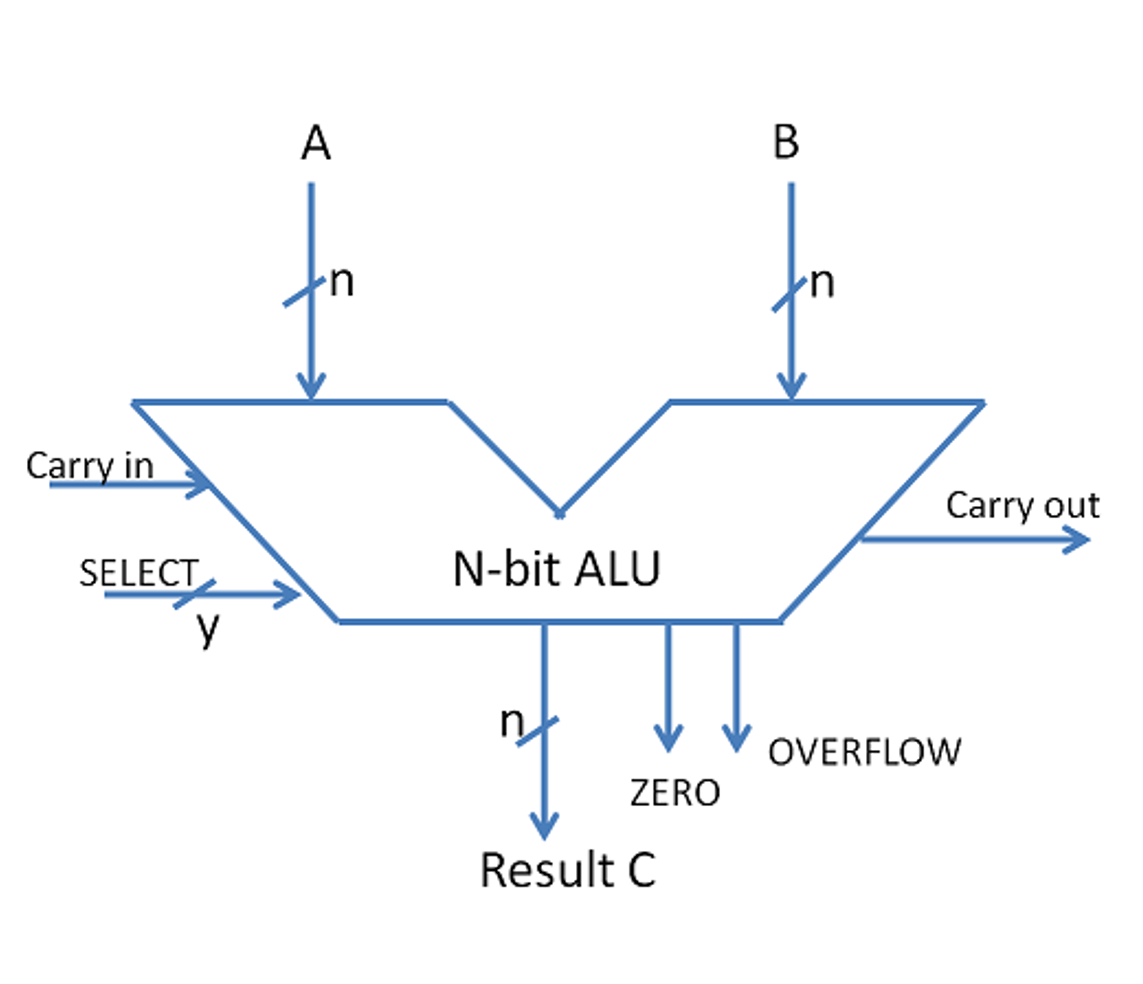
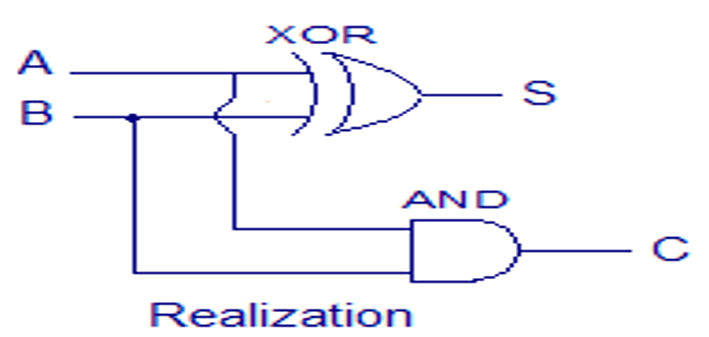
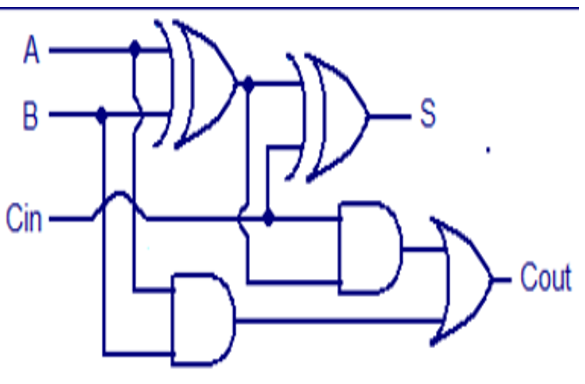


FUNCTIONAL UNITS OF COMPUTER

**KEY**

ISA - Instruction Set Architecture

* **Explain the various functional units of a computer.**
* **Input Unit**
  + The Input Unit is responsible for accepting data and instructions from the outside world and converting them into a form that the computer can understand.
  + - Common input devices include keyboards, mice, scanners, and sensors.
* **Output Unit**
  + - The Output Unit is responsible for presenting the results of computations to the user or transferring data to the outside world.
  + - Common output devices include monitors, printers, speakers, and other display or communication devices.
* **Central processing Unit (ALU and Control Units)**
  + - The CPU is often considered the "brain" of the computer and is responsible for executing instructions. It consists of two main components:
  + The **ALU :** 
    - * Constructed with combinational circuits
      * Digital circuit toperform arithmetic and logical operations on data.
      * Arithmetic operations include addition, subtraction, multiplication, and division.
      * Logical operations involve comparisons and decision-making.(OR AND NOR)
      * Data Movement ops LOAD STORE
      * **Complex ALUs(Co-processor)** are designed for executing floating pt, decimal ops, complex numerical ops
      * Design specifications are derived from ISA, ALU must have the capabilities to execute instructions from ISA.
      * Modern CPUs have multiple ALUs to improve efficiency
      * Size of input qtys of ALU are called word length of a comp.
      * Configs :
        + ISA
        + Acumulator
        + Stack
        + Register-Register architecture
        + Register-Stack architecture
        + Register-memory architecture
      * **Half Adder:**-
        + A basic digital circuit that adds two single-bit binary numbers.
        + It has two outputs, sum (S) and carry (C).
      * **Full Adder**:-
        + An extension of the half adder, capable of adding two binary numbers along with a carry input.
        + It has two outputs: sum (S) and carry (C-out).
      * **Parallel Adder**:-
        + Composed of a cascade of full adders.
        + The number of full adders used depends on the number of bits in the binary numbers being added.
        + Enables simultaneous addition of multiple bits.
      * **Ripple Carry Adder:**-
        + Used for adding two n-bit binary numbers.
        + Used when input sequence is large
        + It processes each bit one at a time, with the carry-out of each full adder serving as the carry-in for the next, creating a ripple effect.
        + Typical Ripple Carry Addition is a Serial Process:

Addition starts by adding LSBs of the augend and addend.

Then next position bits of augend and addend are added along with the carry (if any) from the preceding bit.

This process is repeated until the addition of MSBs is completed.

Speed of a ripple adder is limited due to carry propagation or carry ripple.

Sum of MSB depends on the carry generated by LSB.

* + - * + Drawback :

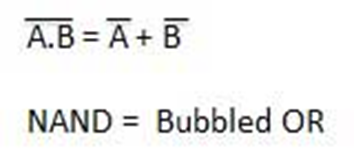
Bit execution in serial manner

So more delay due to carry propagation

To reduce this delay :

* + - * **Carry Look-Ahead Adder(Fast Adder):**-
        + An improved version of the ripple carry adder.
        + It generates the carry-in of each full adder simultaneously, eliminating the ripple effect and reducing the overall delay in the addition process.
        + Each full adder is going to generate the carry simultaneously.
        + It can be contrasted with the simpler, but usually slower, ripple carry adder for which the carry bit is calculated alongside the sum bit, and each bit must wait until the previous carry has been calculated to begin calculating its own result and carry bits (see adder for detail on ripple carry adders)
        + improves speed by reducing the amount of time required to determine carry bits.
        + The carry-look ahead adder calculates one or more carry bits before the sum, which reduces the wait time to calculate the result of the larger value bits.

**De Morgan’s Theorem**

The theorem explains that the complement of the product of all the terms is equal to the sum of the complement of each term. Likewise, the complement of the sum of all the terms is equal to the product of the complement of each term.

* + - * Applications:
        + Hardware design at a cheaper cost
        + Verification of SAS code
        + Computer & electrical engg implementation
        + Employed in Java
* **Control Unit**
  + manages and coordinates the activities of the other units within the computer.
  + It fetches instructions from memory, decodes them, and controls the flow of data between the CPU and other parts of the computer.
* **Memory**
  + Memory is used to store data and instructions that the CPU can quickly access. There are two main types of memory in a computer:
    - **RAM** is volatile memory used for temporary storage of data and program instructions during the operation of the computer. It allows quick read and write access.
    - **ROM** is non-volatile memory that typically contains the computer's firmware or BIOS. It retains data even when the power is turned off.
* **Bus Structure**
  + - The bus structure refers to the communication system that allows data and instructions to be transferred between different components of the computer.
  + It consists of several buses, each serving a specific purpose:
    - Data Bus: Carries data between the CPU, memory, and other peripherals.
    - Address Bus: Specifies the memory location for data transfer.
    - Control Bus: Manages the control signals for communication and coordination between different units.
    - Expansion Bus: Allows connection to peripheral devices.