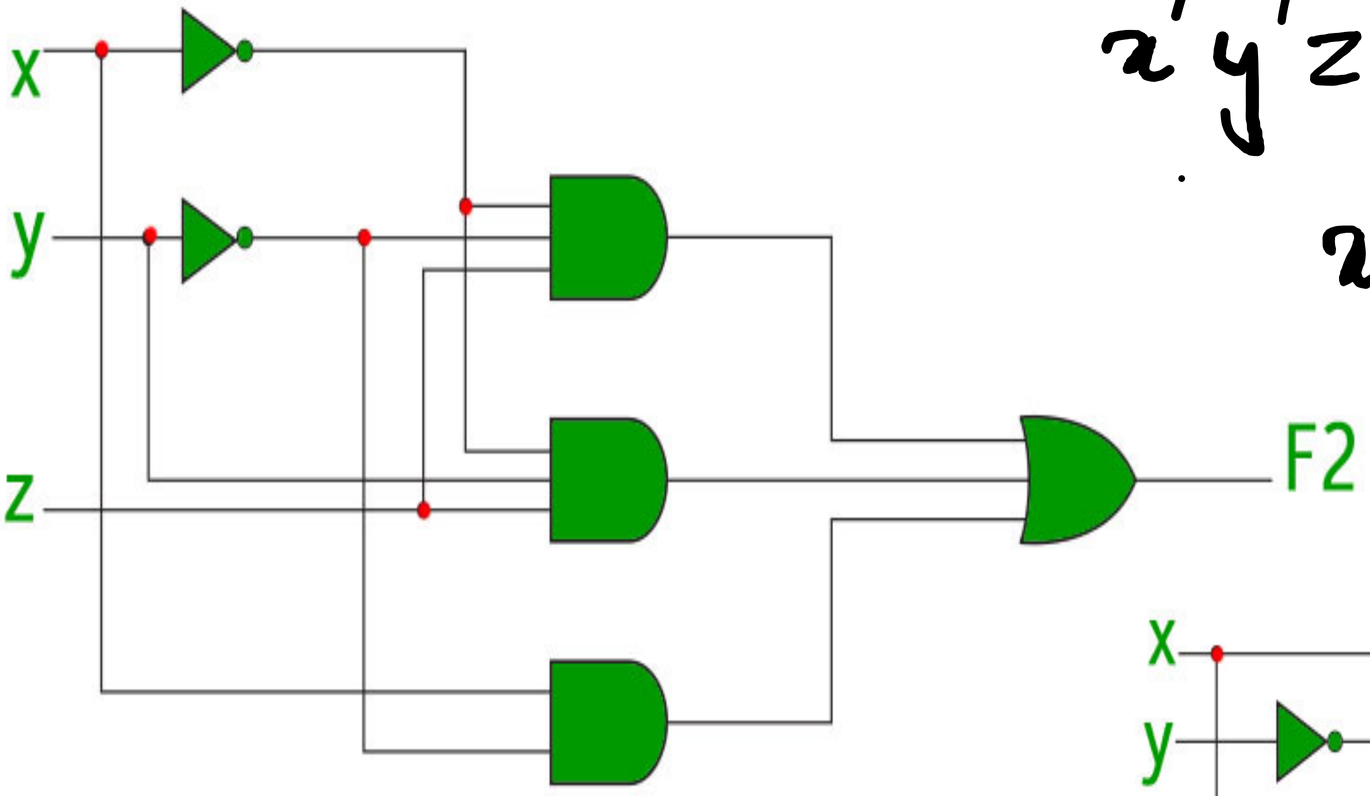
$-\ '$ $\quad \bar{y}, y'$

- The function (a) $F = x'y'z + x'yz + xy'$ can be minimized to (b) $F = x'z + xy'$

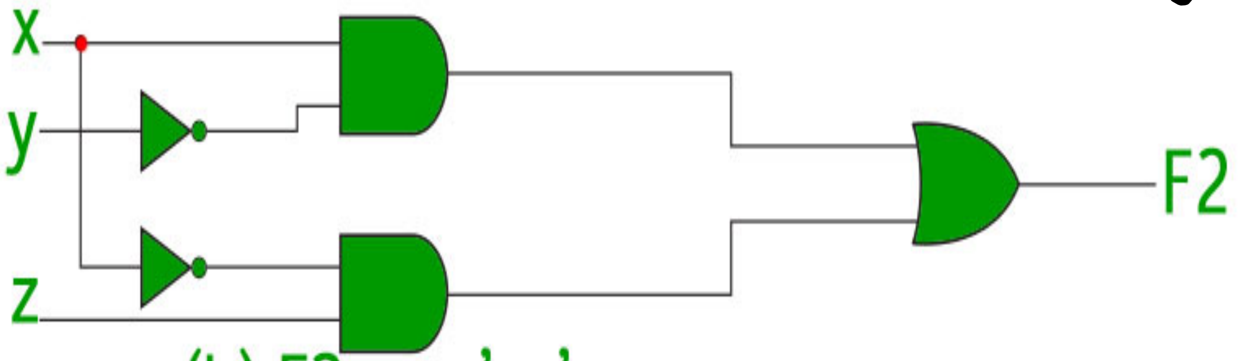
$$x'y'z + x'yz + xy'$$

$$x'z(y'+y) + xy'$$

$$x'z(1) + xy'$$



(a) F2 = x’y’z+x’yz+xy’

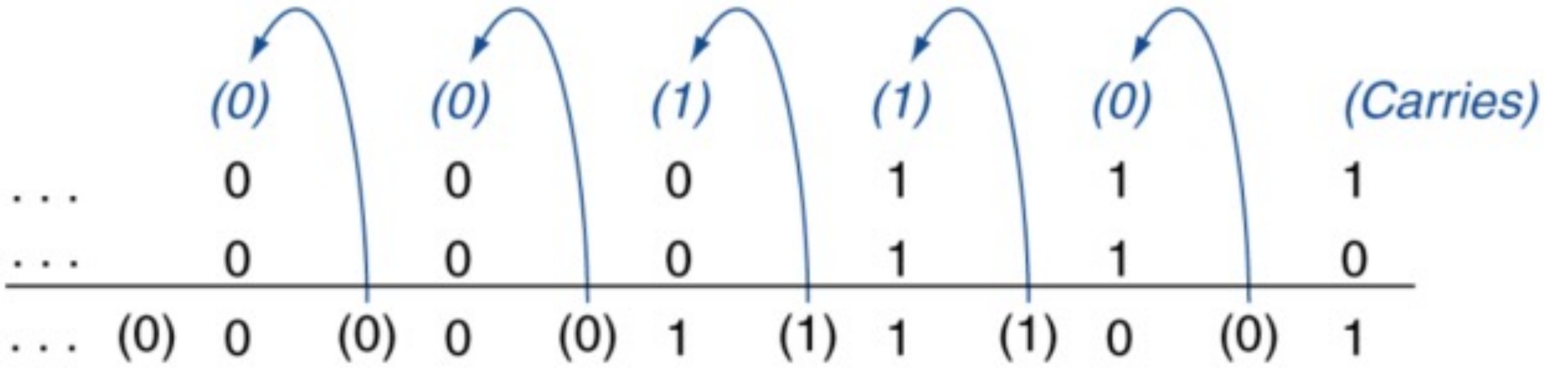


(b) F2 = xy’+x’z

Arithmetic for Computers

- Operation on integers
  - Addition and Subtraction
  - Multiplication and Division
  - Dealing with overflow
- Floating point real numbers
  - Representation and operations

- Example = 7+6



- Overflow if result out of range

    - Adding +ve and –ve operands, no overflow
    - Adding two +ve operands
        - overflow if sign bit is 1 (answer comes negative)
    - Adding two –ve operands
        - overflow if sign bit is 0 (answer comes positive)

- **Overflow:**

- In computer architecture 2’s Complement Number System is widely used.

- N-bit 2’s Complement number System can represent Number from -2^{n-1} to $2^{n-1}-1$

4 Bit can represent numbers from **(-8 to 7)**
5 Bit can represent numbers from **(-16 to 15)** in 2’s Complimentary System.

- Overflow Occurs with respect to addition when 2 N-bit 2’s Complement Numbers are added and the answer is too large to fit into that N-bit Group.
- A computer has N-Bit Fixed registers. Addition of two N-Bit Number will result in a max N+1 Bit number. That Extra Bit is stored in carry Flag. But Carry does not always indicate overflow.

- Adding 7 + 1 in 4-Bit must be equal to 8. But 8 cannot be represented with 4 bit 2’s complement number as it is out of range. Two Positive numbers were added and the answer we got is negative (-8). Here Carry is also 0. It is normally left to the programmer to detect overflow and deal with this situation.

Adding 7 + 1 In 2’s Compliment

	0	0	0	1	(1)
+	0	1	1	1	(7)
	1	0	0	0	(-8)

- **Overflow Detection –**
  Overflow occurs when:

- Two negative numbers are added and an answer comes positive or

- Two positive numbers are added and an answer comes as negative.