**COMBINATIONAL LOGIC CIRCUITS**

**Half Adder**

* Used for adding **two 1-bit** numbers.
* Provides two outputs, sum & carry.
* Sum = LSB, Carry = MSB
* Uses XOR and AND.
* Full adder and multiple adder are made using half adder only.

**Mechanism of Half Adder**

* **XOR** operation between the terms to be added decides the **sum** (LSB).
* And **AND** operation between both decides the **carry** (MSB).
* Input variables are called **augend** and **addend**.
* In this case, two K-Maps are required.

**Half Adder Merits & Demerits**

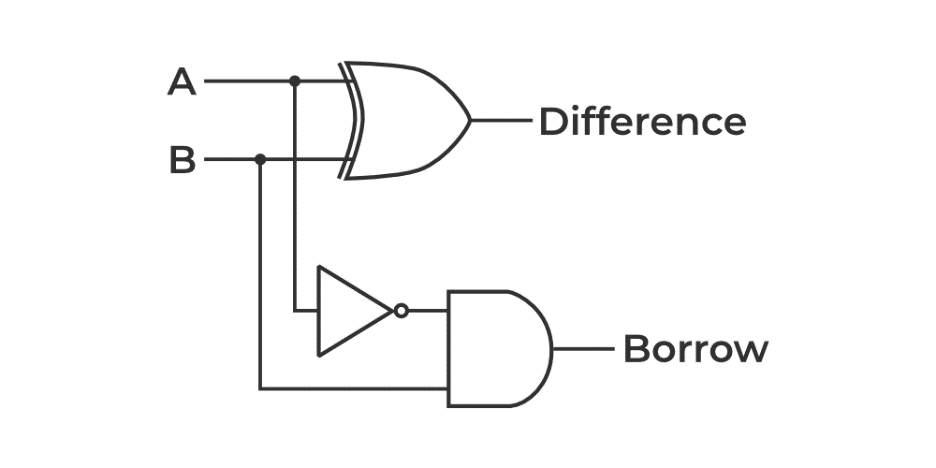
1. **Advantages:-**
   1. Simplicity (easy and not complex)
   2. Speed (circuit flow)
2. **Disadvantages:-**
   1. Limited usefulness (only for two 1-bit numbers)
   2. Lack of convey info (underivable)
   3. Propagation deferral (slow in showing output, when input changed)

**Application of Half Adder**

* Arithmetic circuits (double addition)
* Data handling
* **Address unravelling:** Produce memory location.
* Encoder and decoder circuits.
* Multiplexers & demultiplexers.
* **Counter:** A clock circuit that **“counts”**.

**Half Subtractor**

* Subtraction of two 1-bit number.
* 2 inputs
* 2 output – **borrow, difference**
* **XOR** and **NOT** are used.
* Borrow value is **A’.B**
* A = **minuend** bit, B = **subtrahend** bit



* A XOR B = A’B + B’A

**Half Subtractor Merits & Demerits**

1. **Advantages:-**
   1. Simplicity (easy design)
   2. Building blocks
   3. Low cost (thus low power consumption)
   4. Easy integration (derivable)
2. **Disadvantages:-**
   1. Limited functionality
   2. Inefficient for multi-bits
   3. High propagation delay

**Applications of Half Subtractor**

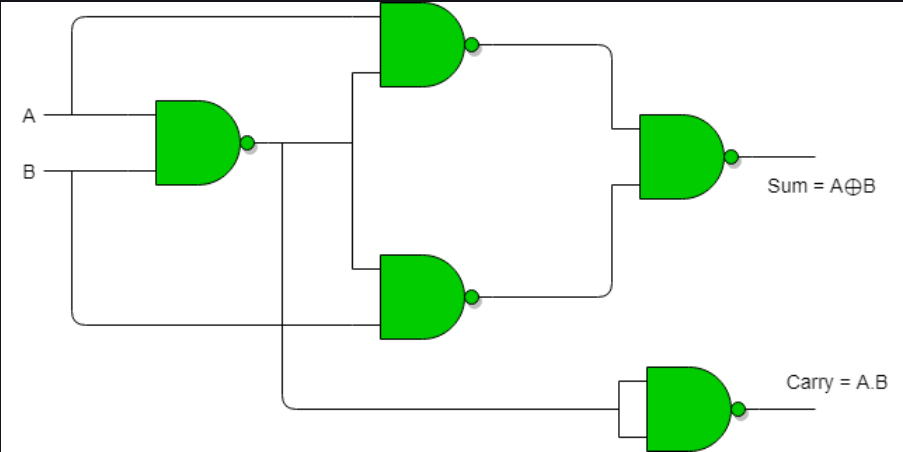
* Calculators (like number cruncher)
* Security alarms (detects when encounters unique circuit flow)
* Automotive vehicles
* Computer systems

**Half Adder & Subtractor Using NAND & NOR**

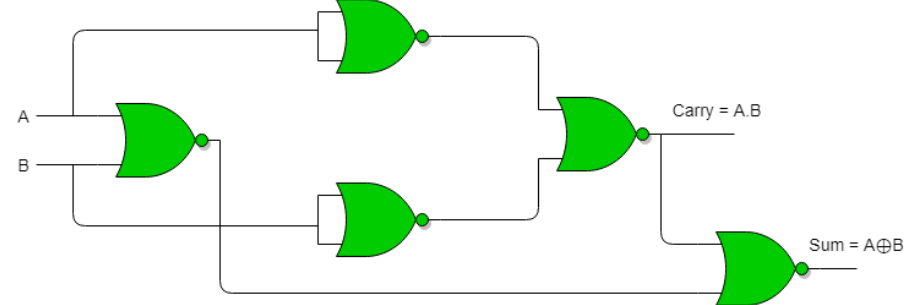
1. **Advantages:-**
   1. Universality (can perform any logic)
   2. Cost-effective (cheaper than other gates)
   3. Low power consumption
2. **Disadvantages:-**
   1. Propagation delay
   2. Noise susceptibility (might affect circuit)

**Diagrams**

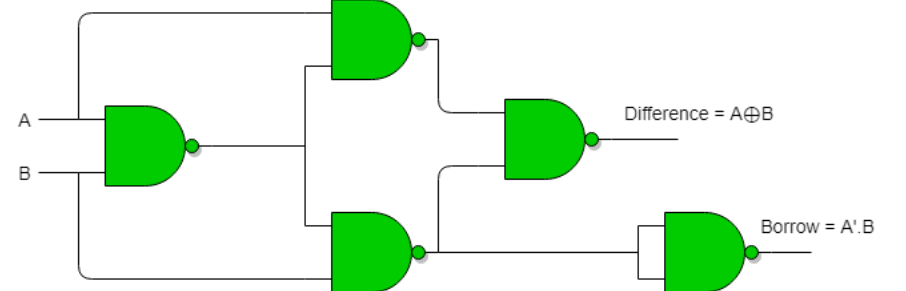
* Half adder using NAND



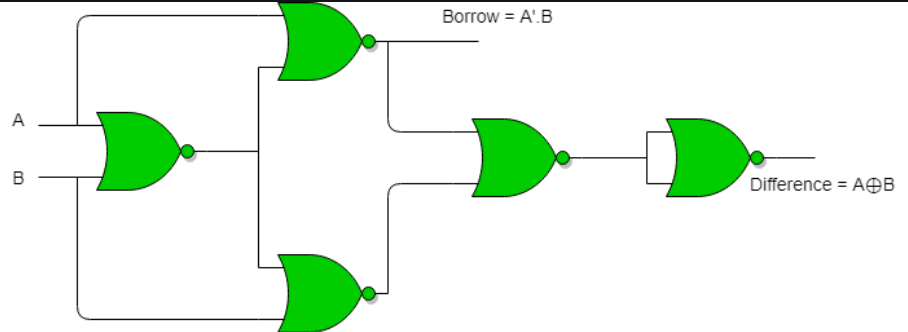
* Half adder using NOR



* Half subtractor using NAND



* Half subtractor using NOR



**Full Adder**

* Used for 1-bit addition.
* 3 inputs – A, B, **C-IN**. 2 outputs – **C-OUT**, S
* C means **carry** and **S is sum**.
* **C-OUT:**
  + Majority 1’s detector
  + Goes **high** when multiple inputs are **high**.
* Can take **upto 8** inputs.
* Thus **byte-wider** (8 bits = 1 byte)

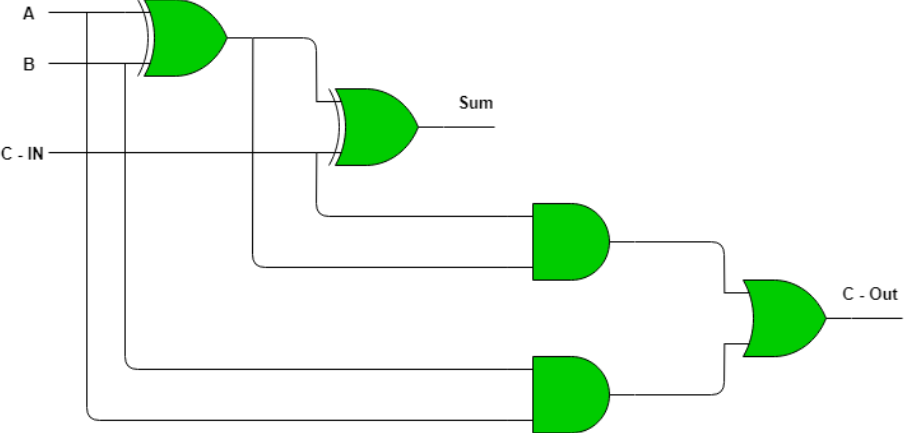
**Cascade:** One after another electrical connection.

* Full adder cascades the bit forward during calculation.

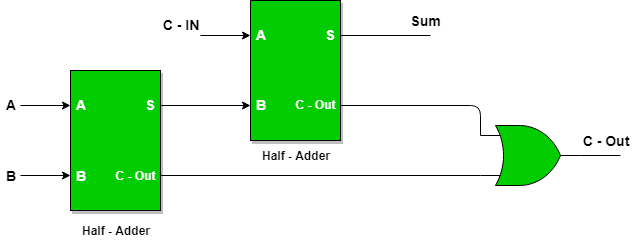
**Expressions**

* **SUM**

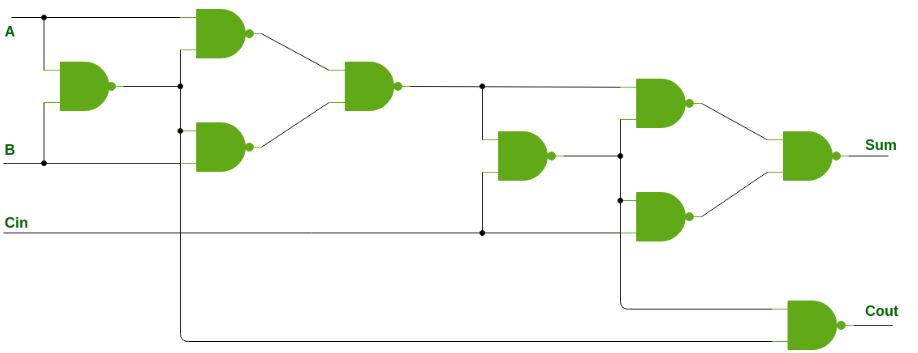
**C-IN XOR (A XOR B)**



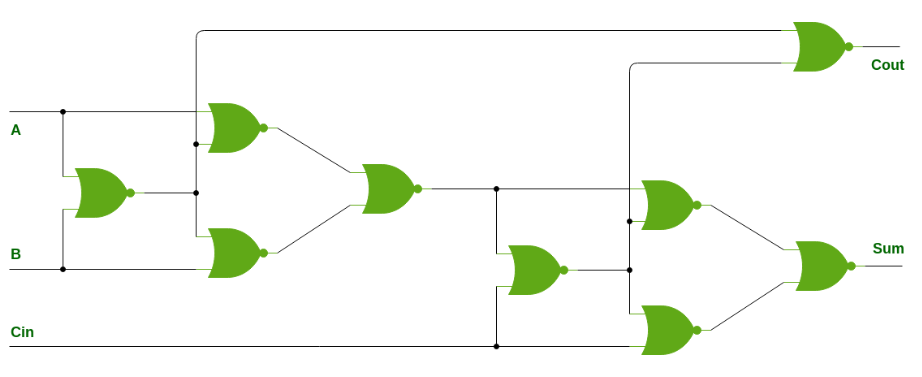
**Full Adder Using Half Adder**



**Full Adder Using NAND**



**Full Adder Using NOR**



**Merits & Demerits of Full Adder**

1. **Advantages:-**
   1. Flexibility (can add three 1-bits)
   2. Carry info (chaining)
   3. Speed
2. **Disadvantages:-**
   1. Complexity
   2. Propagation deferral

**Applications of Full Adder**

* Arithmetic circuits
* Data handling (signal/encryption/correction)
* Counters
* Multiplexers & demultiplexers
* Memory addressing circuits
* ALUs

**Full Subtractor**

* Names are same – minuend, subtrahend
* 3 inputs – A, B, **Bin** and 2 outputs – D, **Bout**
* **D** is **difference** whereas **B** means **borrow**.

**Calculation**

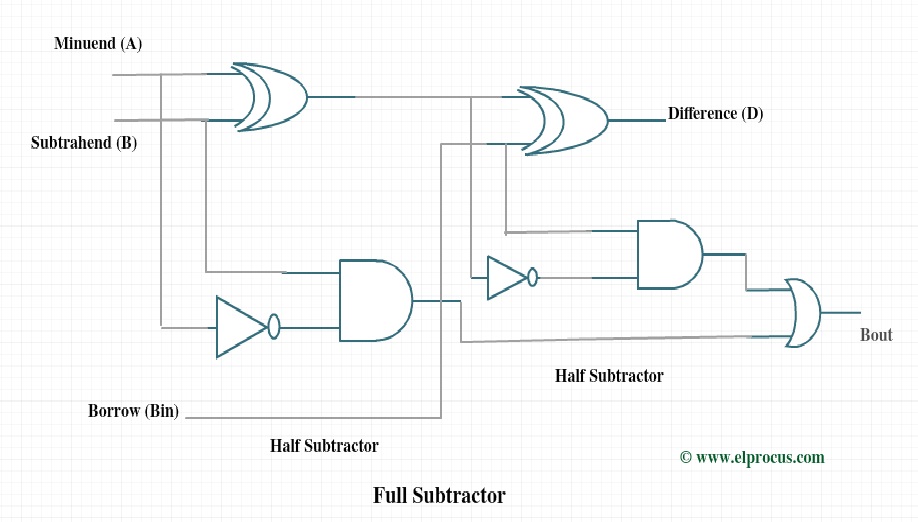
* **Step 1:** Convert the numbers to **2’s complement** form, in case of **negative** number.
* **Step 2 to n:** Do simple subtraction of the numbers step by step.

**Logic Diagram of Full Subtractor**

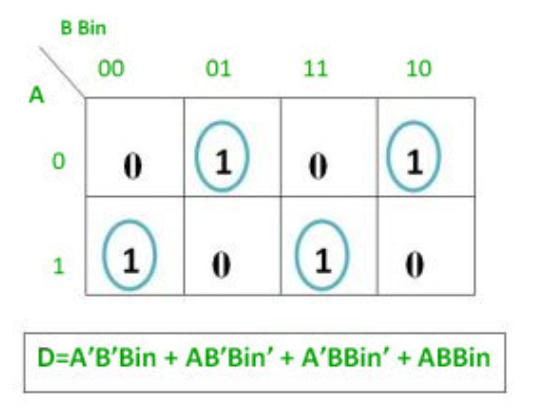
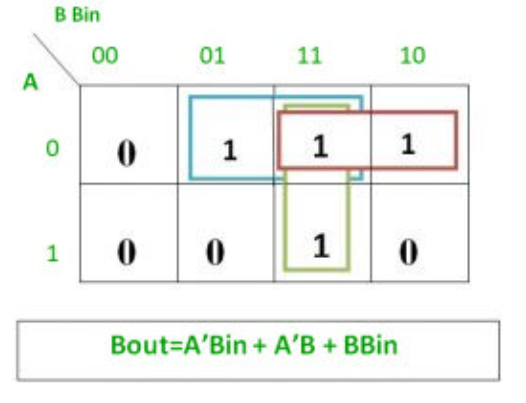
* Consists of **2 half subtractor** and an **additional OR** gate.
* The OR gate **calculates the Bout** bit.

**D = (A XOR B) XOR Bin**

**Bout = Bin (A XOR B)’ + A’B**



**K-Map for Full Subtractor**

**Code Converters**

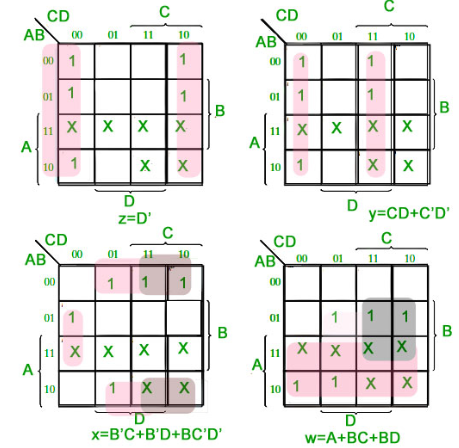
* **Excess-3:** Unweighted, self-complementary code.
* **Self-complimentary:** 1’s complementary of code is excess-3 for 9’s complement of other code.
* For excess-3 code for a given number’s complement, we just need to find 1’s complement of the given number’s excess-3 code.
* It is also excess-3 code for the 9’s complement of 3.

**Decimal to Excess-3 Conversion**

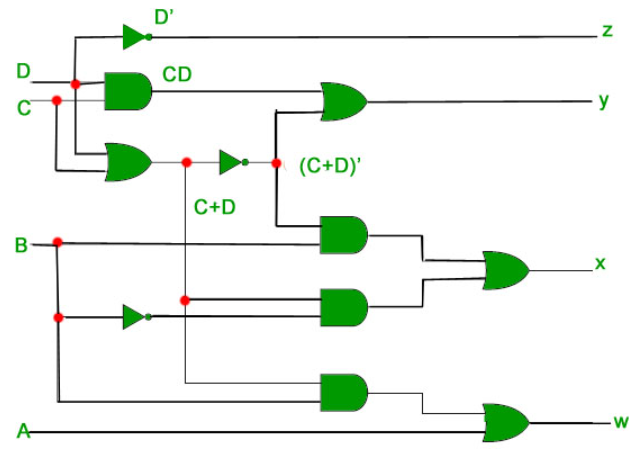
* **Step 1:** Add 3 to each digit of the decimal number.
* **Step 2:** Now convert the number into 4-bit binary.
* We mark numbers after 9 as **X** (as decimal system is 0 to 9).

**BCD(8421) to Excess-3**

* Just add binary 3 (11) to the BCD number.
* **Drawing K-Map Using Truth Table:**



* Corresponding digital circuit:

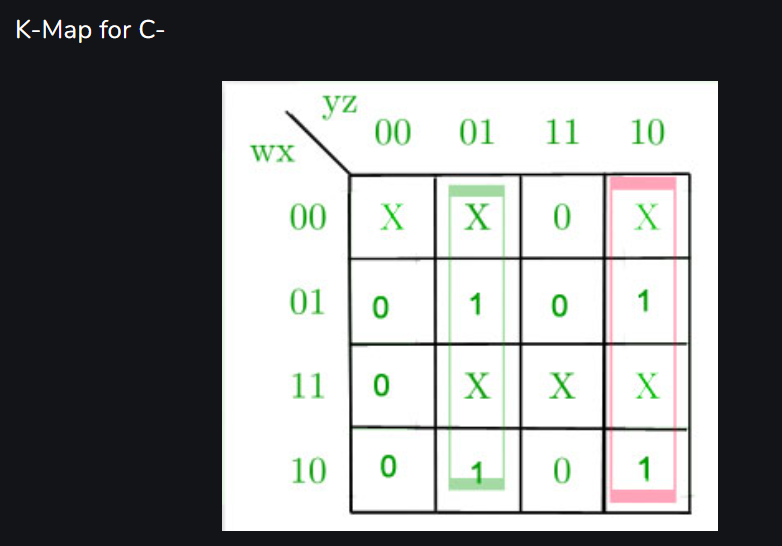


**Excess-3 to Decimal Conversion**

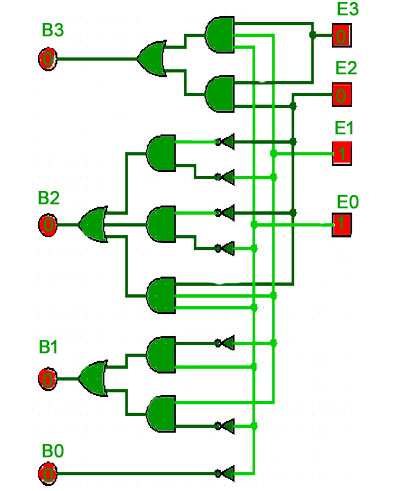
* Exactly **reverse** of decimal to excess-3.

**Excess-3 to BCD(8421)**

* Exactly **reverse** of BCD(8421) to Excess-3.
* Negative numbers are marked as **XXXX**.
* K-Maps are created for separate letters:



* After creating any equation, **minimizing** it is a very good practice.
* **Logic circuit for the same:**

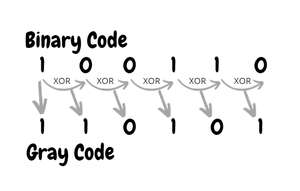


* E3 to E1 = w to z
* B3 to B1 = A to D

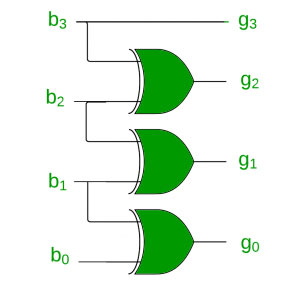
**Gray Code**

* Successive pair of numbers differ by **1-bit**.
* Thus, used for correcting errors.
* **Alternative names for Gray code:-**
  + Unity Hamming Distance Code
  + Cyclic Code
  + Reflecting Code

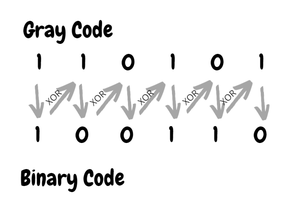
**Binary to Gray Conversion**



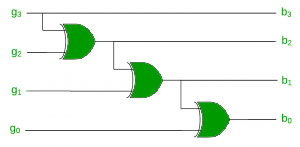
* **Logic circuit:-**



**Binary to Gray**



* **Logic circuit:-**



**BCD to 7 Segment Decoder**

* **BCD:** Encoding decimal numbers as **4-bit** binary numbers.
* **7 segment display:** 7 LEDs connected in a manner to represent all **hexadecimal** numbers (**decimal** **only** in our case).

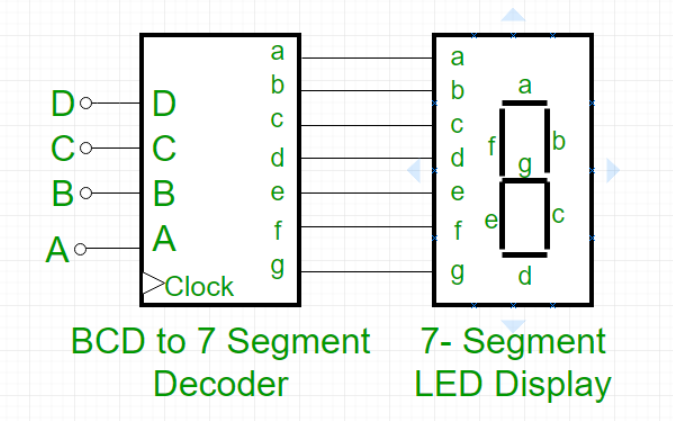
**7 Segment Display Types**

* **Common cathode type:**
  + Cathodes of all 7 LEDs are connected together to **ground** (or **-Vcc**).
  + LED displays digits when **HIGH** signal is supplied to **anodes**.
* **Common anode type:**
  + Anodes of all 7 LEDs are connected together to **battery** (or **+Vcc**).
  + LED displays digits when **LOW** signal is supplied to **cathodes**.

**Process of Working of 7 Segments**

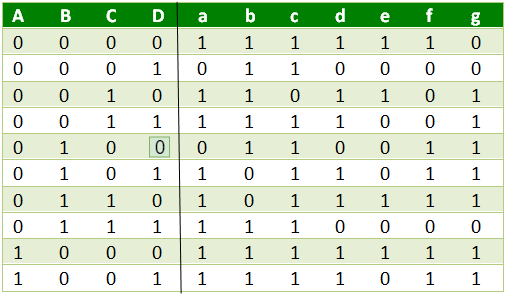
* First, the decimal number is converted to BCD signal.
* Then the BCD codes communicate with the 7 segment LEDs.

**Structure of 7 Segments**

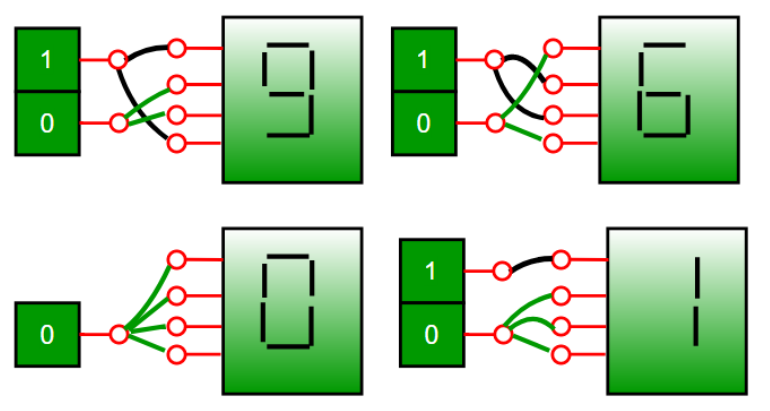


**BCD to 7 Segment Decoder**

* **For common anode type:-**

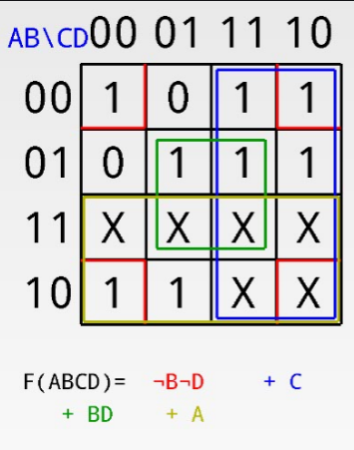


* **For common anode type:-**
  + Just reverse the result outputs in cathode type.
  + Whatever we do, making K-Map and minimizing the equations is a good practice.



**K-Map for 7 Segment**

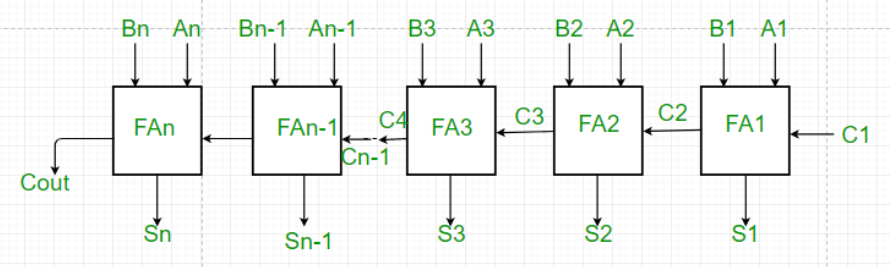
* We make **10 combinations** in its truth table.
* For rest combinations shown in K-Map by default must be marked as **don’t care** conditions.
* **For segment #a:-**



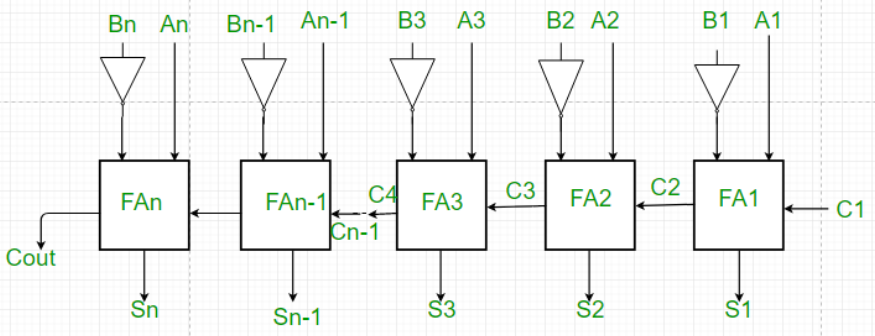
**Parallel Adder**

* Used to calculate addition of numbers **more than 1-bit** in length.
* Consists of full adders connected in chain.

**Number of bits = Number of full adders**



**Parallel Subtractor**



**Merits & Demerits of Parallel Adder/Subtractor**

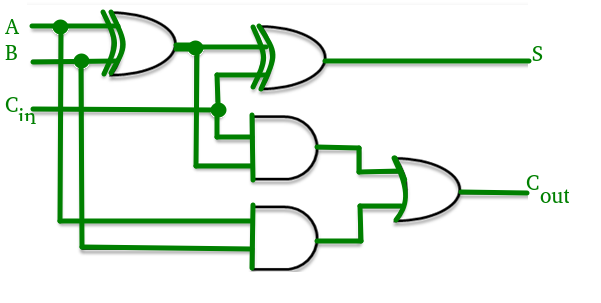
1. **Advantages:-**
   1. Faster than serial adder/subtractor.
   2. Time for calculation doesn’t depend on number of bits.
   3. Less costly.
2. **Disadvantages:-**
   1. Each adder has to wait for previous carry.
   2. Propagation delay.

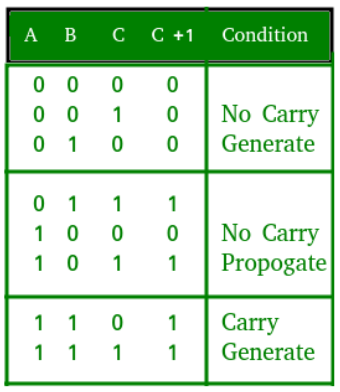
**Carry Look-Ahead Adder**

* There are many ways to reduce propagation deferral.
* One of the way is carry look-ahead.
* **Carry look-ahead:** Calculates carry signals in advance.
* Full carry adder is also known as **ripple carry adder** (just write **Cn** instead **Cout**).
* A block waiting for previous block to pass signals is **one type of** propagation delay.

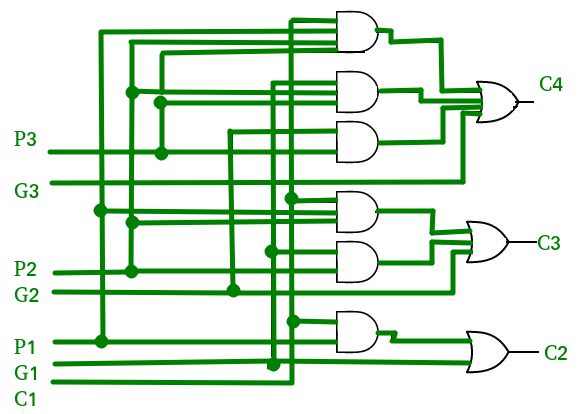
**Propagation time = Propagation delay for other blocks**

* It reduced propagation deferral by sending signals to group of **two bits at a time**, rather than one.





**Carry Generator Figures**



**Time Complexity Analysis (Carry Look-Ahead Adder)**

* **It is made up of two parts:**
  + Part that calculates carry for each bit.
  + Part that adds two bits and carry.
* **Time complexity:** log(n)
  + It generates from the bit generating part.
  + For **k** inputs, it is **logk(n+1)**.

**Merits & Demerits Of Carry Look-Ahead Adder**

1. **Advantages:-**
   1. Propagation delay reduced.
   2. Fastest addition logic
2. **Disadvantages:-**
   1. Gets complicated as the number of variables increases.
   2. Costlier circuit arrangement.

**Note About Carry Generator**

* For addition of **n-bit** variables it requires:
  + **[n(n+1)]/2** AND gates.
  + **n** OR gates.

**Magnitude Comparator**

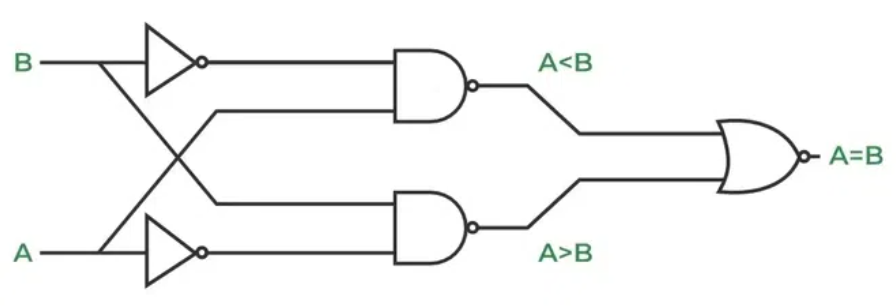
* Compares two binary numbers.
* **3 output terminals:** A>B, A=B, A<B
* The circuit checks starting from MSB to LSB, and **terminates** when **A>B or A<B** is encountered.

**Ways to Implement Magnitude Comparator**

* By using XOR, AND, OR gates.
* Arrangement of full adders.
* **Formula obtained:**

**(A<B)+(A>B) = A’B+AB’**

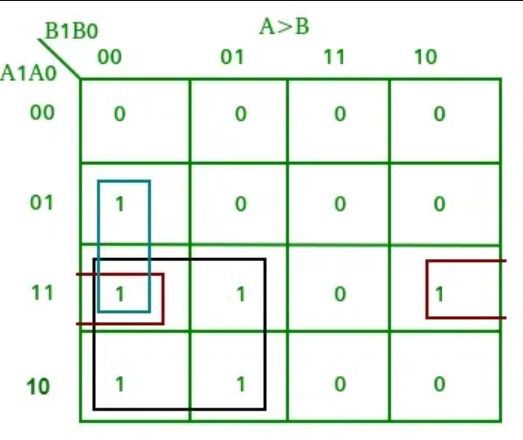
**=> ((A<B)+(A>B))’ = (A=B)**



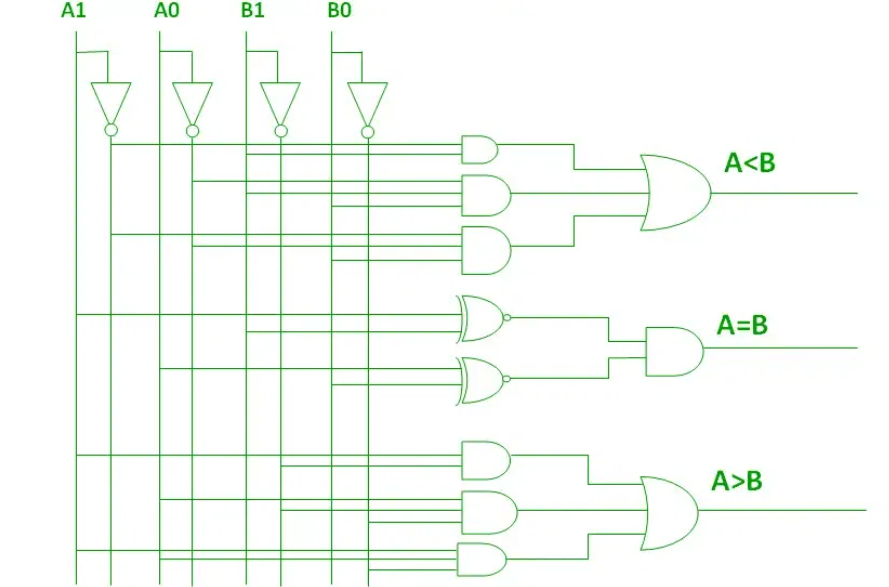
**Magnitude Comparator as per Bits**

* **1-bit each:** Inputs are A and B.
* **2-bit each:** Inputs are A0, A1, B0 and B1.
* Number of output terminals are **3 always**.

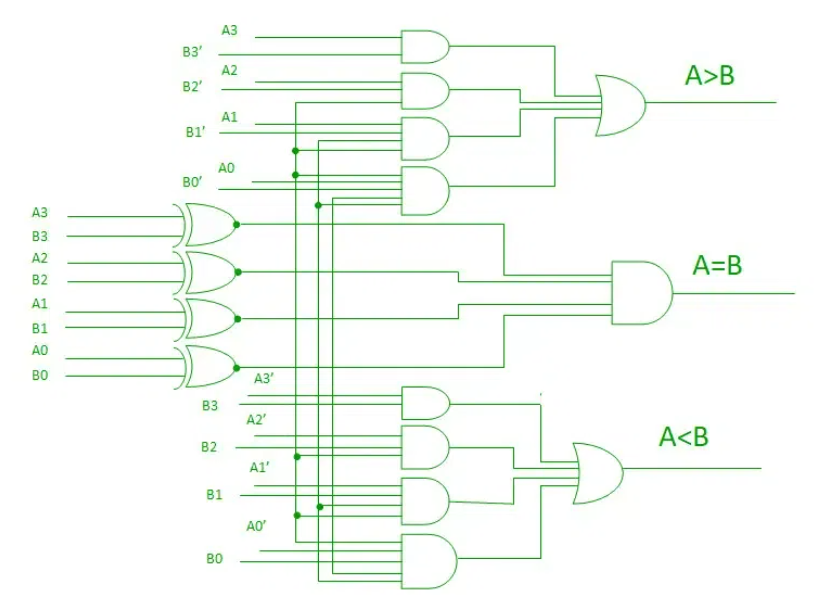
**K-Map for A>B (2-bits)**



* For any given truth table for any topic, there might be **multiple outputs columns**.
* For each output column, always **separate K-map** is created.



**Circuit for 8-bit Comparator**



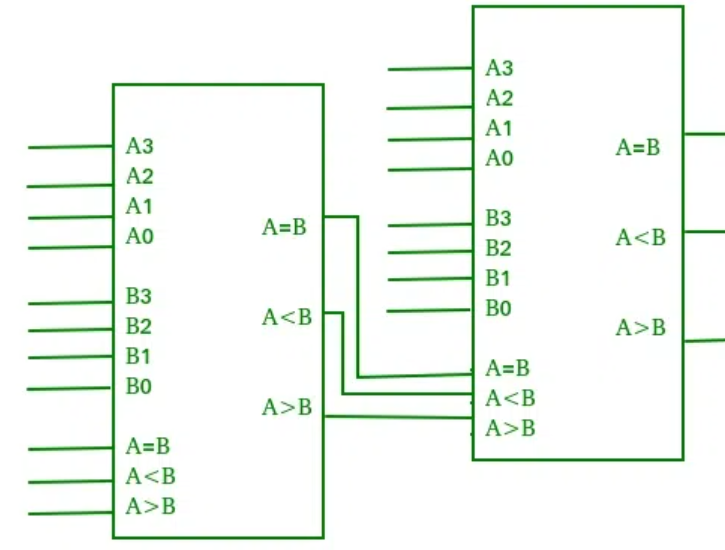
**Number of total combinations = 22n**

**Number of equal combinations = 2n**

**Number of unequal combinations = (Subtraction of above two)**

**Number of A<B or A>B = Enequals/2**

**Cascading Comparator**



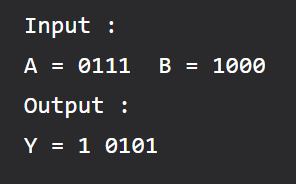
* Used for comparing digital numbers with length **greater than 4-bits**.
* The body on left is called **lower-order** comparator.
* And the one right is called **higher-order** comparator.

**Applications of Comparators**

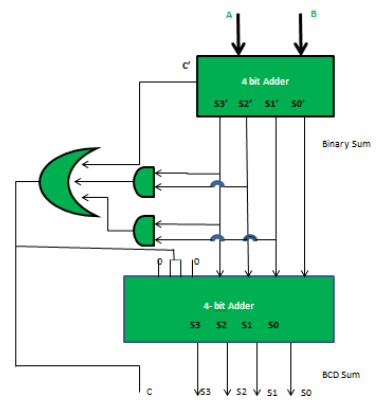
* Are used in CPUs and MCUs (microcontrollers).
* Control and detection applications.
* Servo motor controls.
* Password verification & biometrics system.

**BCD Adder**

* Adds BCD numbers.

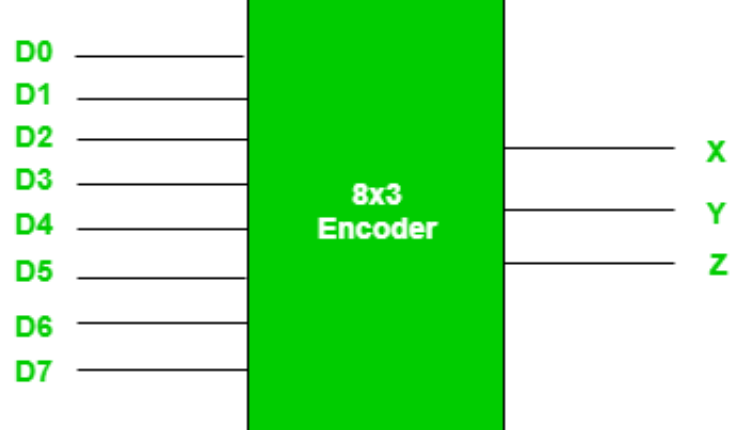


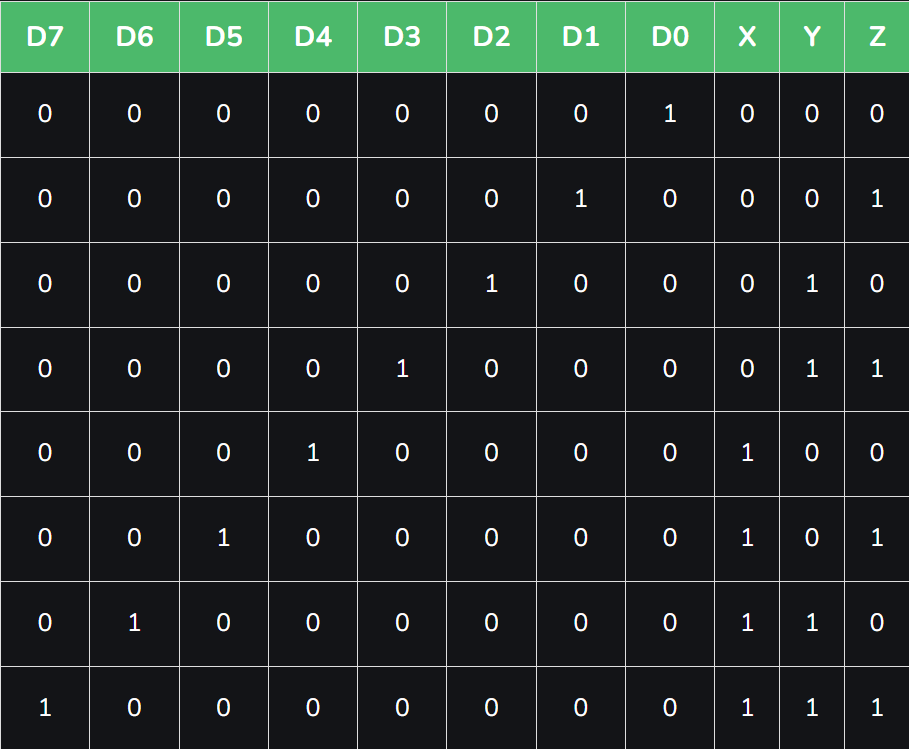
**C' + S3'.S2' + S3'.S1' = 1**



**Encoders And Decoders**

* **Encoder:-**
  + Used for converting one form of information to another.
  + Converts **2n** lines of input into a code of **n** output.
* **Decoder:-**
  + Same use in reverse.
  + Converts **n** lines of input into a code of **2n** outputs.
* **Encoder example:-**





**Boolean Function for Outputs**

**X = D4 + D5 + D6 + D7**

**Y = D2 +D3 + D6 + D7**

**Z = D1 + D3 + D5 + D7**

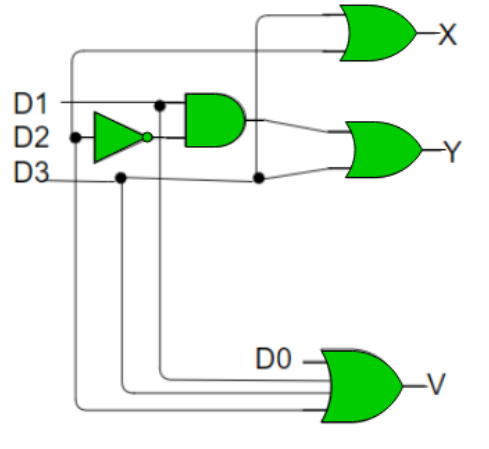
**Limitations to Encoders**

* Limitation is that only one output can be active for a single **high** input.
* Else for multiple high inputs, the output is undefined.
* To avoid similar outputs when multiple inputs are high, we use **priority encoders**.
* Also, all inputs can’t be 0, where we get output same as D0.
* For avoiding it, an extra digit of bit is added, called **valid “V” bit** i.e., 0.



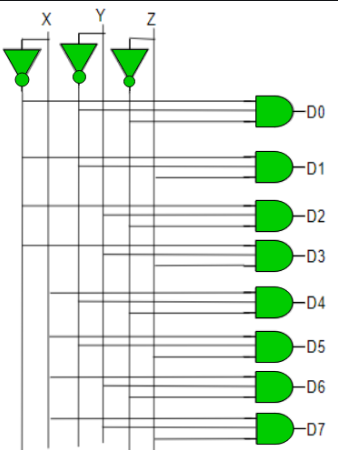
* Don’t care conditions are used for main outputs in such cases, and **V-bit is 0** there.
* Otherwise in other scenarios where output is decided, it is 1.
* V-bit is 0 only when all inputs are 0.

**4-to-2 Encoder**



**Decoders**

* **For example:-** 3-to-8 decoder.
* Truth table drawn totally opposite.

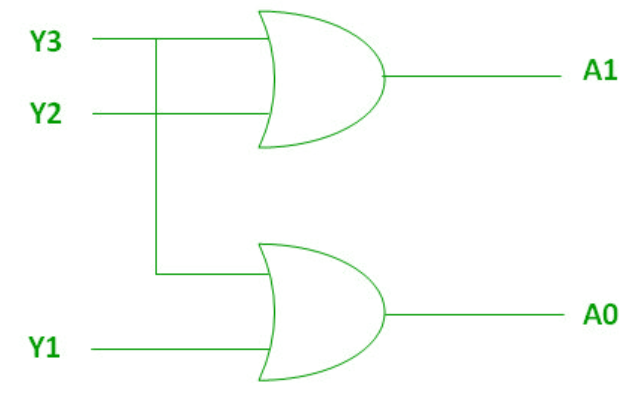


**Encoders**

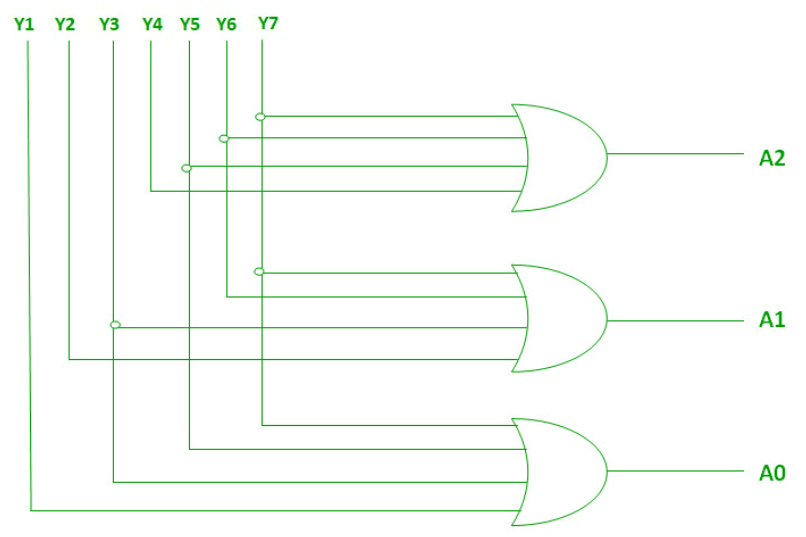
* **Active:** low
* **Inactive:** high
* In encoder, only **one** output is **active low**.
* Converts binary inputs into a certain binary code, of **n** bit.
* **Types of encoders:-**
  + 4 to 2 encoders
  + Octal to binary encoders (8 to 3 encoder)
  + Decimal to BCD encoder
  + Priority encoder

**Some Encoder Diagrams**

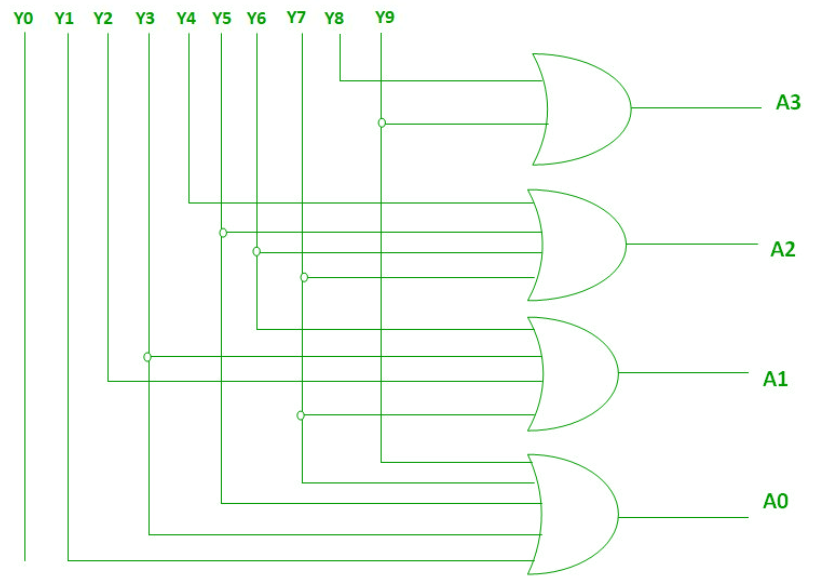
* **4 to 2 encoder:**



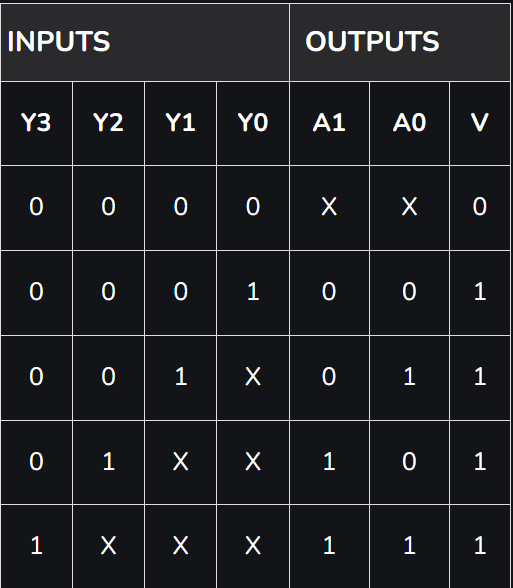
* **8 to 3 encoder:**

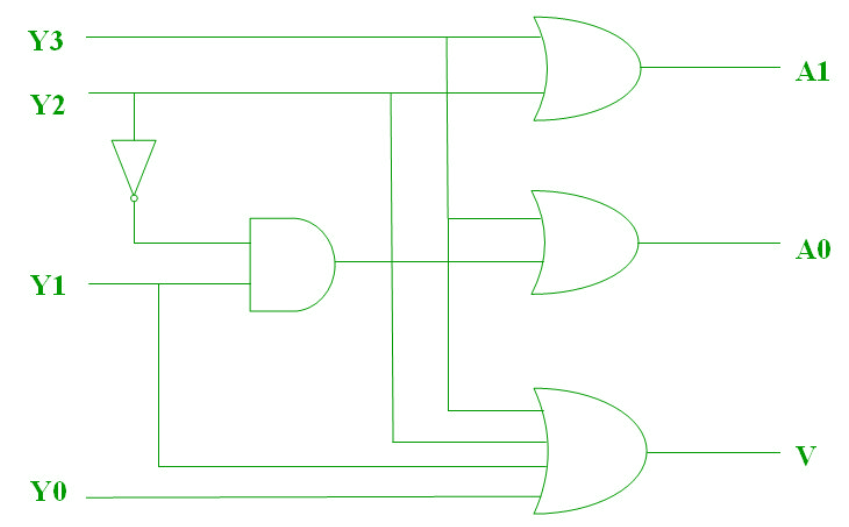


* **Decimal to BCD:**
  + 10 inputs, 4 outputs
  + Truth table similar to other encoders.



* **Priority encoder:**
  + Similar to 4 to 2 encoder.
  + But the triangle below the **1s are don’t care type**.





**Application of Encoders**

* Most digital circuits.
* Basic arithmetic operations.
* Detecting interrupts in microprocessors.

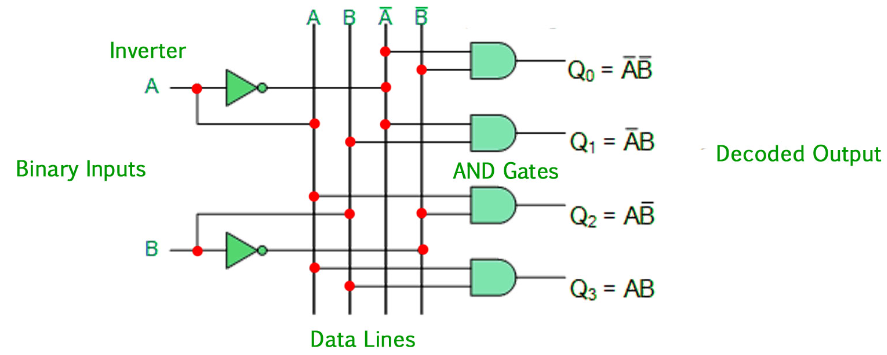
**Merits & Demerits of Encoders**

1. **Advantages:-**
   1. Reduction in number of lines (more inputs to less outputs)
   2. More reliability (less error, due to 1st point)
   3. Improved performance (reduced transmission time)
2. **Disadvantages:-**
   1. More complex circuit.
   2. Limited application.
   3. Less flexible (fixed number inputs and outputs)

**Binary Decoder**

* Converts **serial** codes to **parallel** outputs.
* Only one input is low active.
* Input of **n-bit** and **2n** or less output lines.
* Only one output is activated, others deactivated.
* Unused combinations = Less than 2n outputs
* Each combination of input provides a unique output.

**2-4 Decoder**



* Only one output line is HIGH, determines (decodes) the output code.
* An extra input called **Enable**, denoted by **E**, controls ON and OFF function of the system.
* Enable must always be HIGH in order for the system to work.

**Merits & Demerits of Binary Decoders**

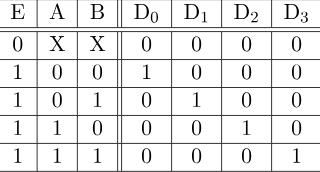
1. **Advantages:-**
   1. Increased flexibility
   2. Improved performance
   3. More reliability (reduced error chances)
2. **Disadvantages:-**
   1. Increased complexity
   2. Limited applications
   3. Limited outputs

**Application of Binary Decoder**

* Memory assigning in computers
* Control circuits
* Display drivers
* Error rectification

**Combinational Circuits Using Decoder**

* A decoder with Enable input acts as demultiplexer.
* So compulsorily 2n output lines (2n minterms).
* **Demux using decoder:**



* Full adder can be constructed by using **3:8 decoder** and **OR gates**.

**Merits & Demerits of These**

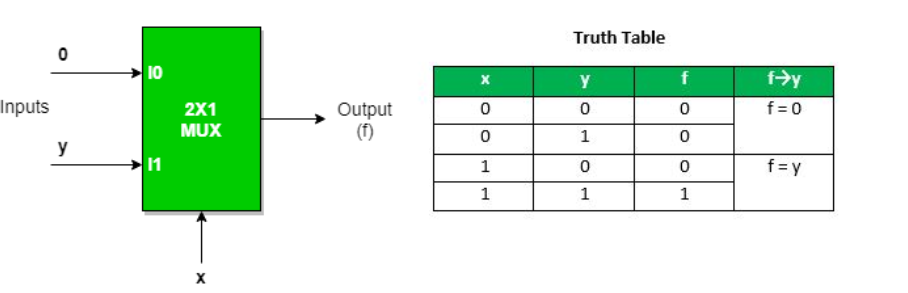
1. **Advantages:-**
   1. Simplification
   2. Flexibility
   3. Modularity
   4. Reliability
2. **Disadvantages:-**
   1. Complexity
   2. Delay
   3. Power utilization
   4. Limited adaptability

**Multiplexers**

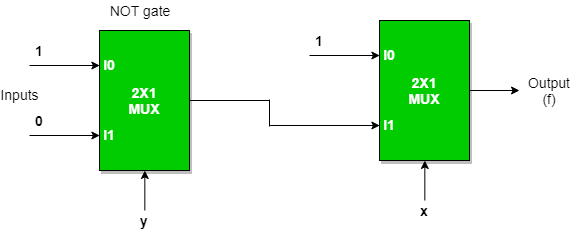
* **Also known as:-**
  + **n** selector
  + Parallel to serial converter
  + Many to one circuit
  + Universal logic circuit
* Used for **increasing** the amount of data that can be carried within certain amount of time and bandwidth.
* **Selector pins** are also known as **control lines**.

**Gates Using Muxes**

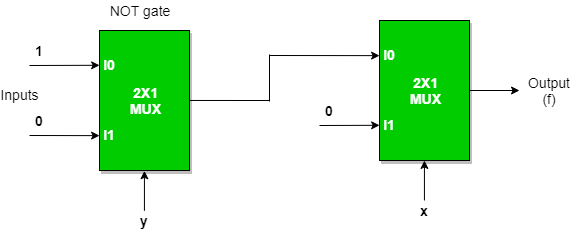
* **AND gate using mux:**



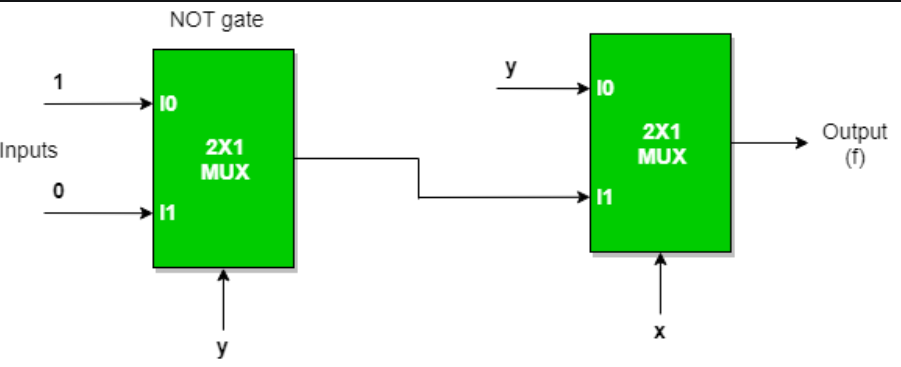
* Implementation of NAND, NOR, XOR, XNOR gate **requires two 2:1 mux**.
* **NAND:**



* **NOR:**



* **XOR:**



**Using Mux for Creating Larger**

**For example:** Creating 64:1 mux using 4:1.

**64:1 -> m:1, 4:1 -> n:1**

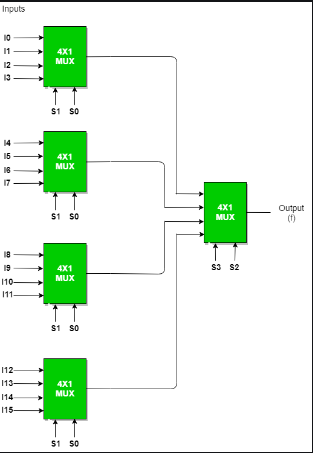
Continuous **m/n** until answer is 1.

**64/4 = 16**

**16/4 = 4**

**4/4 = 1**

We did so 3 times, so 3+1 = 4. **Four** 4:1 mux are required for making a 64:1 mux.



**Merits & Demerits of Multiplexers**

1. **Advantages:-**
   1. Space-saving
   2. Cost-successful
   3. Time-saving
   4. Flexibility
2. **Disadvantages:-**
   1. Limited input numbers
   2. Delay
   3. Complex connections
   4. Power utilization

Hazards

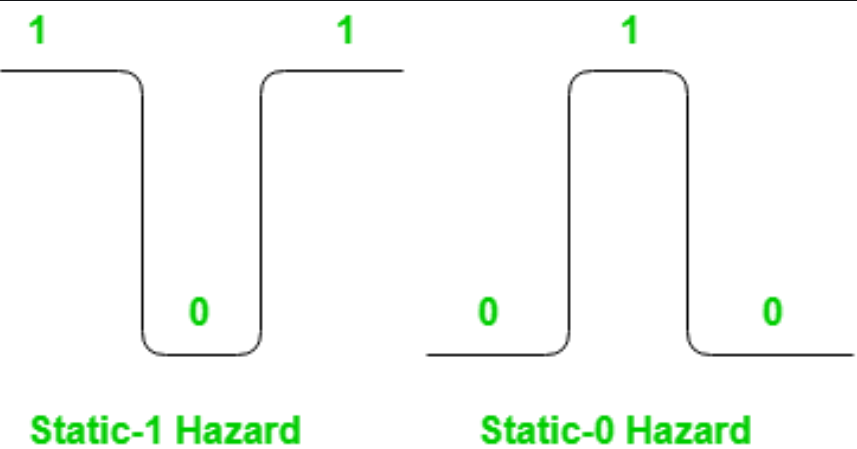
* Temporary fluctuation in digital device.
* Gets resolved by itself.
* Updates wrong output momentarily due to propagation delay.
* Of Three Types:-
  + Static hazard
  + Dynamic hazard
  + Function hazard

Static-1 Hazard

* Occurs in SOP circuits.
* Output is 1, but momentarily changes to 0 before becoming 1.

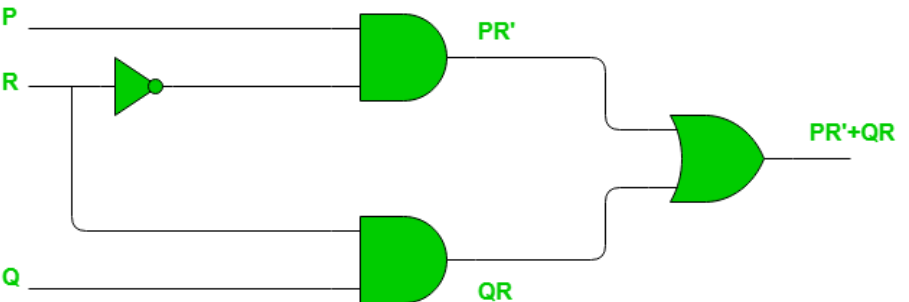
Static-0 Hazard

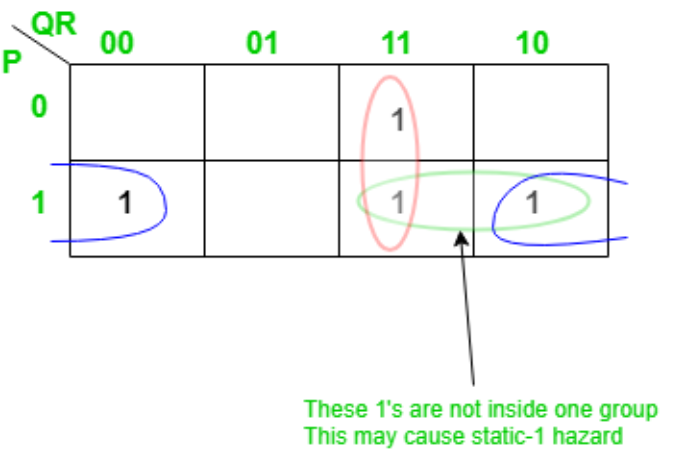
\*you know it now\*



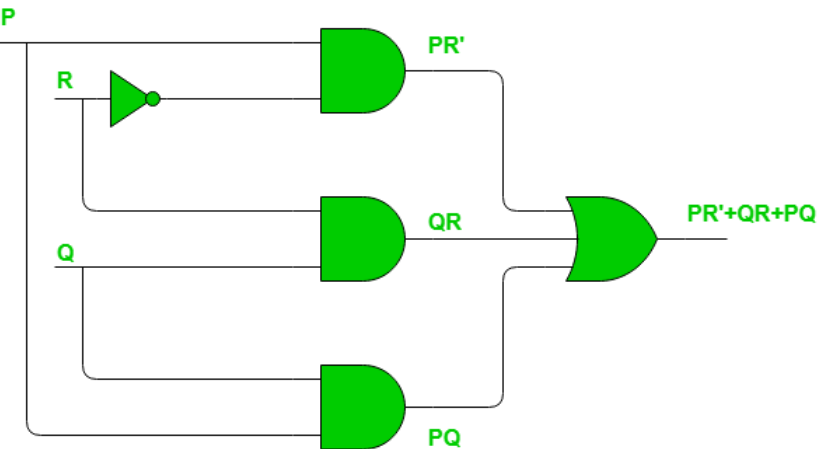
Static Hazards on K-Map

* Mark pair of cells that do not occur to be in same group (prime implicants).
* Each such pair is static-1 hazard.





* Some changes in circuit are made in order to fix hazard.



* Similarly, in static-0 hazard we consider 0s instead 1s.