Combinational Logic Circuits

Unit: 2

DIGITAL LOGIC & CIRCUIT DESIGN

SUBJECT CODE: ACSE0304

B Tech: 3rd Sem.

Ms. Nidhi Sharma

Associate Professor

Department of Electronics and Communication Engineering

Brief Introduction

Ms. Nidhi Sharma

• PhD	Banasthali Vidyapith, Banasthali
• M.Tech	MDU, Rothak, India
• B.Tech.	CCS University, Meerut, Uttar Pradesh

Research Interests

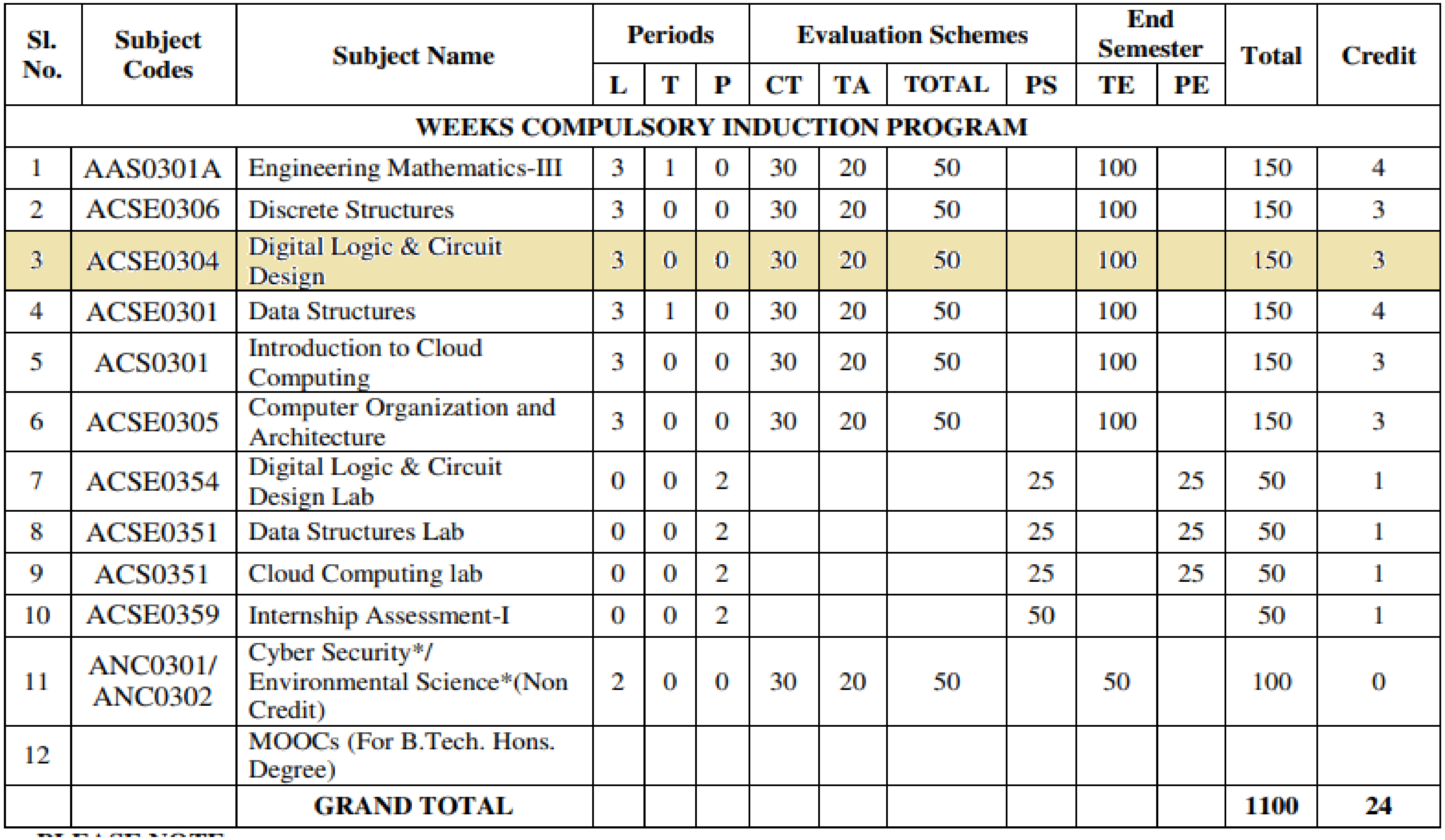
Optoelectronics , Communication System, Wireless Communication, Internet of Things

Academic Experience: (12+ years)

Publication : 16 (SCI), 04 (Scopus), 1(Book Chapter),10(International Conference)

NOIDA INSTITUTE OF ENGINEERING AND TECHNOLOGY, GREATER NOIDA

Evaluation Scheme



Sl. No.	Subject Codes	Subject Name	Periods			Evaluation Schemes				End Semester		Total	Credit
			L	T	P	CT	TA	TOTAL	PS	TE	PE		
WEEKS COMPULSORY INDUCTION PROGRAM													
1	AAS0301A	Engineering Mathematics-III	3	1	0	30	20	50		100		150	4
2	ACSE0306	Discrete Structures	3	0	0	30	20	50		100		150	3
3	ACSE0304	Digital Logic & Circuit Design	3	0	0	30	20	50		100		150	3
4	ACSE0301	Data Structures	3	1	0	30	20	50		100		150	4
5	ACS0301	Introduction to Cloud Computing	3	0	0	30	20	50		100		150	3
6	ACSE0305	Computer Organization and Architecture	3	0	0	30	20	50		100		150	3
7	ACSE0354	Digital Logic & Circuit Design Lab	0	0	2				25		25	50	1
8	ACSE0351	Data Structures Lab	0	0	2				25		25	50	1
9	ACS0351	Cloud Computing lab	0	0	2				25		25	50	1
10	ACSE0359	Internship Assessment-I	0	0	2				50			50	1
11	ANC0301/ ANC0302	Cyber Security*/ Environmental Science*(Non Credit)	2	0	0	30	20	50		50		100	0
12		MOOCs (For B.Tech. Hons. Degree)											
		GRAND TOTAL										1100	24

Course Contents / Syllabus

UNIT-I: Digital System and Binary Numbers: Number System and its arithmetic, Signed binary numbers, Binary codes, Cyclic codes, Hamming Code, Simplification of Boolean Expression: K-map method up to five variable, SOP and POS Simplification Don't Care Conditions, NAND and NOR implementation, Quine Mc-Clusky Method (Tabular Method).

UNIT II : Combinational Logic: Combinational Circuits: Analysis Procedure, Design Procedure, Code Converter, Binary Adder-Subtractor, Decimal Adder, Binary Multiplier, Magnitude Comparator, Decoders, Encoders Multiplexers, Demultiplexers.

UNIT III: Sequential Logic and Its Applications: Storage elements: Latches & Flip Flops, Characteristic Equations of Flip Flops, Excitation Table of Flip Flops, Flip Flop Conversion, Registers, Shift Registers, Ripple Counters, Synchronous Counters, Other Counters: Johnson & Ring Counter.

UNIT IV: Synchronous & Asynchronous Sequential Circuits: Analysis of clocked Sequential Circuits with State Machine Designing, State Reduction and Assignments, Design Procedure.
Analysis procedure of Asynchronous Sequential Circuits, Circuit with Latches, Design Procedure, Reduction of State and flow Table, Race-free State Assignment, Hazards.

UNIT-V: Memory & Programmable Logic Devices: Basic concepts and hierarchy of Memory, Memory Decoding, RAM: SRAM, DRAM, ROM: PROM, EPROM, Auxiliary Memories, PLDs: PLA, PAL; Circuit Implementation using ROM, PLA and PAL; CPLD and FPGA..

Content

Course Objective

The intended objectives of this course are given as follows:

- To learn number representation and conversion between different representation.
- To analyze logic processes and implement logical operations using combinational logic circuits.
- To learn concepts of sequential circuits and analyze sequential circuits in terms of state machine.
- To learn to design synchronus and asynchronus circuits.
- To learn concept of memories and PLDs.

Unit Objective

The intended objectives of this unit are:

1. To learn various combinational logic circuits such as half- adder, full-adder, Multiplexer (MUX), Demultiplexer (DEMUX).
2. To understand the design and analysis procedure of combinational circuits.
3. To learn serial adder and Barrel shifter.

Course Outcome

At the end of this course students will able to:

- Develop a digital logic and apply it to solve real life problems.
- Design and analyze modular combinational circuits with MUX / DEMUX, Decoder & Encoder.
- Design and analyze different sequential circuits and their applications.
- Classify digital logic families, memories and implement logic circuits through programmable logic devices.
- Understand and analyze ADC and DAC.

NIET Greater Noida
GET FUTURE READY

CO-PO and PSO Mapping

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
ACSE0304.1	3	2	-	-	-	-	-	-	-	-	-	2
ACSE0304.2	3	3	2	-	-	-	-	-	-	-	-	2
ACSE0304.3	2	3	2	2	-	-	-	-	-	-	-	2
ACSE0304.4	3	3	3	2	-	-	-	-	-	-	-	2
ACSE0304.5	3	2	1	-	-	-	-	-	-	-	-	2
AVERAGE	2.8	2.6	2	2	-	-	-	-	-	-	-	2

Course Outcome	PSO1	PSO2	PSO3
ACSE0304.1	3	-	3
ACSE0304.2	2	-	3
ACSE0304.3	2	-	3
ACSE0304.4	2	-	3
ACSE0304.5	2	-	3
Average	2.2	-	3

NOIDA INSTITUTE OF ENGINEERING AND TECHNOLOGY, GREATER NOIDA

ROAD MAP

- PRE REQUISIT E
- INTRODU CTION (CONCEP TS / THEORY)
- NUMERIC AL PRACTIC E
- QUIZZE ASSIGNM ENTS & UNIT TESTS
- L A B O R A T O R Y
- PERFOR MANCE EVALUAT ION (Mid Term, End Sem Exam)

Introduction to Combinational Logic Circuits

Topic Objective	Mapping with CO
To understand the concept of combinational circuit.	CO2
To know the design procedure for combinational circuits.	CO2
To understand various combinational circuits like adders and subractors.	CO2

Introduction to Combinational Logic Circuits

Prerequisite:

- Basic knowledge of number system.
- Basic knowledge of logic gates.
- Basic knowledge of k-map up to 4 & 5 variables.

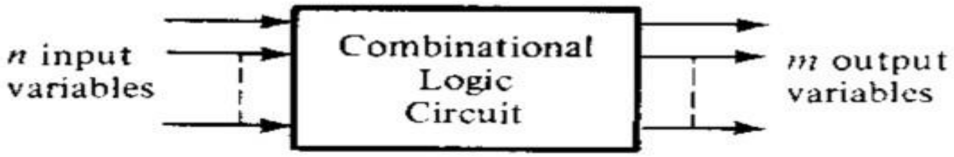
Introduction

$$\text{LOGIC CIRCUITS:}\begin{cases}\text{1. Combinational}\\[2ex]\text{2. Sequential}\end{cases}$$

- Combinational Logic Circuits (Circuits without a memory): In this type of logic circuits outputs depend only on the current inputs.
- Sequential Logic Circuits (Circuits with memory): In this type of logic circuits outputs depend on the current inputs and previous inputs. These circuits employs storage elements and logic gates.

Combinational Logic Circuits

- A combinational circuit consists of input variables (n), logic gates, and output variables (m).



- For (*n*) input variables there are 2^n possible combinations of binary input values.
- For each possible input combination there is one and only one possible output combination, a combinational circuit can be describe by (*m*) Boolean functions one for each output variable.
- Each output function expressed in terms of the (*n*) input variables

Combinational Logic Circuits

To design a combinational circuits use the following steps:

- The problem is stated (Verbal description).
- Specify the number of inputs and required numbers of outputs. The input and output variables are assigned letter symbols.
- Construct the truth table to define relationship between inputs and outputs.
- The simplified Boolean function for each output is obtained (using K-Map, Tabulation method and Boolean Algebra rules).
- The logic diagram is drawn.

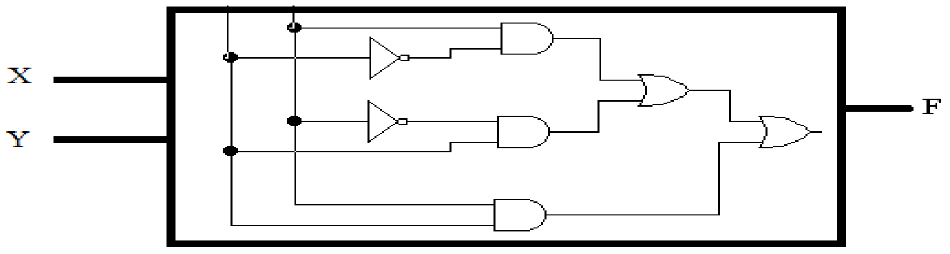
Combinational Logic Circuits

Example: Simplify two inputs OR gate truth table by using K-Map?

- Problem is stated.
- No. of inputs are two (X and Y) & No. of required output is (F)
- Construct truth table (F= X+Y), inputs =2 =4 possibilities
- Using K-Map to simplify the circuit.
- Final design.

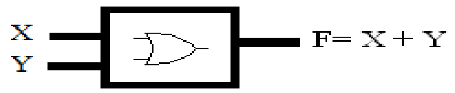
$F = X + Y$ .... (Two inputs OR gate equation)

X	Y	F
0	0	0
0	1	1
1	0	1
1	1	1



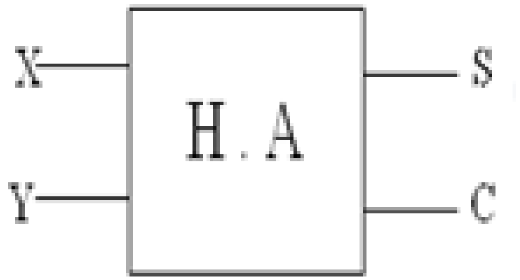
$F = \overline{X}Y + X\overline{Y} + XY$

X \ Y	0	1
0	0	1
1	1	1



Half Adder

- Half adder is a combinational arithmetic circuit that adds two numbers and produces a sum bit(S) and carry bit (C) as the output. If A and B are the input bits ,then sum bit(S) is the X-OR of X and Y and the carry bit (C) will be the AND of X and Y.
- The block diagram of half adder is:

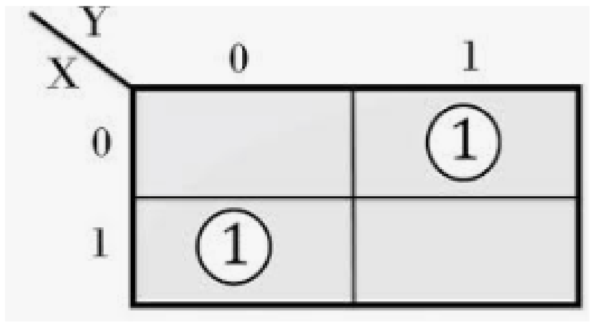


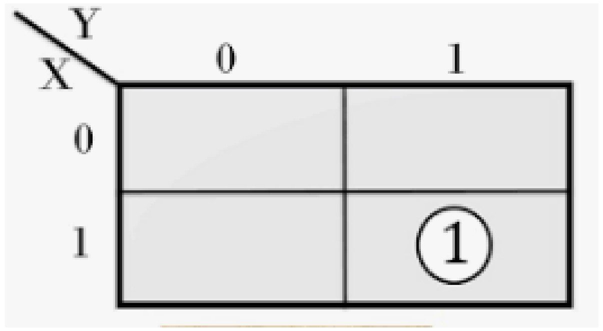
- Truth Table for half adder is:

Inputs		Outputs	
X	Y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Half Adder

- K-map of sum and carry for half adder:





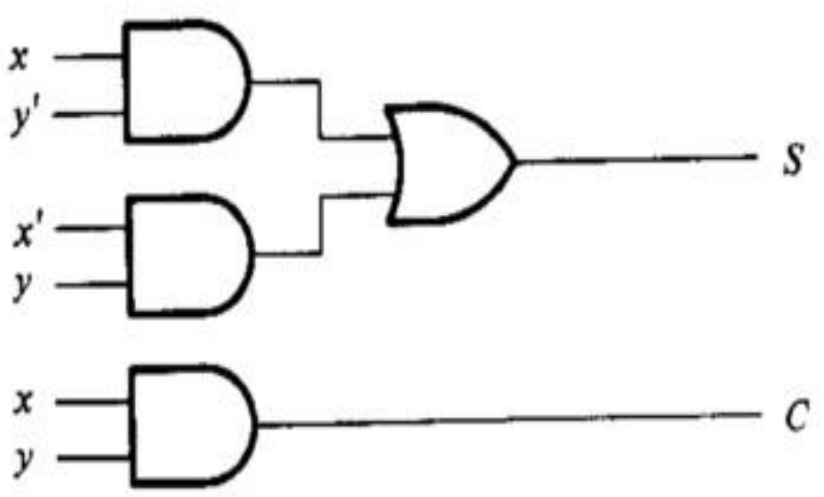
- From K-map Boolean equations for sum and carry are:

$$S = \overline{X}Y + X\overline{Y}$$

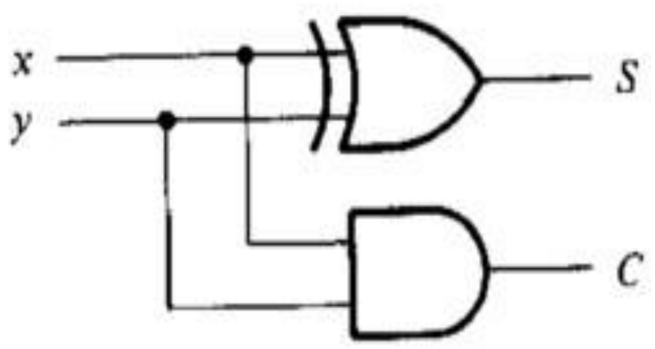
$$C = XY$$

Half Adder

- Logic diagram of half adder:



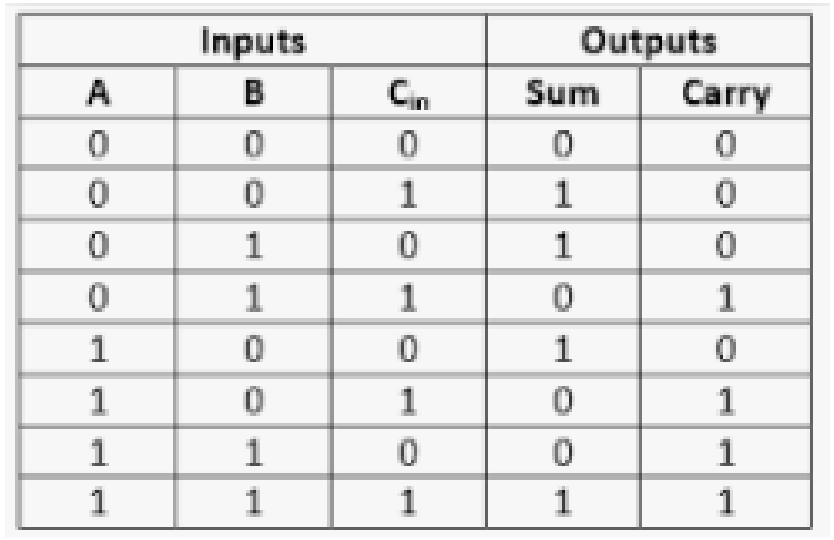
(a) $S = xy' + x'y$
$C = xy$



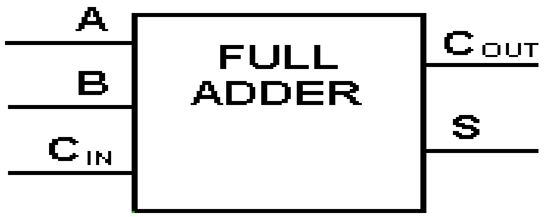
(b) $S = x \oplus y$
$C = xy$

Full Adder

- Full adder is developed to overcome the drawbacks of Half Adder circuit. It can add two one-bit numbers A and B, sum S and carry C.
- The full adder is a three input and two output combinational circuit
- **Truth Table :** **Block Diagram:**

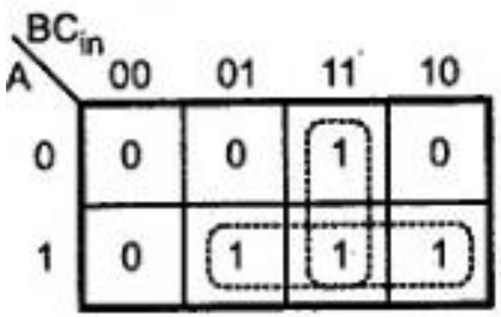


Inputs			Outputs	
A	B	C_{in}	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

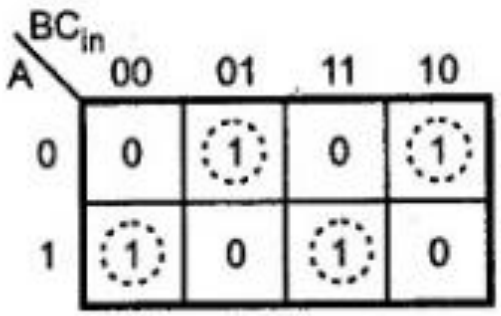


Full Adder

For Carry (C_{out})

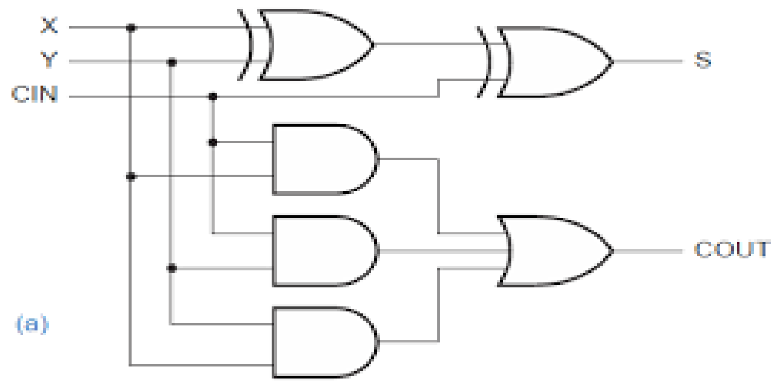


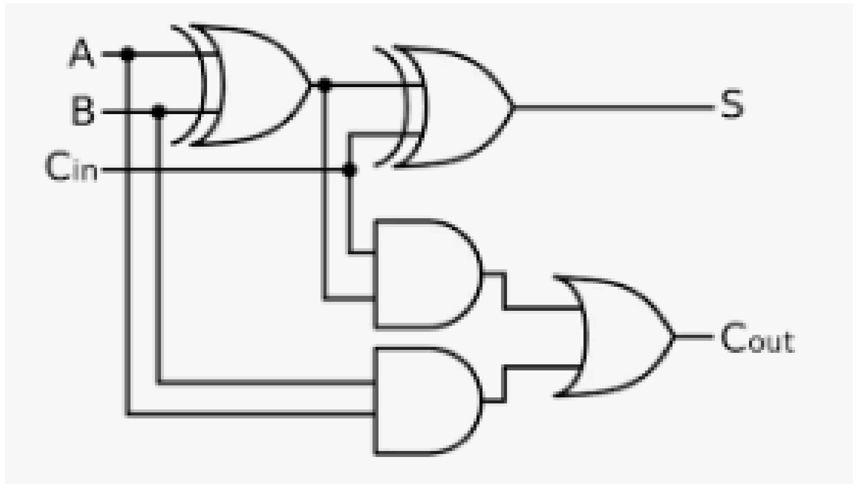
For Sum



- **Logical Expression for SUM:**
  $= A' B' C_{in} + A' B C_{in}' + A B' C_{in}' + A B C_{in}$
  $= C_{in} (A' B' + A B) + C_{in}' (A' B + A B')$
  $= C_{in}$ XOR (A XOR B)
  = A XOR B XOR C_{in}
- **Logical Expression for Carry:**
  $= AB + BC_{in} + C_{in} A$

Full Adder





Circuit diagram of full adder

Sum = A XOR B XOR C_{in}
Carry = AB + BC_{in} + C_{in} A

Full Adder

- **Full Adder using two half adders :**

Sum = = A XOR B XOR C_{in}

Carry = AB + BC_{in} + C_{in} A

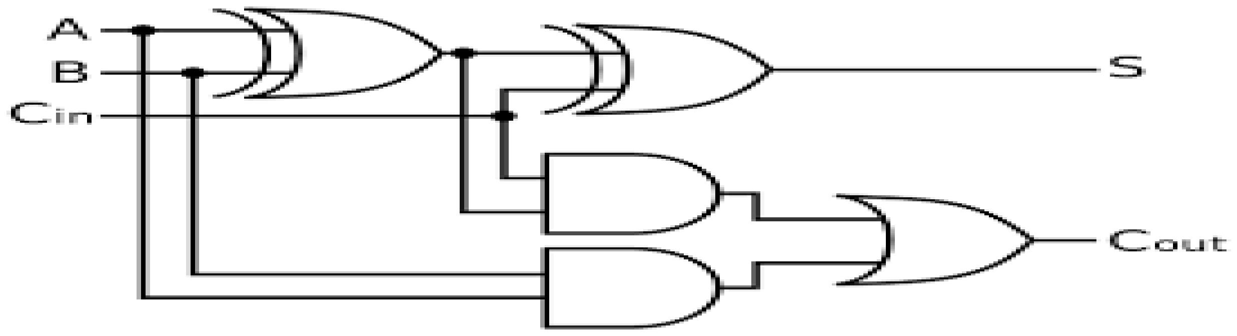
$= A\,B + A\,C_{in} + B\,C_{in}\,(A + A') = A\,B\,C_{in} + A\,B + A\,C_{in} + A'\,B\,C_{in}$

$= A\,B\,(1 + C_{in}) + A\,C_{in} + A'\,B\,C_{in} = A\,B + A\,C_{in} + A'\,B\,C_{in}$

$= A\,B + A\,C_{in}\,(B + B') + A'\,B\,C_{in} = A\,B\,C_{in} + A\,B + A\,B'\,C_{in} + A'\,B\,C_{in}$

$= A\,B\,(C_{in} + 1) + A\,B'\,C_{in} + A'\,B\,C_{in} = A\,B + A\,B'\,C_{in} + A'\,B\,C_{in}$

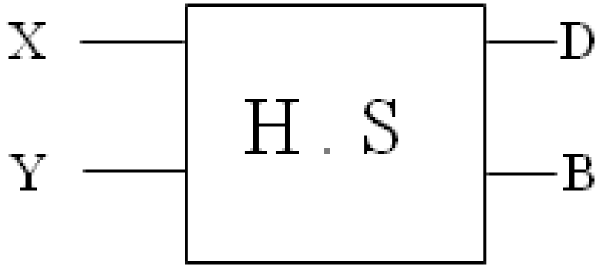
$= AB + C_{in}\,(A'\,B + A\,B') = AB + C_{in}\,(A \text{ xor } B)$



Half Subtractor

- Half subtractor is an arithmetic circuit that subtracts two binary numbers from each other, for example, X – Y to find the resulting difference between the two numbers.
- Half subtractor produces a difference (D) by using a borrow bit(B) from the previous column.
- Block Diagram and Truth Table :

Inputs		Outputs	
X	Y	B	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

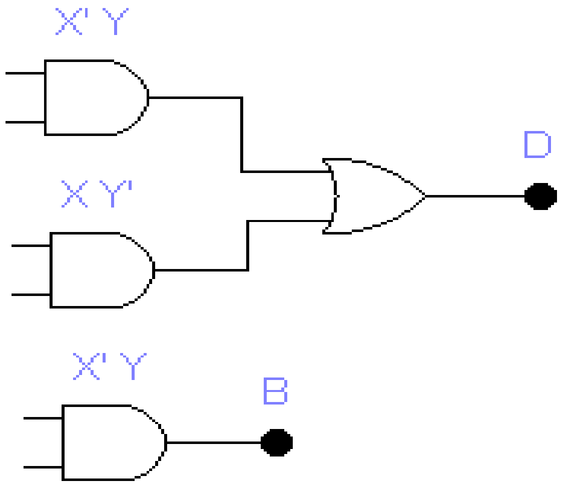


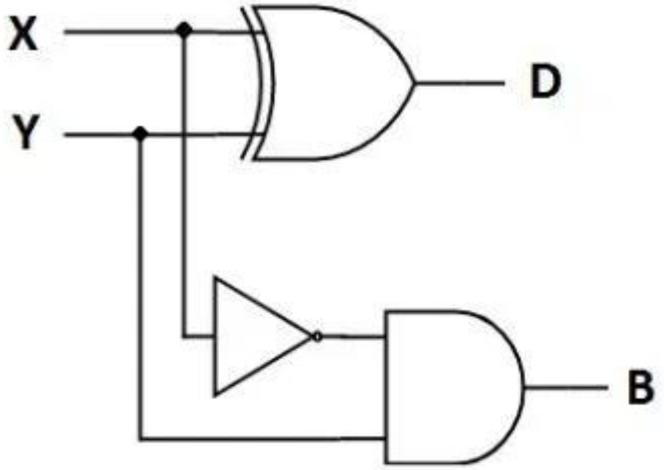
$$D = \overline{X}Y + X\overline{Y} = X \oplus Y$$

$$B = \overline{X}Y$$

Half Subtractor

Logic Diagram of Half Subtractor :



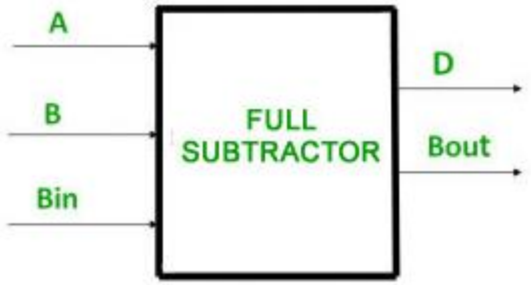


$$D = \overline{X}Y + X\overline{Y} = X \oplus Y$$

$$B = \overline{X}Y$$

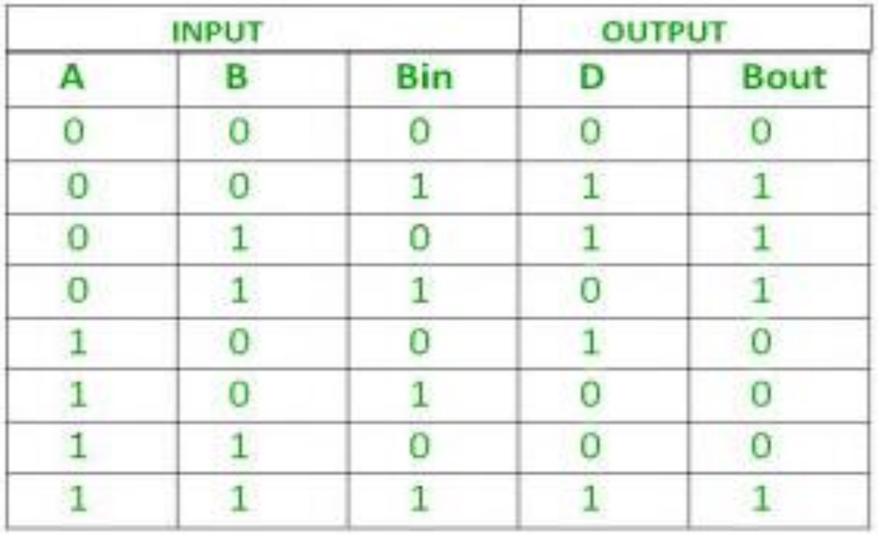
Full Subtractor

- A full subtractor is a combinational circuit that performs subtraction of two bits, one is minuend and other is subtrahend, taking into account borrow of the previous adjacent lower minuend bit. This circuit has three inputs and two outputs. The three inputs A, B and Bin, denote the minuend, subtrahend, and previous borrow, respectively. The two outputs, D and Bout represent the difference and output borrow.



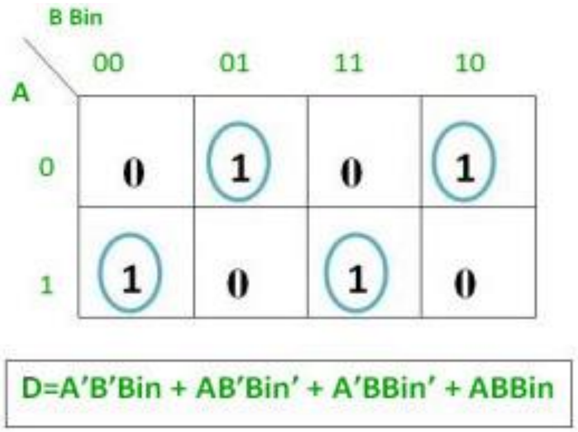
Full Subtractor

- **Truth Table of Full subtractor:**



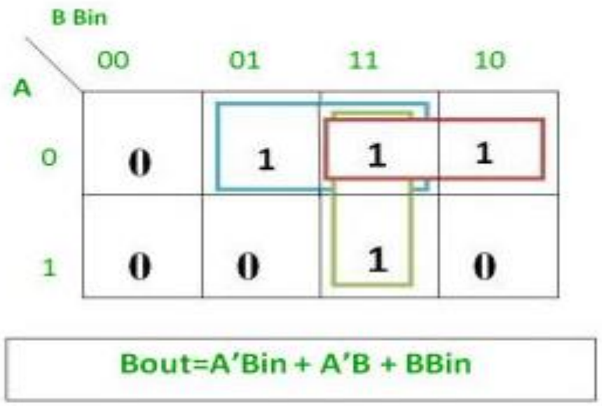
INPUT			OUTPUT	
A	B	Bin	D	Bout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

- **K-map for full subtractor:**



$$\begin{array}{c|cccc} A \backslash B\,Bin & 00 & 01 & 11 & 10 \\ \hline 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{array}$$

D=A'B'Bin + AB'Bin' + A'BBin' + ABBin



$$\begin{array}{c|cccc} A \backslash B\,Bin & 00 & 01 & 11 & 10 \\ \hline 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{array}$$

Bout=A'Bin + A'B + BBin

Full Subtractor

- **Logical expression for difference –**

  D = A’B’Bin + A’BBin’ + AB’Bin’ + ABBin = Bin(A’B’ + AB) + Bin’(AB’ + A’B)

  = Bin(A XNOR B) + Bin’(A XOR B) = Bin (A XOR B)’ + Bin’(A XOR B)

  = Bin XOR (A XOR B) = (A XOR B) XOR Bin

- **Logical expression for borrow –**

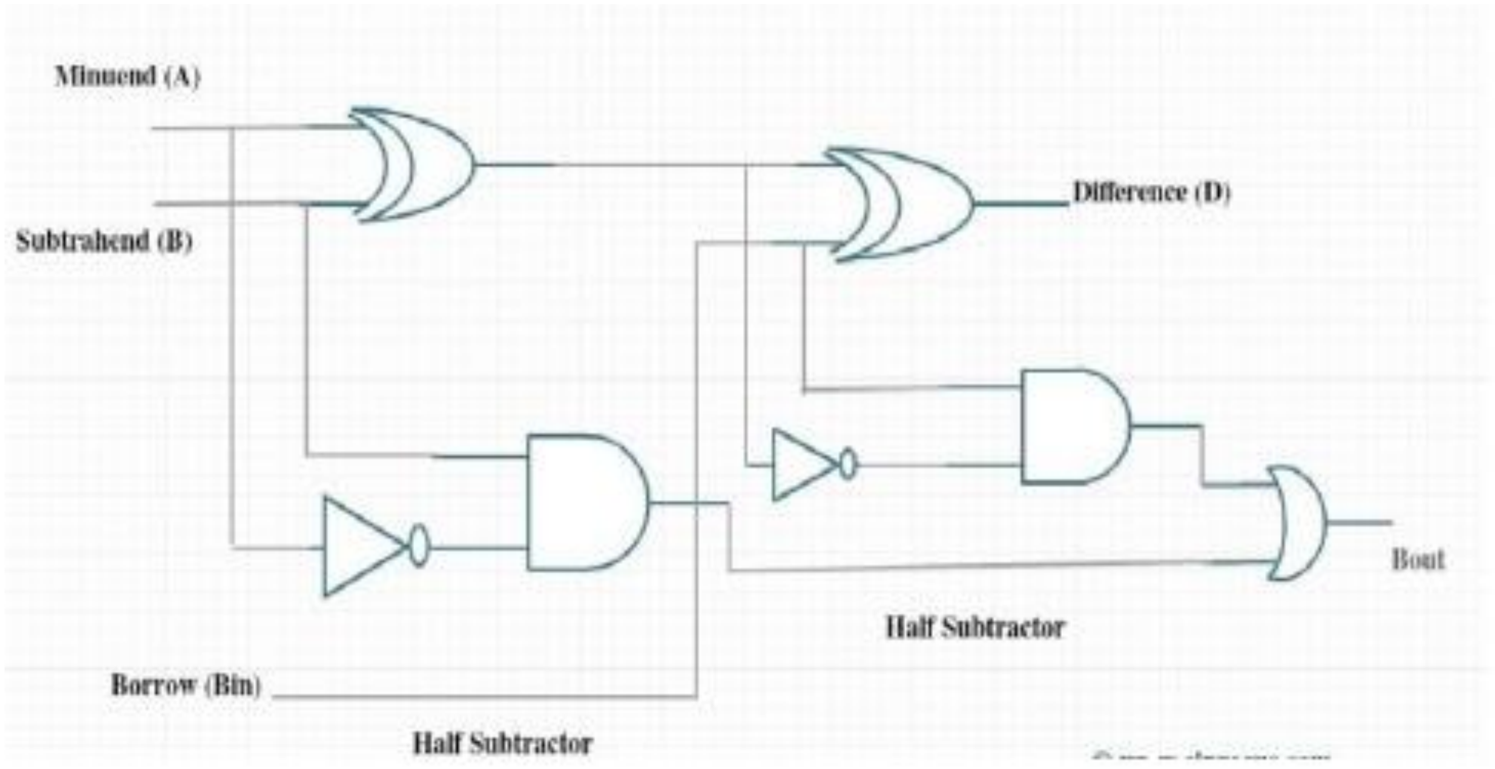
  Bout = A’B’Bin + A’BBin’ + A’BBin + ABBin

  = Bin(AB + A’B’) + A’B(Bin + Bin’) = Bin(A XNOR B) + A’B

  = Bin (A XOR B)’ + A’B

Full Subtractor

Full Subtractor using two half subtractor:



D = (A XOR B) XOR Bin & Bout = Bin (A XOR B)’ + A’B

Daily Quiz

- What are the steps to design any combinational circuits.
- How many NAND are needed to implement a half adder:

a) 2
b) 3
c) 4
d) 5

- How many NAND are needed to implement a full adder:

a) 2
b) 8
c) 9
d) 12

- The difference bit output of a half subtractor is the same as:

a) Difference bit output of a full subtractor
b) sum bit output of a full adder
c) sum bit output of a half adder
d) carry bit output of a half adder

Faculty Video Links, Youtube & NPTEL Video Links and Online Courses Details

- Youtube/other Video Links
  - https://www.youtube.com/watch?v=aLUY-s7LSns&ab_channel=NesoAcademyNesoAcademy
  - https://www.youtube.com/watch?v=36hCizOk4PA&ab_channel=nptelhrd
  - https://www.youtube.com/watch?v=CwIk8lyrMI8&ab_channel=nptelhrd

Old Papers

- Implement full adder using two half adder.
- Implement Full adder using NAND gates only.
- Design and explain carry look ahead adder.
- Implement Full subtractor using NOR gates only.

Recap

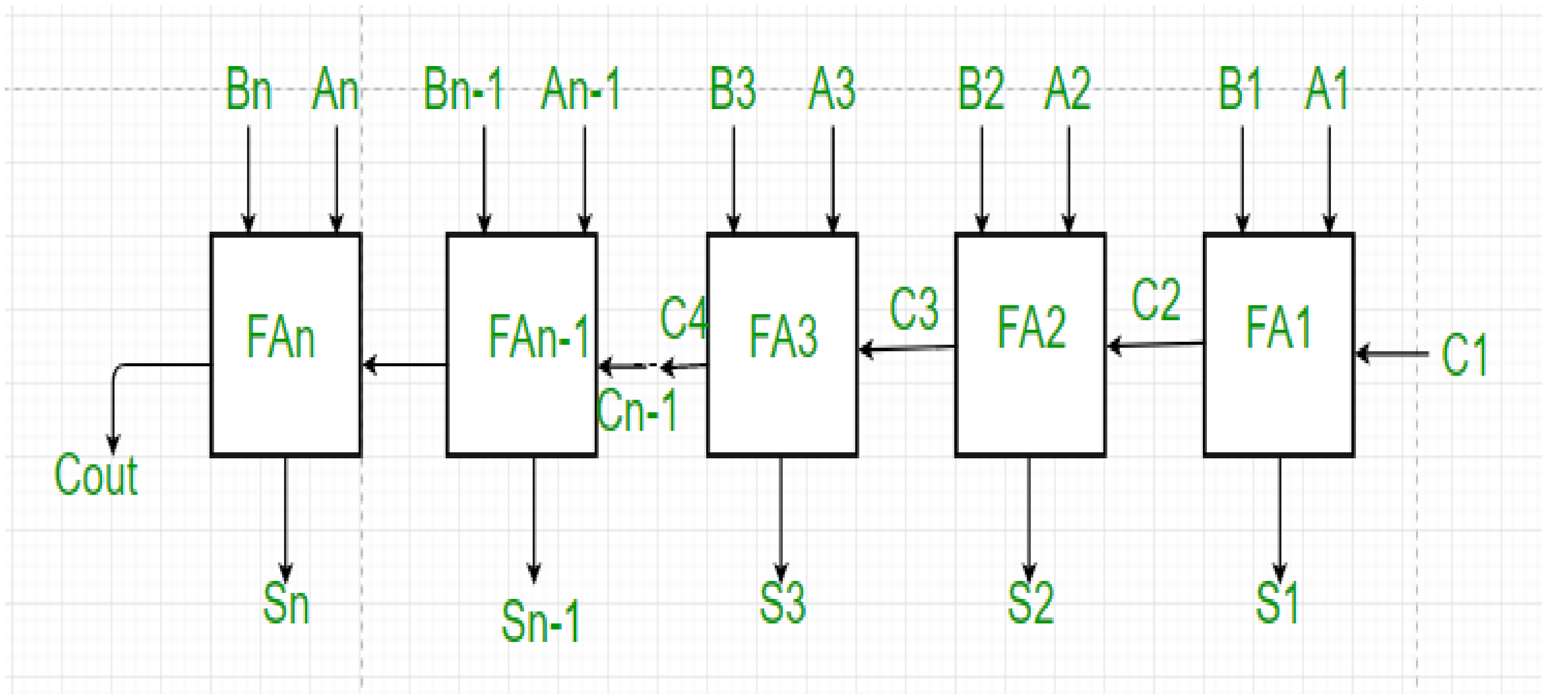
- A combinational circuit consists of input variables (n), logic gates, and output variables (m).
- To design a combinational circuits at first specify the number of inputs and required numbers of outputs then write the simplified Boolean function for each output is obtained (using K-Map, Tabulation method and Boolean Algebra rules). Then logic diagram is drawn.
- Adder and subtractor are the basic combinational arithmetic circuits.

Parallel Adder

- A single full adder performs the addition of two one bit numbers and an input carry. But a Parallel Adder is a digital circuit capable of finding the arithmetic sum of two binary numbers that is greater than one bit in length by operating on corresponding pairs of bits in parallel.
- It consists of full adders connected in a chain where the output carry from each full adder is connected to the carry input of the next higher order full adder in the chain.
- A n bit parallel adder requires n full adders to perform the operation.

Parallel Adder

Logic Diagram of Parallel Adder :



Inputs:

A1 A2 A3 A4……A $_{(n-1)}$ A$_n$ for A

B1 B2 B3 B4…….B $_{(n-1)}$ B$_n$ for B

Parallel Adder

Working of parallel Adder –

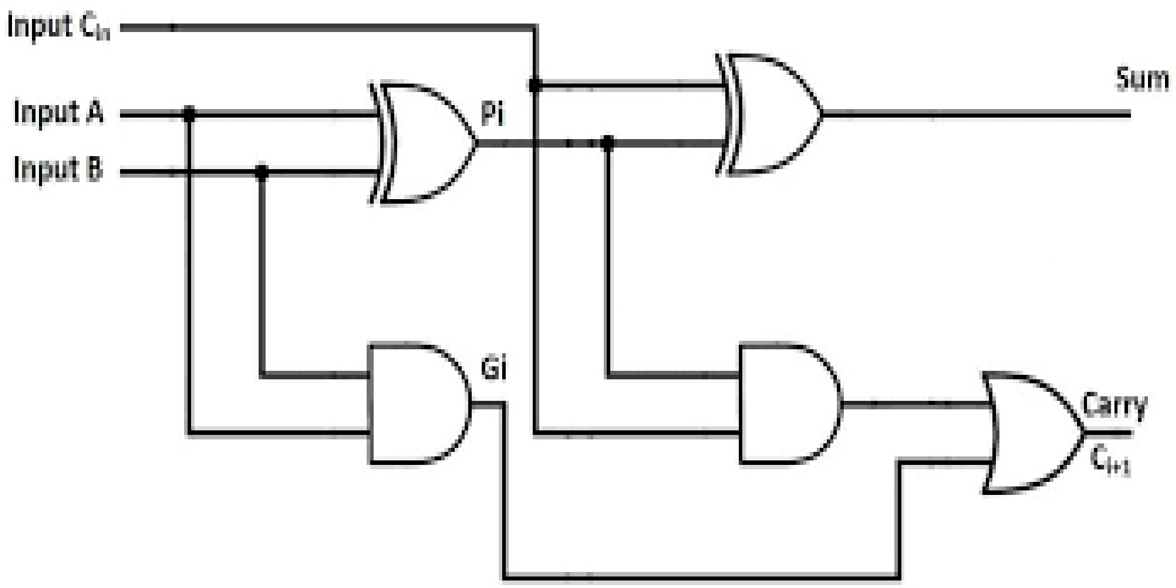
- As shown in the figure, firstly the full adder FA1 adds A1 and B1 along with the carry C1 to generate the sum S1 (the first bit of the output sum) and the carry C2 which is connected to the next adder in chain.
- Next, the full adder FA2 uses this carry bit C2 to add with the input bits A2 and B2 to generate the sum S2(the second bit of the output sum) and the carry C3 which is again further connected to the next adder in chain and so on.
- The process continues till the last full adder Fan uses the carry bit Cn to add with its input An and Bn to generate the last bit of the output along last carry bit Cout.

Parallel Adder

- The speed of parallel adder depends on the carry propagation time. However, signals must be propagated through the gates at a given enough time to produce the correct or desired output.
- The following are the methods to get the high speed in the parallel adder to produce the binary addition.
  - By employing faster gates with reduced delays, we can reduce the propagation delay. But there will be a capability limit for every physical logic gate.
  - Another way is to increase the circuit complexity in order to reduce the carry delay time. There are several methods available, one commonly used method employs the principle of look ahead-carry addition by eliminating inter stage carry logic.

Carry-Look-ahead Adder

- A carry Look-ahead adder is a fast parallel adder as it reduces the propagation delay by more complex hardware, hence it is costlier.
- This method makes use of logic gates so as to look at the lower order bits of the augend and addend to see whether a higher order carry is to be generated or not.



Carry Look-ahead Adder

- Consider the full adder circuit shown above with corresponding truth table. If we define two variables as carry generate G_i and carry propagate P_i then,
  - $P_i = A_i \oplus B_i$
  - $G_i = A_i B_i$
- The sum output and carry output can be expressed as
  - $S_i = P_i \oplus C_i$
  - $C_{i+1} = G_i + P_i C_i$
- Where G_i is a carry generate which produces the carry when both A_i, B_i are one regardless of the input carry. P_i is a carry propagate and it is associate with the propagation of carry from C_i to C_{i+1}

Carry Look-ahead Adder

- The carry output Boolean function of each stage in a 4 stage carry Look-ahead adder can be expressed as
- $C_1 = G_0 + P_0 C_0$
- $C_2 = G_1 + P_1 C_1$

  $= G_1 + P_1 G_0 + P_1 P_0 C_0$
- $C_3 = G_2 + P_2 C_2$

  $= G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$

- From the above Boolean equations we can observe that C_3 does not have to wait for C_2 and C_1 to propagate but actually C_3 is propagated at the same time as C_2 and C_1. Since the Boolean expression for each carry output is the sum of products so these can be implemented with one level of AND gates followed by an OR gate.

Carry-Look-ahead Adder

- The implementation of three Boolean functions for each carry output (C1, C2 and C3) for a carry Look-ahead carry generator shown in below figure.

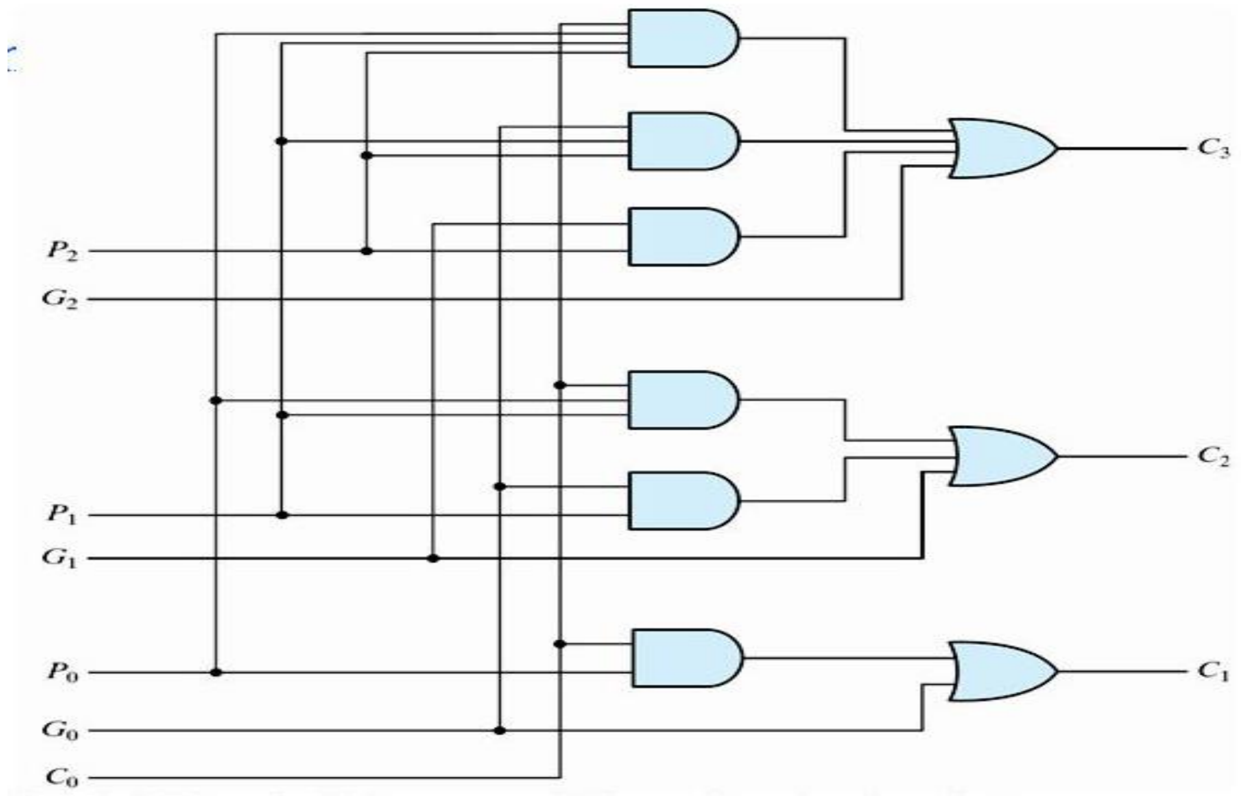
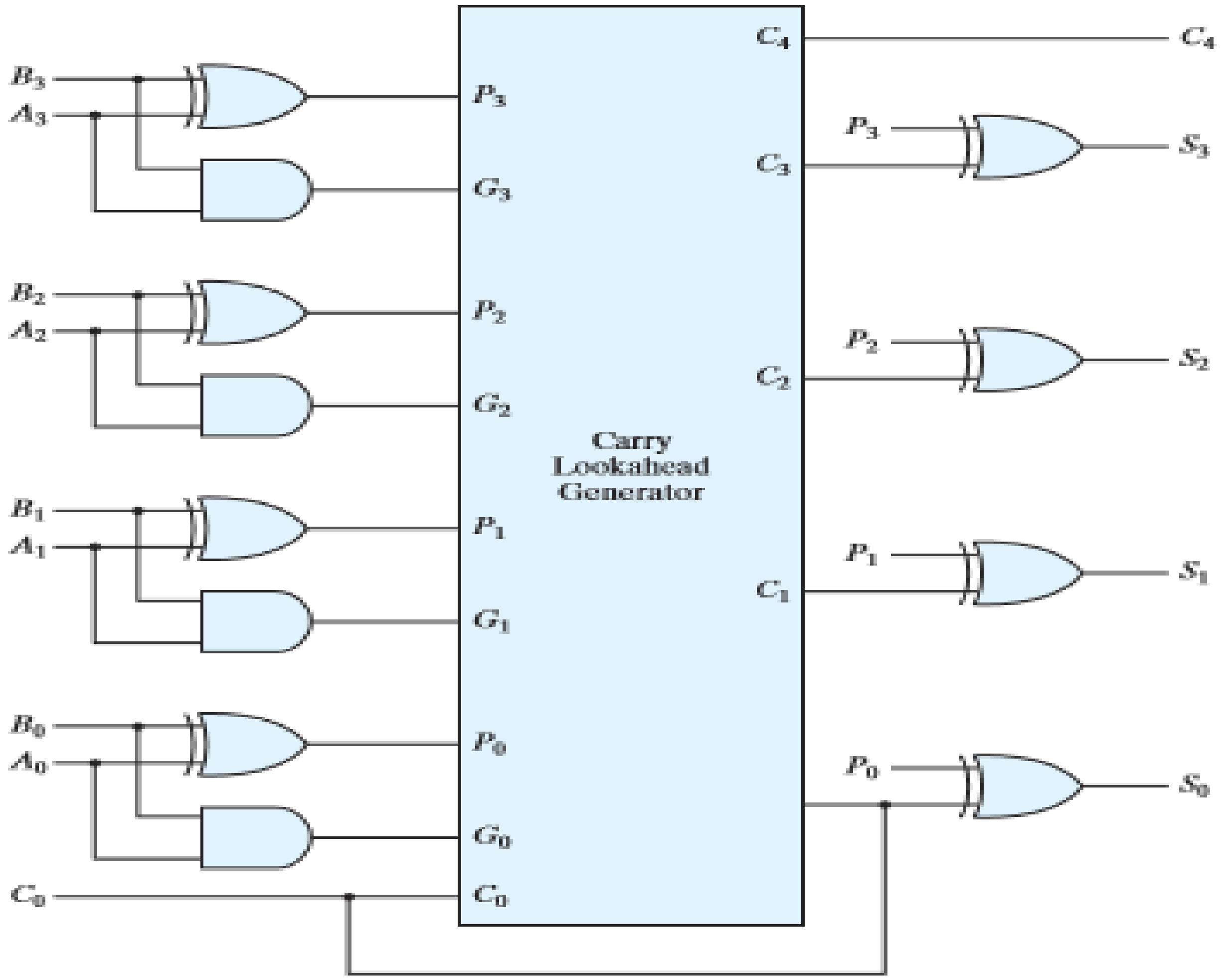


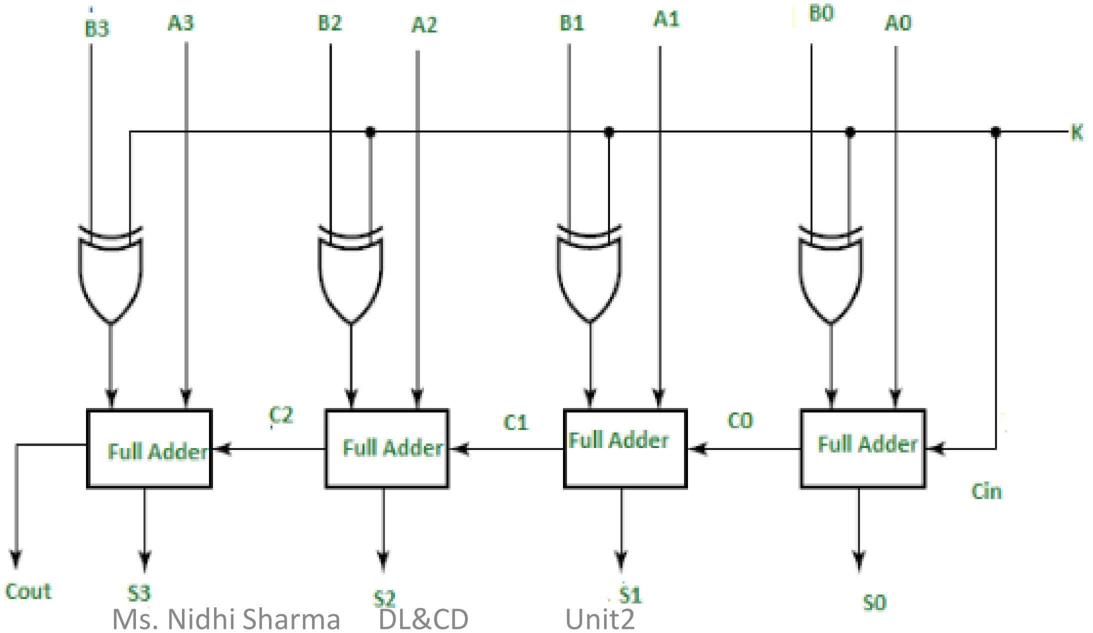
Fig. 4.11 Logic Diagram of Carry Look-ahead Generator

Carry Look-ahead Adder



Parallel Adder/ Subtractor

- Lets consider two 4-bit binary numbers A and B as inputs to the Digital Circuit for the operation with digits
- Inputs A0 A1 A2 A3 for A and inputs B0 B1 B2 B3 for B
- The circuit consists of 4 full adders since we are performing operation on 4-bit numbers. There is a control line K that holds a binary value of either 0 or 1 which determines that the operation being carried out is addition or subtraction.



Parallel Adder/ Subtractor

- As shown in the figure, the first full adder has control line directly as its input(input carry C0), The input A0 (The least significant bit of A) is directly input in the full adder.

- The third input is the exor of B0 and K. The two outputs produced are Sum/Difference (S0) and Carry (C1).

- If the value of K (Control line) is 1, the output of B0(exor)K=B0′(Complement B0). Thus the operation would be A+(B0′).

- Now 2's complement subtraction for two numbers A and B is given by A+B'. This suggests that when K=1, the operation being performed on the four bit numbers is subtraction.

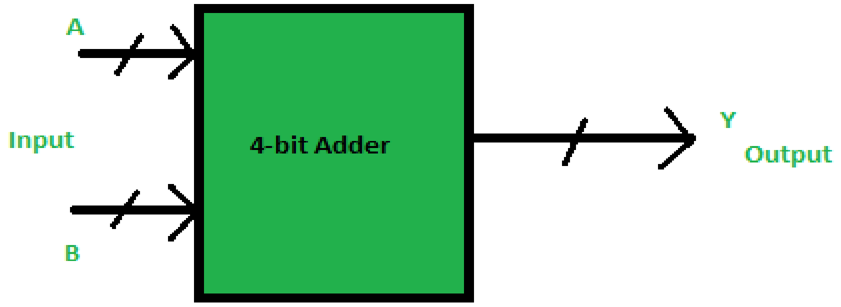
- Similarly If the Value of K=0, B0 (exor) K=B0. The operation is A+B which is simple binary addition. This suggests that When K=0, the operation being performed on the four bit numbers is addition.

Parallel Adder/ Subtractor

- Then C0 is serially passed to the second full adder as one of it’s outputs.
- The sum/difference S0 is recorded as the least significant bit of the sum/difference. A1, A2, A3 are direct inputs to the second, third and fourth full adders.
- Then the third input is the B1, B2, B3 EXORed with K to the second, third and fourth full adder respectively.
- The carry C1, C2 are serially passed to the successive full adder as one of the inputs. C3 becomes the total carry to the sum/difference. S1, S2, S3 are recorded to form the result with S0.
- For an n-bit binary adder-subtractor, we use n number of full adders.

BCD Adder

- **BCD adder** A 4-bit binary adder that is capable of adding two 4-bit words having a BCD (binary-coded decimal) format.
- The result of the addition is a BCD-format 4-bit output word, representing the decimal sum of the addend and augend, and a carry that is generated if this sum exceeds a decimal value of 9.



BCD Adder

- The output will varies from 0 to 18, if we are not considering the carry from the previous sum.
- But if we are considering the carry, then the maximum value of output will be 19 (i.e. 9+9+1 = 19).
- When we are simply adding A and B, then we get the binary sum. Here, to get the output in BCD form, we will use BCD Adder.
- If the sum of two number is less than or equal to 9, then the value of BCD sum and binary sum will be same otherwise they will differ by 6(0110). Now let’s move to the table and find out the logic when we are going to add 6(0110).

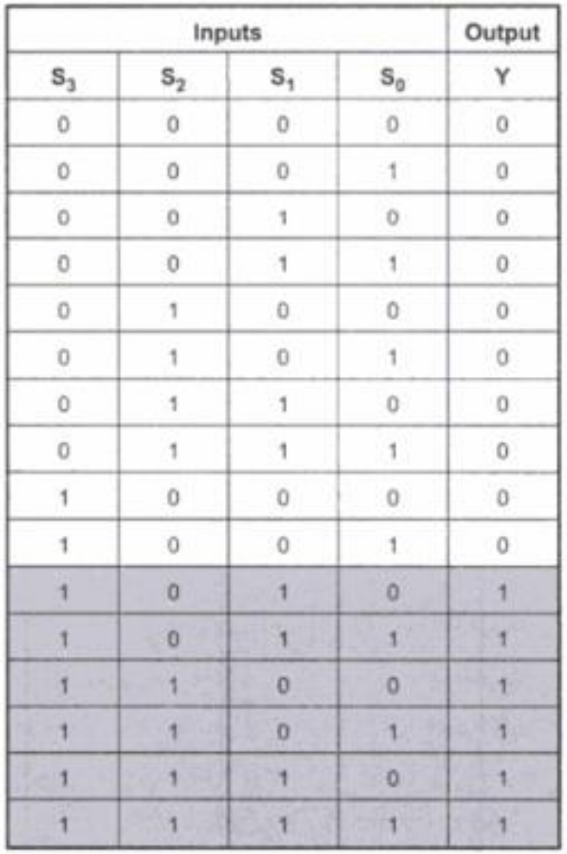
BCD Adder

- The digital systems handle the decimal number in the form of binary coded decimal numbers (BCD).
- A BCD adder is a circuit that adds two BCD digits and produces a sum digit also in BCD.
- To implement BCD adder we require:

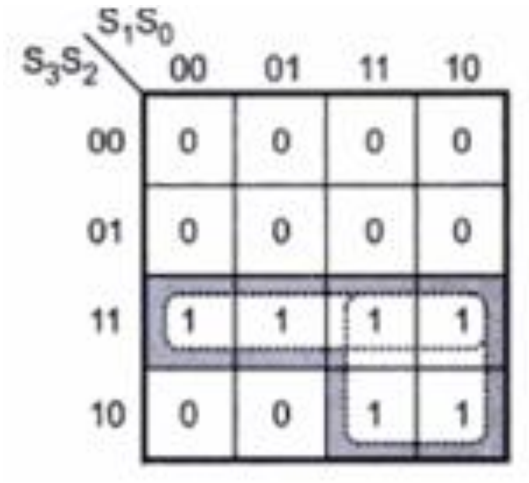
  1. 4-bit binary adder for initial addition
  2. Logic circuit to detect sum greater than 9
  3. One more 4-bit adder to add 0110201102 in the sum if sum is greater than 9 or carry is 1
  4. The logic circuit to detect sum greater than 9 can be determined by simplifying the Boolean expression of given truth Table.

```
          0101
        + 0110
          ----
          1011  → Invalid BCD number
        + 0110  → Add 6
     ---- ----
     0001 0001  → Valid BCD number
```

BCD Adder



Inputs				Output
S_3	S_2	S_1	S_0	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1



S_3S_2 \ S_1S_0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	0	0	1	1

$Y = S_3S_2 + S_3S_1$

- Y=1 indicates sum is greater than 9. We can put one more term, C_out in the above expression to check whether carry is one. So $\mathbf{Y = S_3\ S_2 + S_3\ S_1 + C_{out}}$
- If any one condition is satisfied we add 6(0110) in the sum.

BCD Adder

Circuit Diagram of BCD Adder:

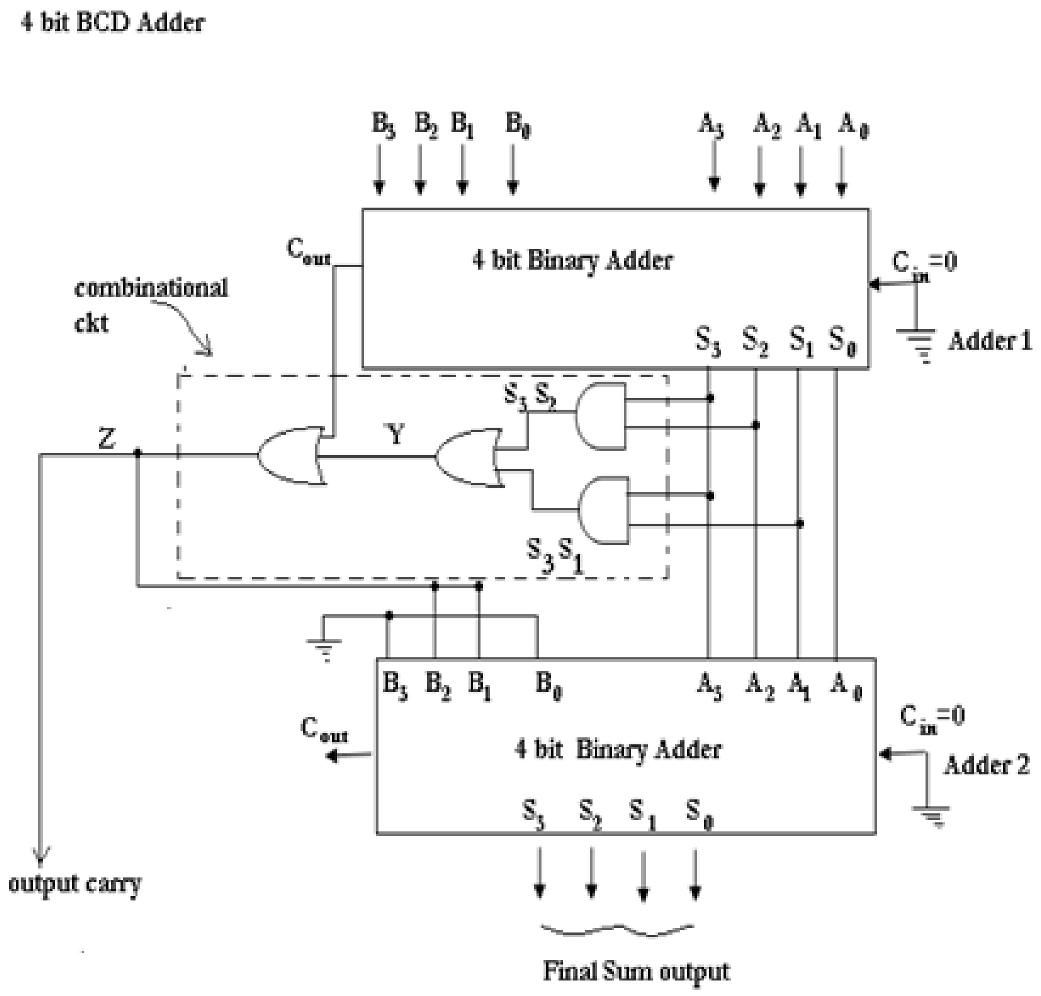


Fig5: 4-bit BCD adder

Daily Quiz

- How many full adder are needed to implement 4 bit parallel adder:

a) 2

b) 3

c) 4

d) 5

- What are the advantages of carry look ahead adder over parallel adder?
- What is the major difference between half-adders and full-adders?

a) Full-adders are made up of two half-adders
b) Full adders can handle double-digit numbers
c) Full adders have a carry input capability
d) Half adders can handle only single-digit numbers

Old Questions

- Design and explain carry look ahead adder.
- What is difference between “Ripple Carry Adder” and Carry Look-ahead Geneartor?
- Draw a decimal adder to add BCD numbers.
- Draw a BCD adder and explain its working.

Faculty Video Links, Youtube & NPTEL Video Links and Online Courses Details

- Youtube/other Video Links
  - https://www.youtube.com/watch?v=aLUY-s7LSns&ab_channel=NesoAcademyNesoAcademy
  - https://www.youtube.com/watch?v=36hCizOk4PA&ab_channel=nptelhrd
  - https://www.youtube.com/watch?v=CwIk8lyrMI8&ab_channel=nptelhrd

Weekly Assignment

- Implement full adder using two half adder.
- Implement Full adder using NAND gates only.
- Design and explain carry look ahead adder.
- Implement Full subtractor using NOR gates only.
- Design a Combinational circuit with mode control that can perform 4 bit addition and subtraction.
- Draw a BCD adder and explain its working.

Recap

- Parallel adder is used to do arithmetic operation of two or more bits.
- Carry look ahead adder is used to minimize the drawbacks of parallel adder.
- **BCD adder** A 4-bit binary adder that is capable of adding two 4-bit words having a BCD (binary-coded decimal) format.
- The result of the addition is a BCD-format 4-bit output word, representing the decimal sum of the addend and augend, and a carry that is generated if this sum exceeds a decimal value of 9.

Multiplexers

Topic objective	Mapping with CO
To understand the working of MUX.	CO2
To understand the circuit diagram of MUX.	CO2
To implement the circuits and logic gates using MUX	CO2
To implement the boolean function using MUX.	CO2

Multiplexers

Prerequisite:

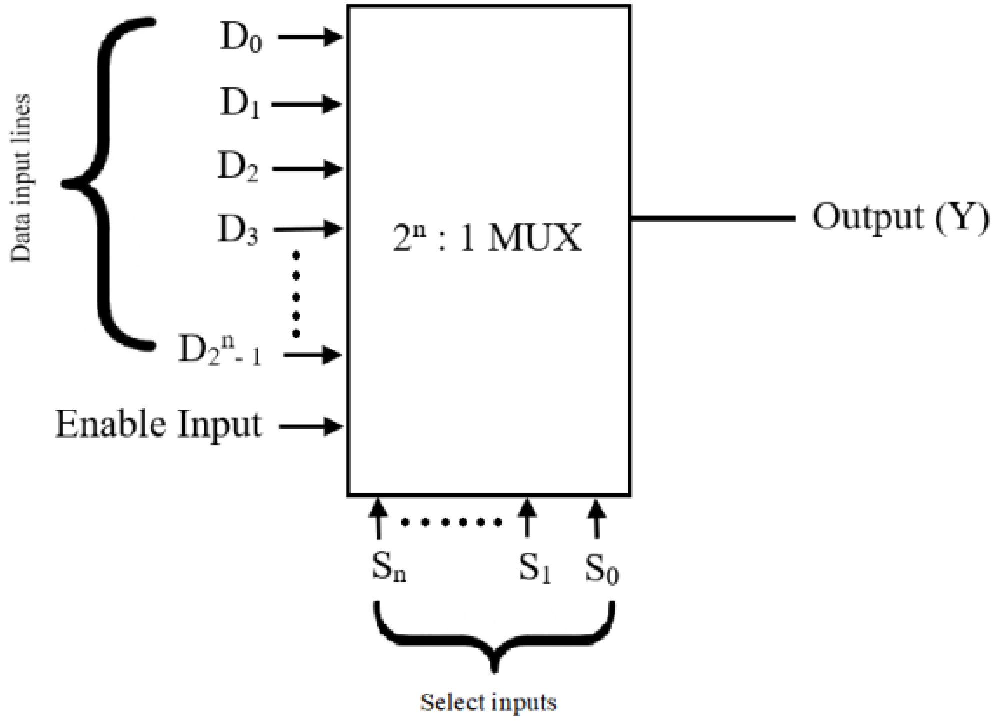
- Basic knowledge of logic gates.
- Basic knowledge of Boolean algebra.

Multiplexer

- Multiplexing means transmitting a large number of information units over a smaller number of channels or lines. A digital multiplexer is CLC that selects binary information from one of many input lines and directs it to a single output line. The selection of a particular input line is controlled by selection lines.
- Multiplexer is a special type of combinational circuit. There are n-data inputs, one output and m select inputs with $2^m = n$.
- It is a digital circuit which selects one of the n data inputs and routes it to the output. The selection of one of the n inputs is done by the selected inputs.

Multiplexer

- Depending on the digital code applied at the selected inputs, one out of n data sources is selected and transmitted to the single output Y. E is called the strobe or enable input which is useful for the cascading. It is generally an active low terminal, that means it will perform the



Multiplexer

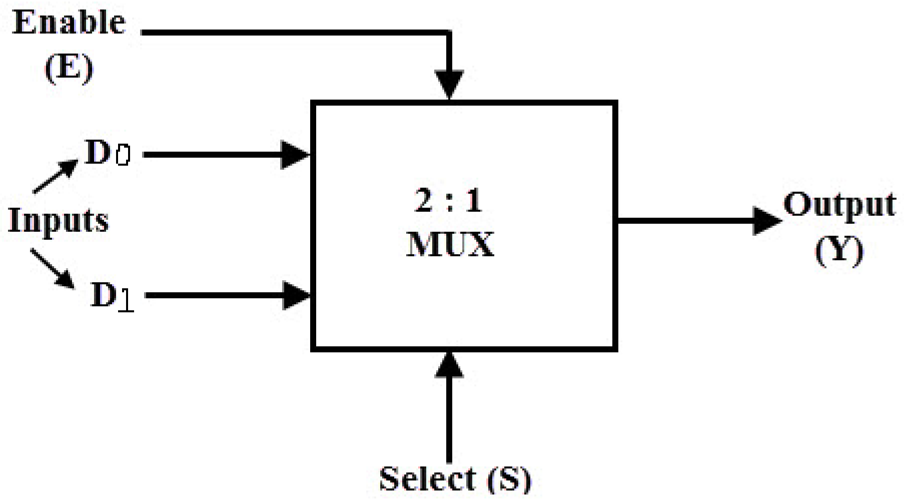
Applications of Multiplexers: Multiplexers are used in various applications wherein multiple-data need to be transmitted by using single line.

- **Communication System**: By using a multiplexer, the efficiency of the communication system can be increased by allowing the transmission of data.
- **Computer Memory**: Multiplexers are used in computer memory to maintain a huge amount of memory in the computers.
- **Telephone Network**: In telephone networks, multiple audio signals are integrated on a single line of transmission with the help of a multiplexer.
- **Transmission from the Computer System of a Satellite**: Multiplexer is used to transmit the data signals from the computer system of a spacecraft or a satellite to the ground system by using a GSM satellite.

Multiplexer

2 *1 MUX:

- A 2-to-1 multiplexer consists of two inputs D0 and D1, one select input S and one output Y. Depends on the select signal, the output is connected to either of the inputs.
- If the select line is low, then the output will be switched to D0 input, whereas if select line is high, then the output will be switched to D1 input.

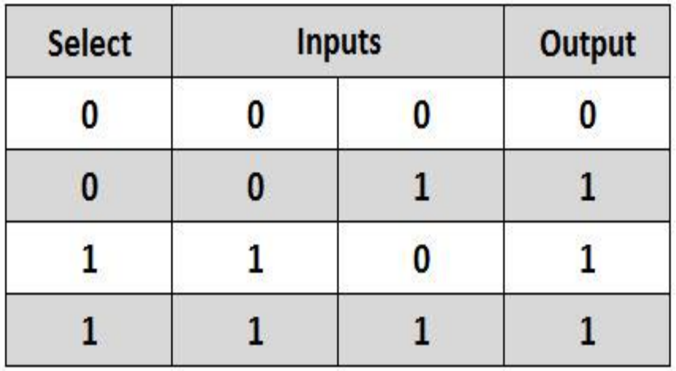


Multiplexer

- From the truth table the Boolean expression of the output is given as:

$$Y = D_0\,\bar{S} + D_1\,S$$

- **Truth Table :**

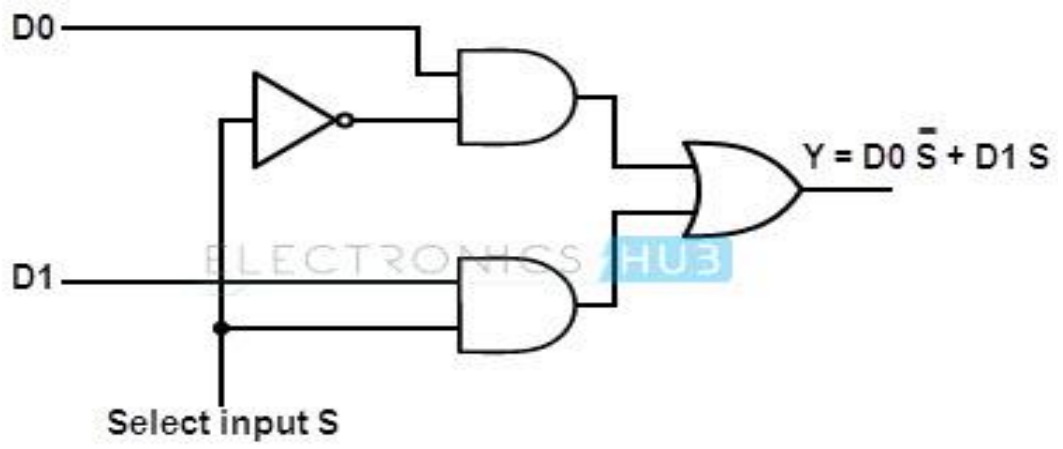


Select	Inputs		Output
0	0	0	0
0	0	1	1
1	1	0	1
1	1	1	1

- It consists of two AND gates, one NOT gate and one OR gate. When the select line, S=0, the output of the upper AND gate is zero, but the lower AND gate is D0.

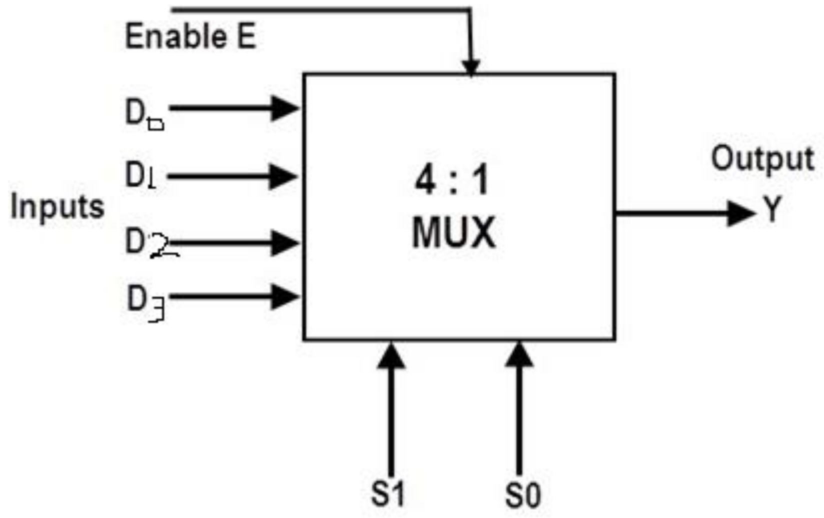
Multiplexer

- Thus, the output generated by the OR gate is equal to D0. Similarly, when S=1, the output of the lower AND gate is zero, but the output of upper AND gate is D1. Therefore, the output of the OR gate is D1. Thus, the above given Boolean expression is satisfied by this circuit.



Multiplexer

- **4*1 MUX :**A 4-to-1 multiplexer consists four data input lines as D0 to D3, two select lines as S0 and S1 and a single output line Y. The select lines S1 and S2 select one of the four input lines to connect the output line. The particular input combination on select lines selects one of input (D0 through D3). output.

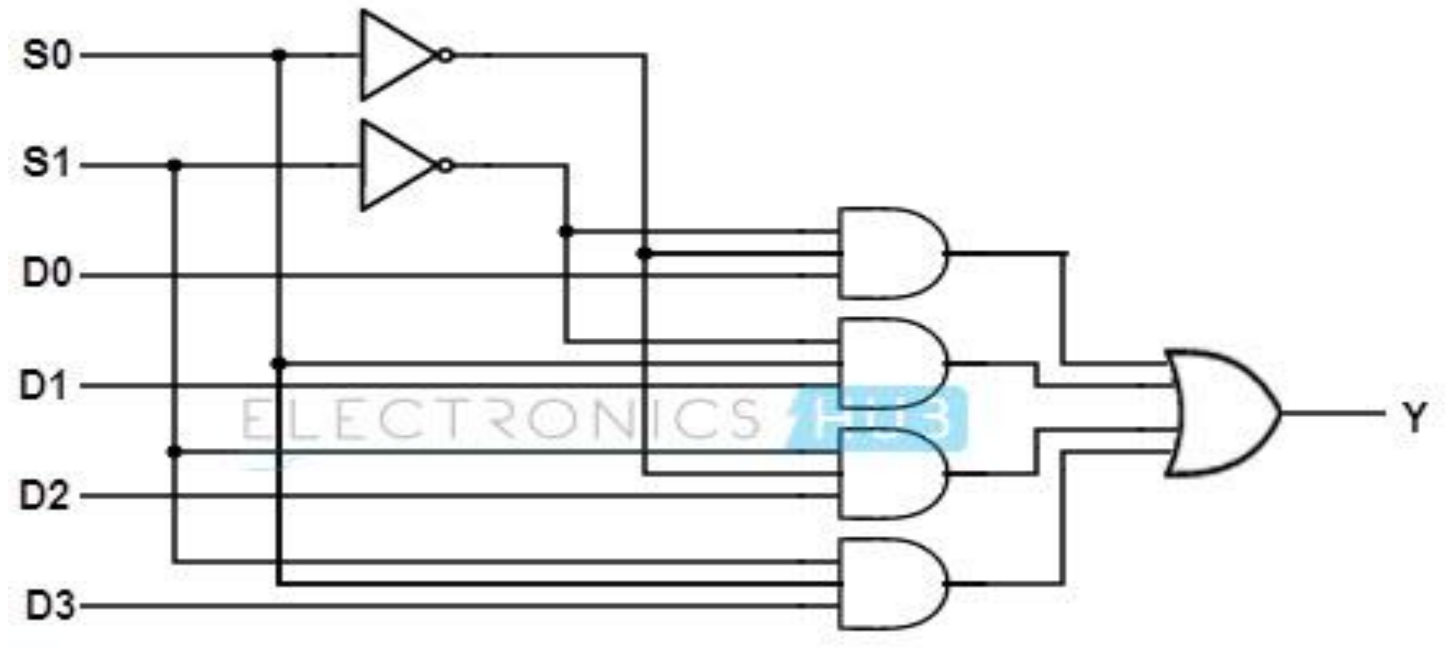


Select Data Inputs		Output
S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

Multiplexer

- Boolean expression of this multiplexer is given as:

$$Y = D0\,\overline{S1}\,\overline{S0} + D1\,\overline{S1}\,S0 + D2\,S1\,\overline{S0} + D3\,S1\,S0$$

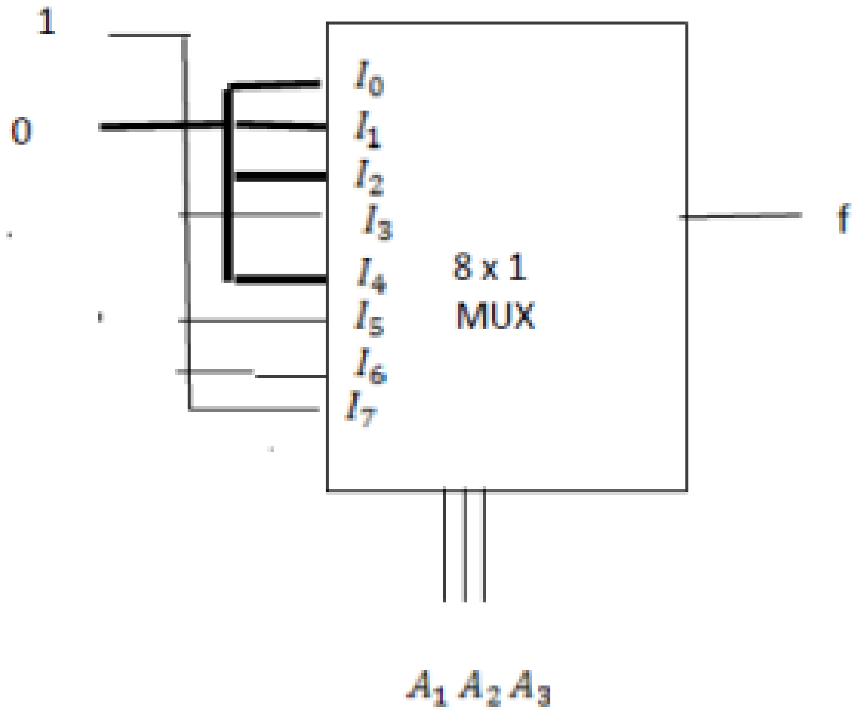


Multiplexer

Eg: Implement F (A1, A2, A3) = ∑ (3, 5, 6, 7) Using

(a) 8x1 multiplexer (b) 4x1 multiplexer

(a) Using 8x1 multiplexer: A1, A2, A3 will be select lines.

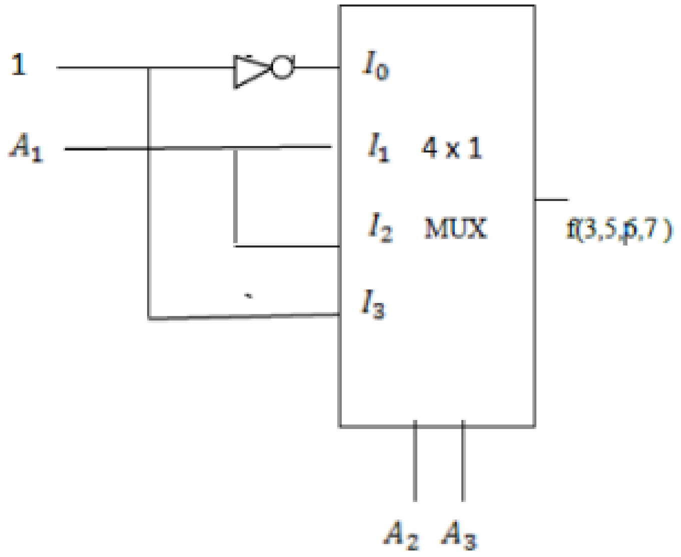


Multiplexer

(b) Implement F (A1, A2, A3) = ∑ (3, 5, 6, 7) using 4x1 multiplexer:
If MSB i.e. A1 is used as input variable and A2, A3 as select lines.

Minterms	A_1	A_2	A_3	f
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

	I_0	I_1	I_2	I_3
$\overline{A_1}$	0	1	2	$\boxed{3}$
A_1	4	$\boxed{5}$	$\boxed{6}$	$\boxed{7}$
	0	A_1	A_1	1

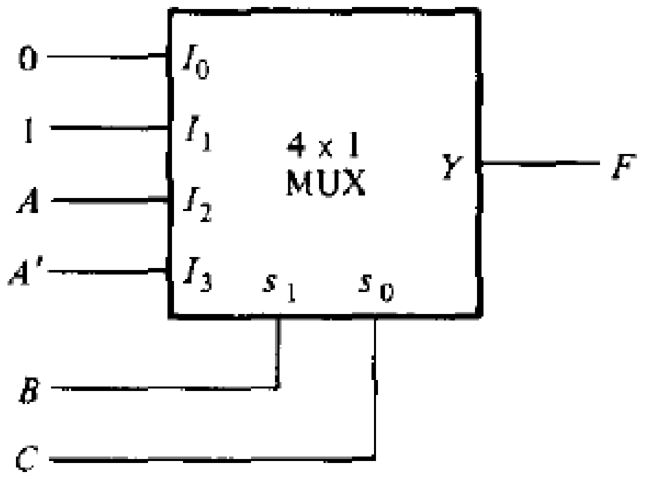


Multiplexer

Example: Implement F (A, B, C) = ∑ (1, 3, 5, 6) using 4x1 multiplexer: If MSB i.e. A is used as input variable and B, C as select lines.

Minterm	A	B	C	F
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

	I_0	I_1	I_2	I_3
A'	0	①	2	③
A	4	⑤	⑥	7
	0	1	A	A'



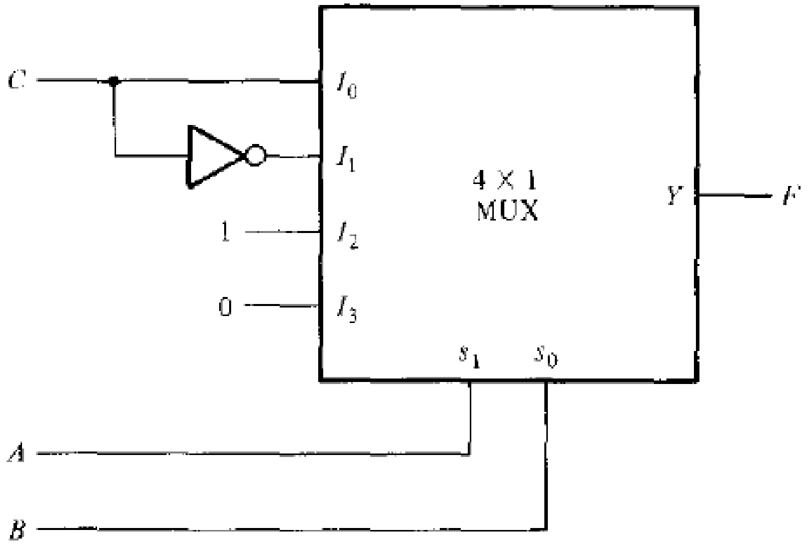
Multiplexer

Eg: Implement F (A, B, C) = ∑ (1, 2, 4, 5) using 4x1 multiplexer:

If LSB i.e. C is used as input variable and A, B as select lines.

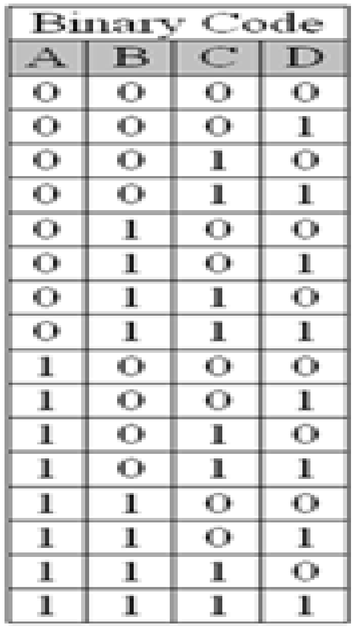
A	B	C	F	
0	0	0	0	$F = C$
0	0	1	1	
0	1	0	1	$F = C'$
0	1	1	0	
1	0	0	1	$F = 1$
1	0	1	1	
1	1	0	0	$F = 0$
1	1	1	0	

	I_0	I_1	I_2	I_3
C'	0	②	④	6
C	①	3	⑤	7
	C	C'	1	0

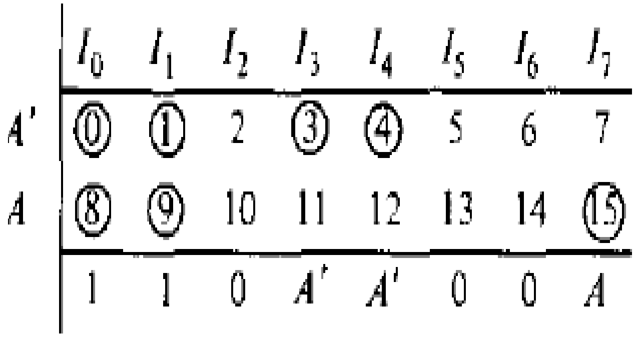


Multiplexer

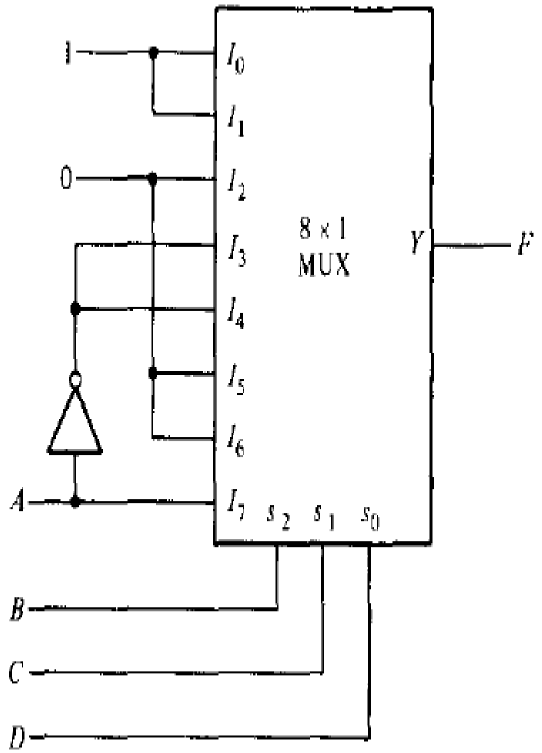
Example: Implement F (A, B, C, D) = ∑ (0,1,3,4,8,9,15) using 8x1 multiplexer: If MSB i.e. A is used as input variable and B, C, D as select lines.



Binary Code			
A	B	C	D
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1



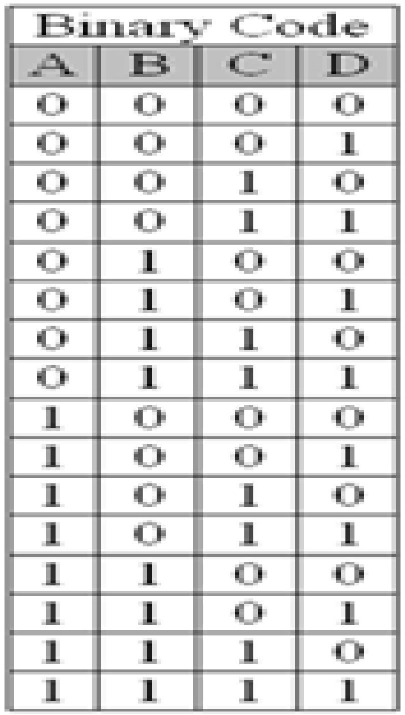
	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
A'	⓪	①	2	③	④	5	6	7
A	⑧	⑨	10	11	12	13	14	⑮
	1	1	0	A'	A'	0	0	A



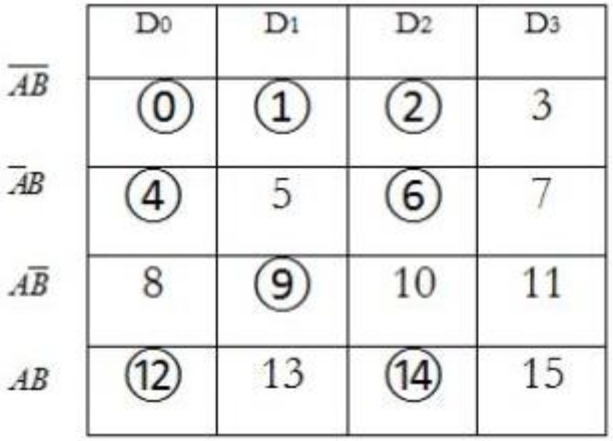
Multiplexer

Example: Implement the function F(A,B,C,D)=∑m(0,1,2,4,6,9,12,14) Using 4x1 multiplexer

Sol: A, B are used as input variables and C, D as select lines.



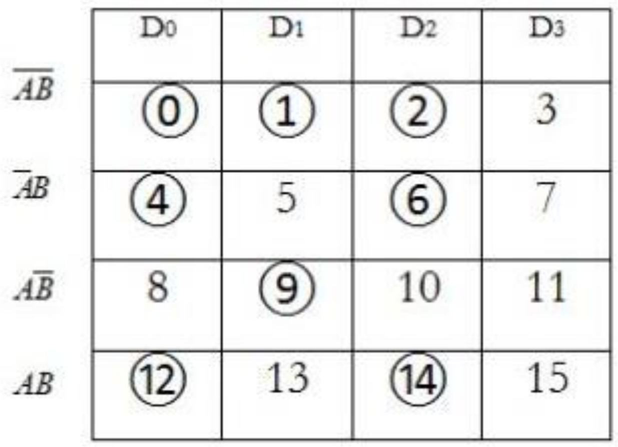
Binary Code			
A	B	C	D
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1



	D_0	D_1	D_2	D_3
$\overline{AB}$	⓪	①	②	3
$\overline{A}B$	④	5	⑥	7
$A\overline{B}$	8	⑨	10	11
AB	⑫	13	⑭	15

Multiplexer

Input Variables from the given table can be written as:



	D_0	D_1	D_2	D_3
$\overline{A}\,\overline{B}$	⓪	①	②	3
$\overline{A}B$	④	5	⑥	7
$A\overline{B}$	8	⑨	10	11
AB	⑫	13	⑭	15

D_0 = A'B' + A'B + AB
= A'B' +A'B + A'B + AB
= A' (B' + B) + B (A' + A)   as (A' + A) = 0
= A' + B

D_1 = A'B' + AB'
= B' (A' + A)
= B'

D_2 = A'B' + A'B + AB
= A'B' +A'B + A'B + AB
= A' (B' + B) + B (A' + A)   as (A' + A) = 0
= A' + B

D_3 = 0

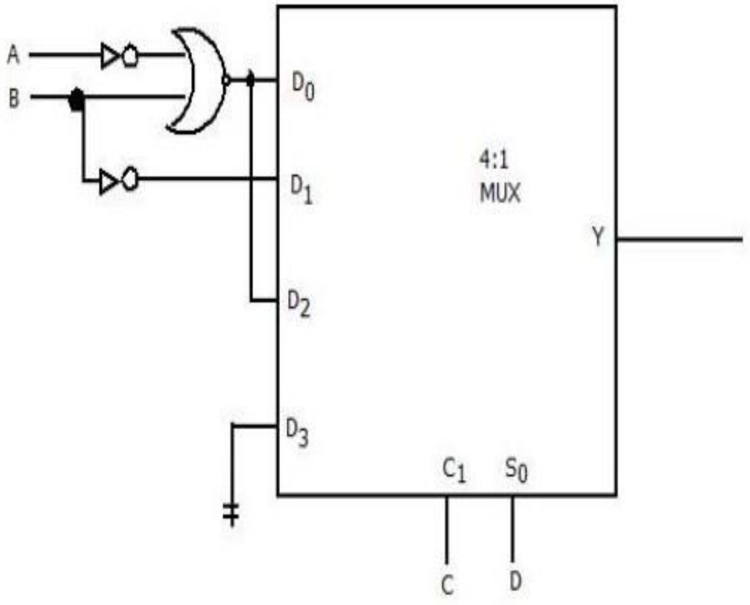
Multiplexer

D_0 = A’B’ + A’B + AB
= A’B’ +A’B + A’B + AB
= A’ (B’ + B) + B (A’ + A) as (A’ + A) = 0
= A’ + B

D_1 = A’B’ + AB’
= B’ (A’ + A)
= B’

D_2 = A’B’ + A’B + AB
= A’B’ +A’B + A’B + AB
= A’ (B’ + B) + B (A’ + A) as (A’ + A) = 0
= A’ + B

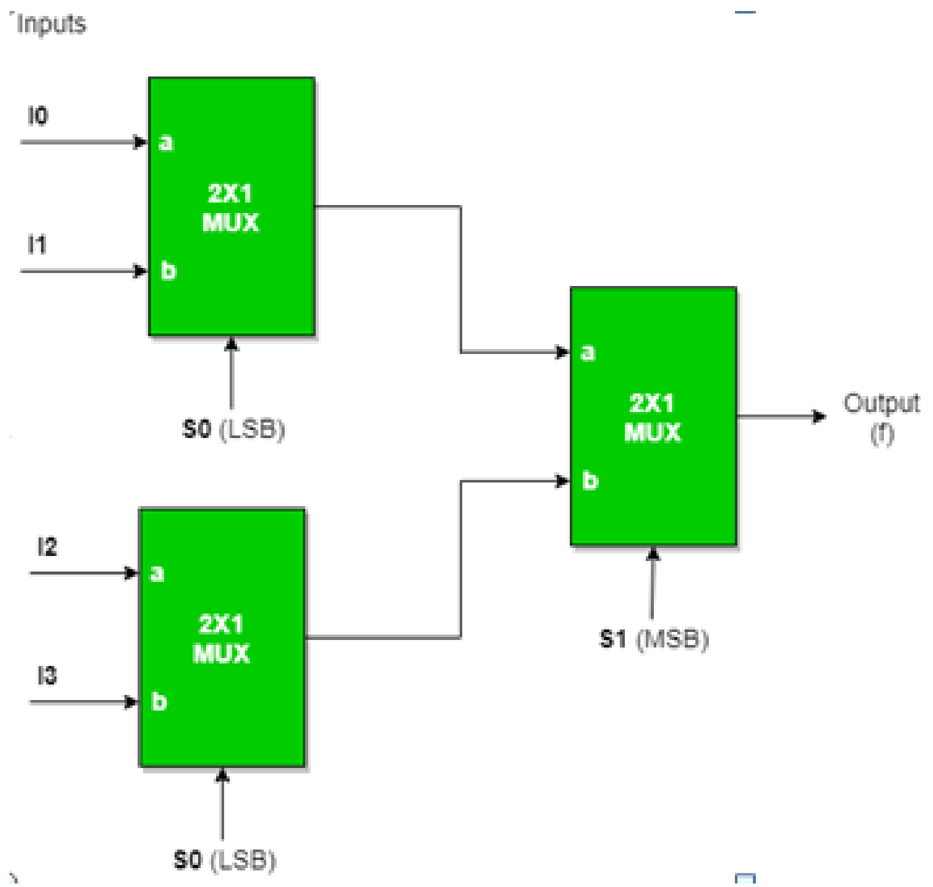
D_3 = 0



Multiplexer Implementation

Multiplexer

- Implement 4*1 MUX using 2*1 Mux.

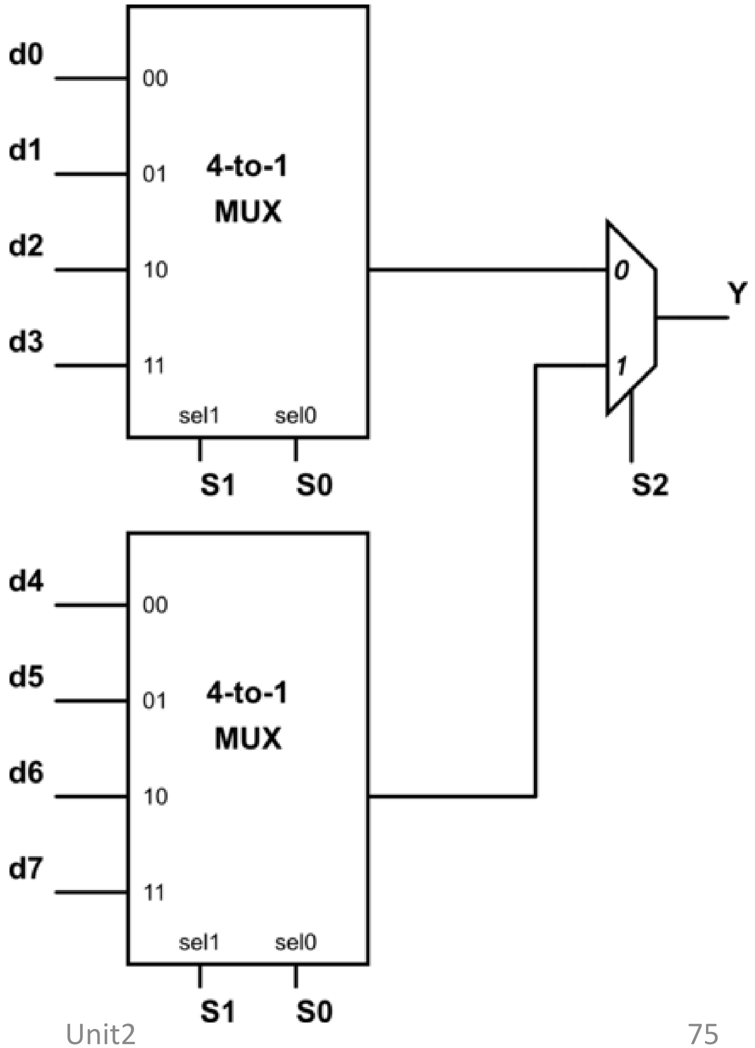


Selection Lines		Output
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Multiplexer

- Implement 8*1 MUX using 4*1 Mux

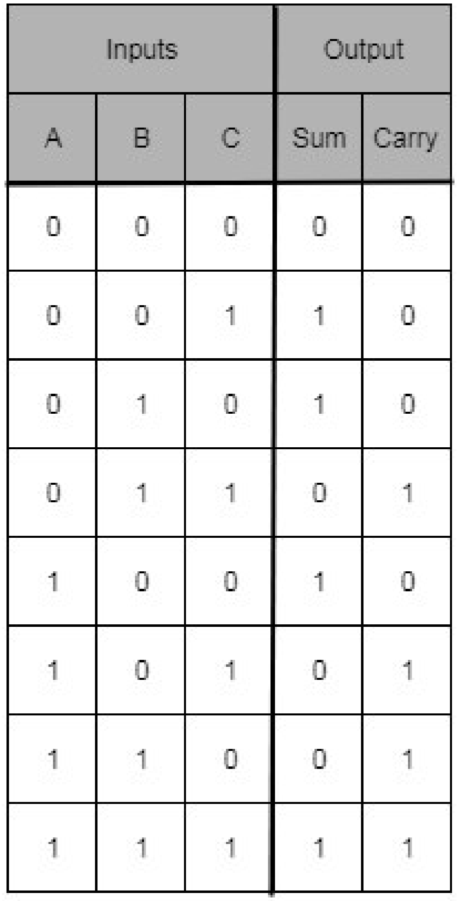
Select Data Inputs			Output
S_2	S_1	S_0	Y
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7



Multiplexer

- **Implement a full adder circuit using two 4:1 multiplexers.**

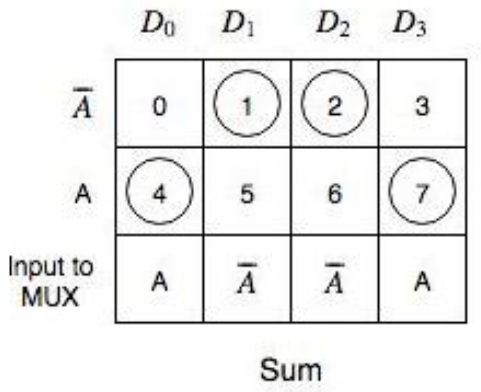
Step 1: Truth table



Inputs			Output	
A	B	C	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

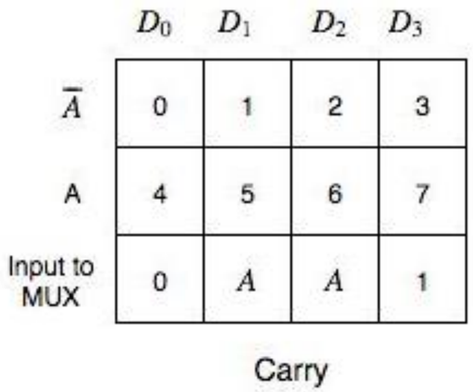
Step 2: Tables for sum and carry outputs

(A as input line and B,C as select



	D_0	D_1	D_2	D_3
$\bar{A}$	0	$\textcircled{1}$	$\textcircled{2}$	3
A	$\textcircled{4}$	5	6	$\textcircled{7}$
Input to MUX	A	$\bar{A}$	$\bar{A}$	A

Sum

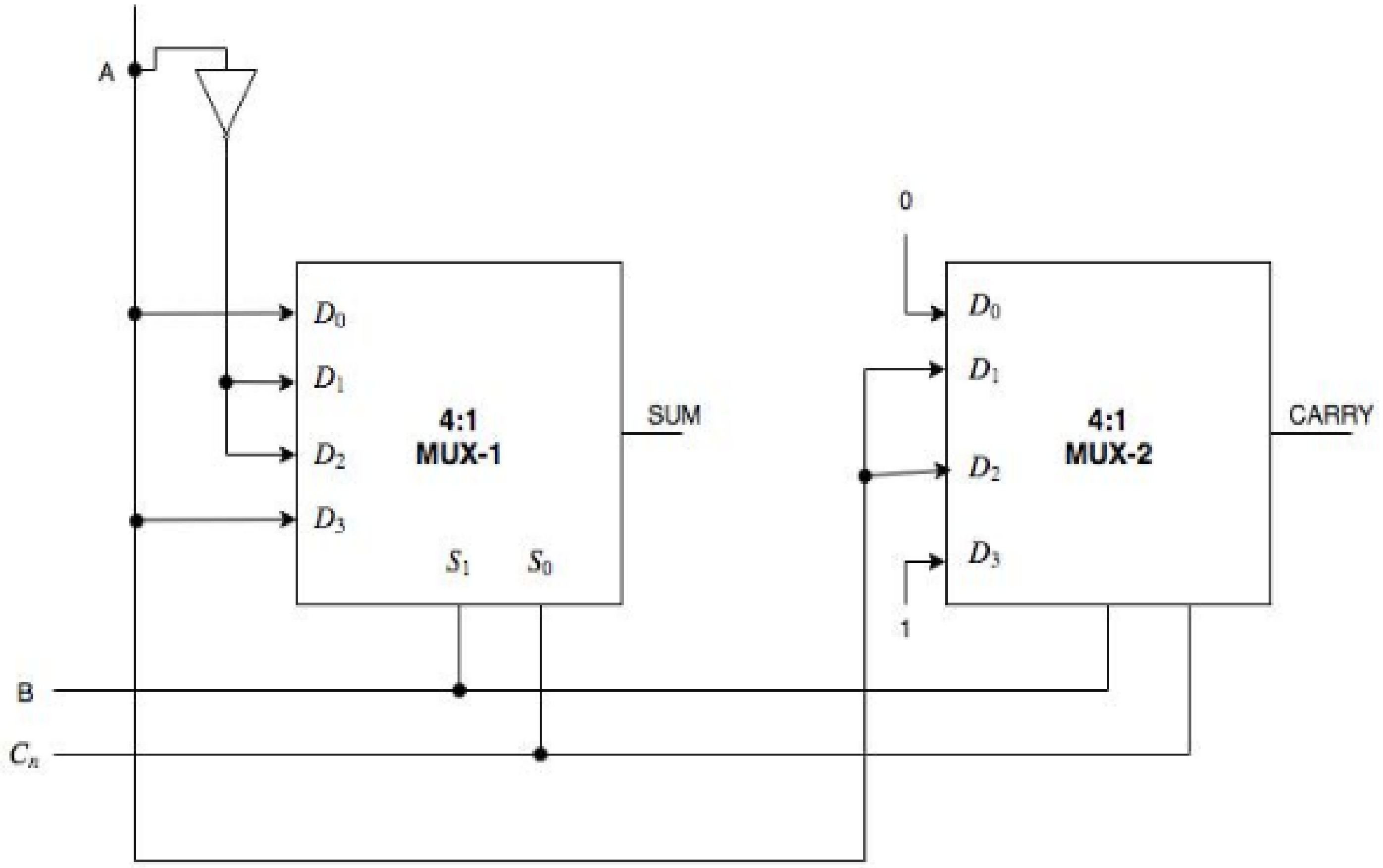


	D_0	D_1	D_2	D_3
$\bar{A}$	0	1	2	3
A	4	5	6	7
Input to MUX	0	A	A	1

Carry

Multiplexer

- Step 3: The full adder using 4:1 multiplexer



Faculty Video Links, Youtube & NPTEL Video Links and Online Courses Details

- Youtube/other Video Links
  - ➢ https://www.youtube.com/watch?v=3aWLCH9_EPA&ab_channel=IITKharagpurJuly2018IITKharagpurJuly2018
  - ➢ https://nptel.ac.in/courses/117/106/117106086/

Daily Quiz

- What is a multiplexer?

  a) It is a type of decoder which decodes several inputs and gives one output

  b) A multiplexer is a device which converts many signals into one

  c) It takes one input and results into many output

  d) It is a type of encoder which decodes several inputs and gives one output

- How many 2:1 multiplexers are required to generate output of a 2 input NAND gate?

a) 1 b) 2 c) 3 d) 4

- Which combinational circuit is renowned for selecting a single input from multiple inputs & directing the binary information to output line?

  a) Data Selector

  b) Data distributor

  c) Both data selector and data distributor

  d) DeMultiplexer

Old Questions

- Combinational circuit is defined by F (A,B,C)= ∑(3,5,6,7).Implement the circuit using 4:1 MUX
- Design 32:1 MUX using 2:1 MUX.
- Implement F (A, B, C, D) =∑(0,2,4,8,9,12) using 4x1 multiplexer.
- Implement full adder using 8:1 mux.
- Implement a 4:1 multiplexer using 2:1 multiplexer.

Recap

- Multiplexing means transmitting a large number of information units over a smaller number of channels or lines.
- A digital multiplexer is CLC that selects binary information from one of many input lines and directs it to a single output line.
- The selection of a particular input line is controlled by selection lines.

Demultiplexers

Topic objective	Mapping with CO
To understand the working and circuit diagram of DEMUX.	CO2

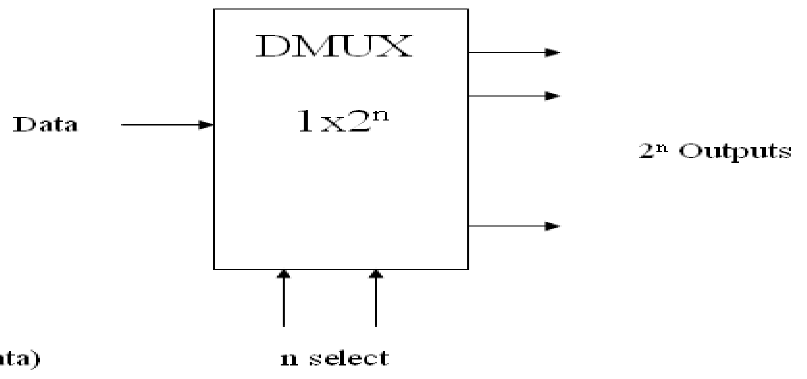
Demultiplexers

Prerequisite:

- Basic knowledge of logic gates.
- Basic knowledge of Boolean algebra.

Demultiplexer

- A demultiplexer (or demux) is a device that takes a single input line and routes it to one of several digital output lines. A demultiplexer of 2^n outputs has n select lines, which are used to select which output line to send the input. A demultiplexer is also called a data distributor.



No. of inputs =1 (Data)

No. of outputs =<2^n

No. of select= n

Demultiplexer block diagram

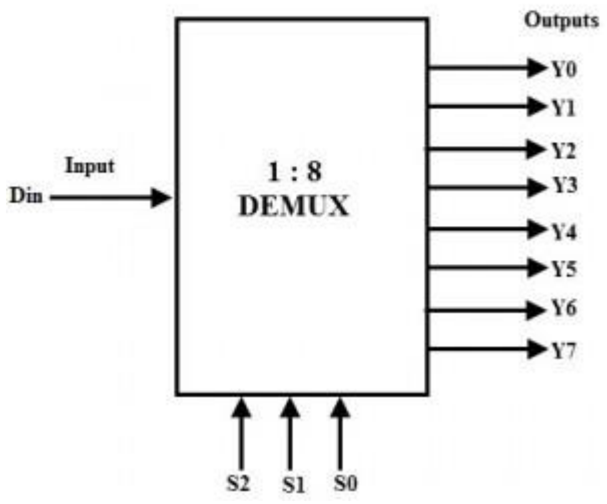
Demultiplexer

Applications of Demultiplexer: Demultiplexer is used to connect a single source to multiple destinations.

- **Communication System** – Communication system use multiplexer to carry multiple data like audio, video and other form of data using a single line for transmission.
- **ALU (Arithmetic Logic Unit)** – In an ALU circuit, the output of ALU can be stored in multiple registers or storage units with the help of demultiplexer.
- **Serial to parallel converter** – A serial to parallel converter is used for reconstructing parallel data from incoming serial data stream.

Demultiplexer

- The below figure shows the block diagram of a 1-to-8 demultiplexer that consists of single input D, three select inputs S2, S1 and S0 and eight outputs from Y0 to Y7.
- It is also called as 3-to-8 demultiplexer due to three select input lines. It distributes one input line to one of 8 output lines depending on the combination of select inputs.



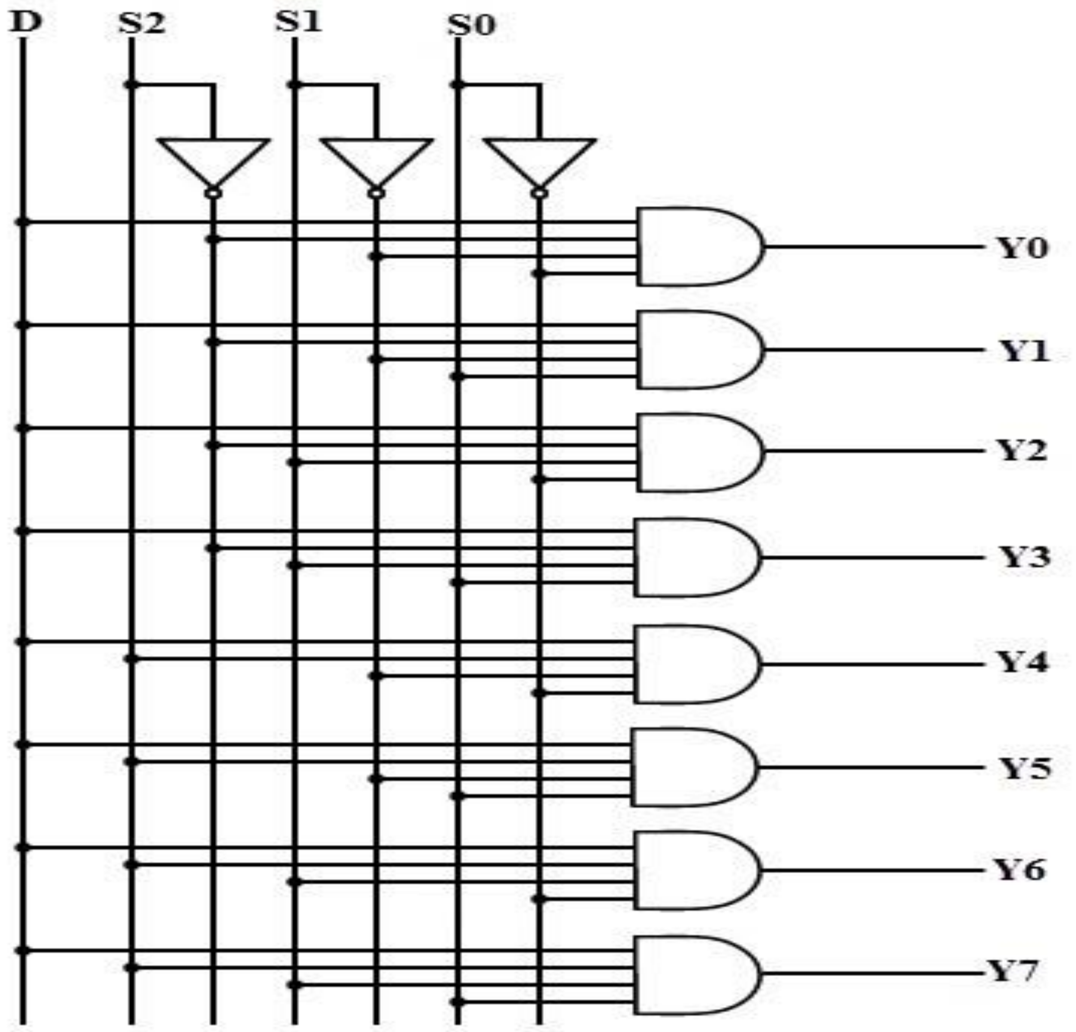
Demultiplexer

Truth Table :

Data Input	Select Inputs			Outputs							
D	S_2	S_1	S_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
D	0	0	0	0	0	0	0	0	0	0	D
D	0	0	1	0	0	0	0	0	0	D	0
D	0	1	0	0	0	0	0	0	D	0	0
D	0	1	1	0	0	0	0	D	0	0	0
D	1	0	0	0	0	0	D	0	0	0	0
D	1	0	1	0	0	D	0	0	0	0	0
D	1	1	0	0	D	0	0	0	0	0	0
D	1	1	1	D	0	0	0	0	0	0	0

Demultiplexer

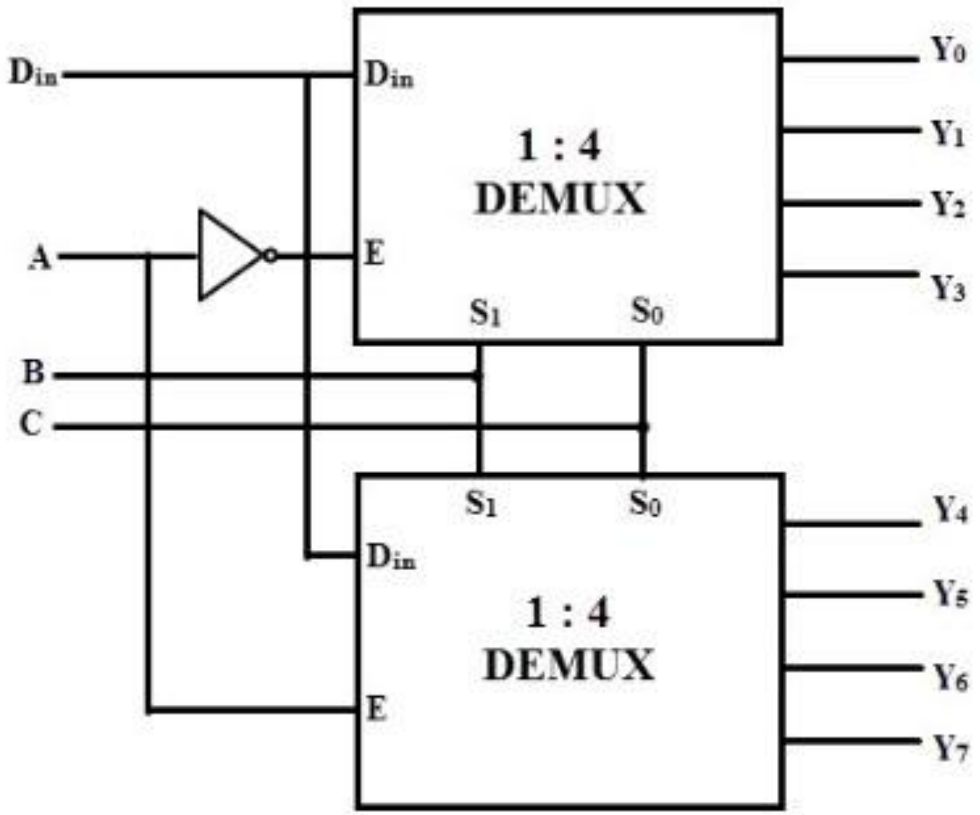
- Logic Diagram :



De-multiplexer Tree

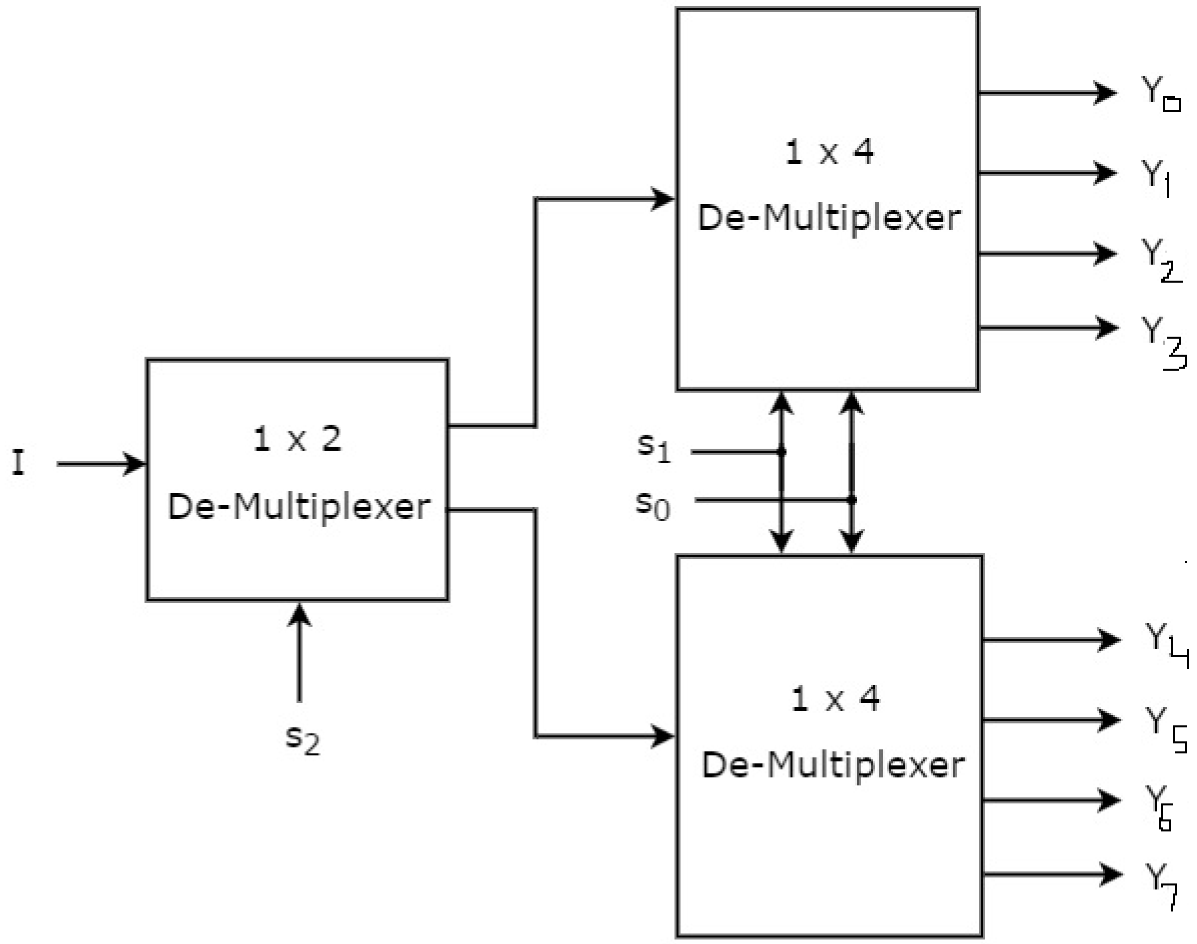
- Design 1x8 de-mux using 1x4 de-mux.

  Sol: The input is common to both demux and select line A is acting as enable signal and B and C are select lines for 1x4 de-mux.



De-multiplexer Tree

- Design 1x8 de-mux using lower order demux.



NIET Greater Noida GET FUTURE READY

Daily Quiz

- In a 1:16 demultiplexer, the number of control inputs will be:

a) 3 c) 4

b) 5 d) 16

- Implement 1:8 de-multiplexer using 1:2 de-multiplexer.
- The word demultiplex means ____________
  a) One into many
  b) Many into one
  c) Distributor
  d) One into many as well as Distributor
- In 1-to-4 multiplexer, if C1 = 1 & C2 = 1, then the output will be ____________
  a) Y0
  b) Y1
  c) Y2
  d) Y3

Old Questions

- Design 1:8 demux using 1:2 demux.
- Implement full adder using 1:8 demux.
- Write a short note on demultiplexer.

Faculty Video Links, Youtube & NPTEL Video Links and Online Courses Details

- Youtube/other Video Links
  - https://www.youtube.com/watch?v=7G1i5PUgz3w&ab_channel=IITKharagpurJuly2018
  - https://nptel.ac.in/courses/117/106/117106086/

Weekly Assignment

- Combinational circuit is defined by F (A,B,C)= ∑(3,5,6,7).Implement the circuit using 4:1 MUX
- Design 32:1 MUX using 2:1 MUX.
- Implement F (A, B, C, D) =∑(0,2,4,8,9,12) using 4x1 multiplexer.
- Implement full adder using 8:1 mux.
- Implement full subtractor using 1:8 demux.

Recap

- A demultiplexer of 2^n outputs has n select lines, which are used to select which output line to send the input.
- A demultiplexer is also called a data distributor.

Applications of Demultiplexer:

- Communication System
- ALU (Arithmetic Logic Unit)
- Serial to parallel converter

Decoders

Topic objective	Mapping with CO
To understand the working Decoder.	CO2
To understand the circuit of Decoder.	CO2

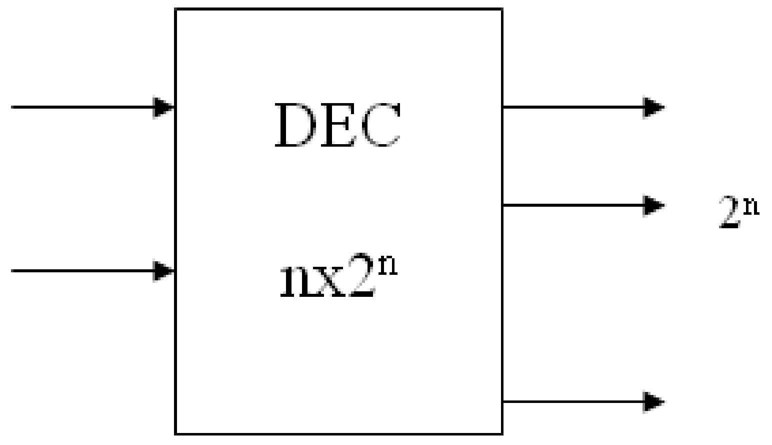
Decoders

Prerequisite:

- Basic knowledge of logic gates.
- Basic knowledge of Boolean algebra.

Decoder

- A *decoder is* a combinational circuit that converts binary information from *n* input lines to a maximum of 2^n unique output lines.

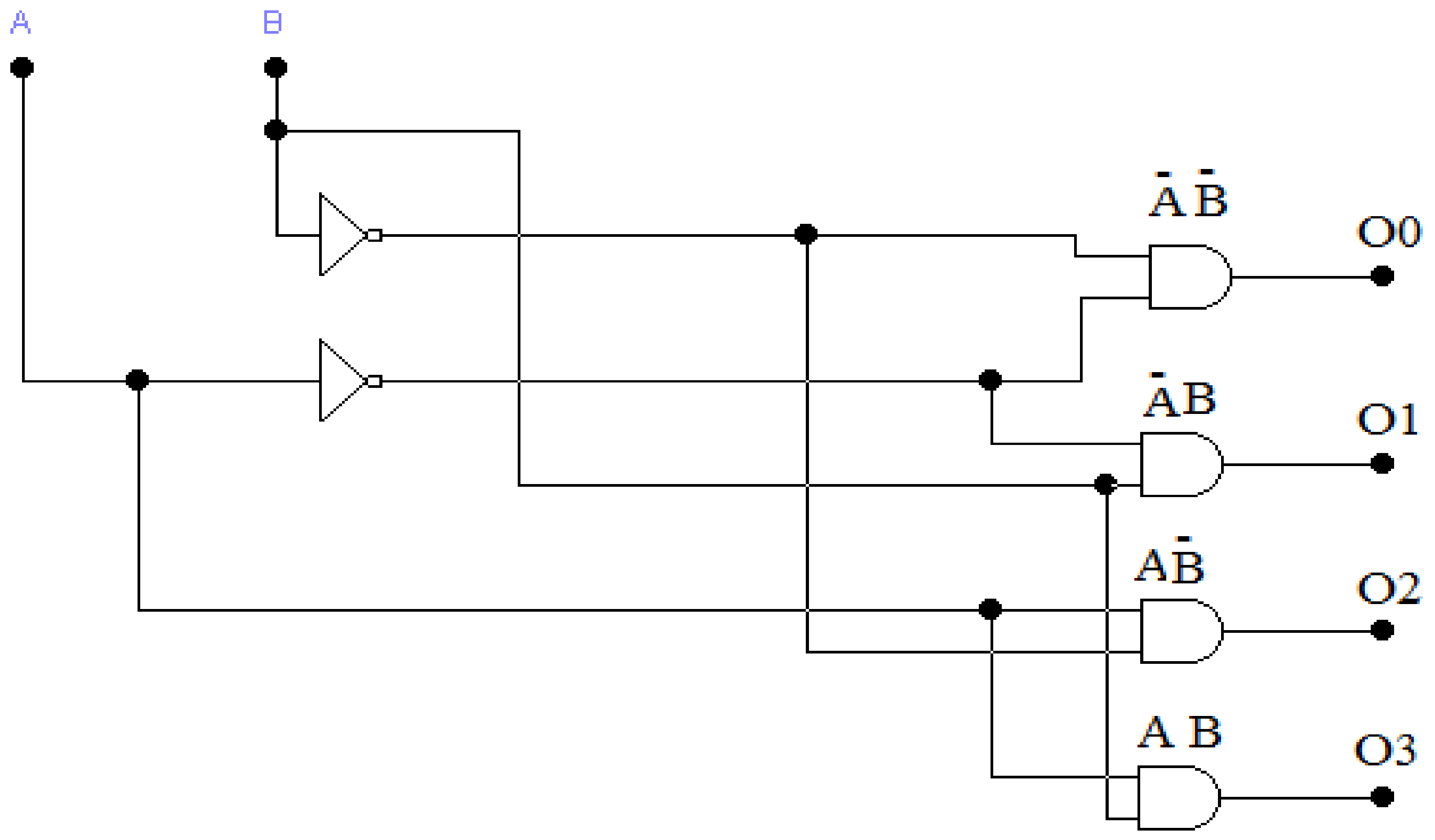


No. of inputs = n

No. of outputs =< 2^n

Decoder Block Diagram

Decoder



2x4 Decoder logical diagram

Decoder

Eg . Implement full adder using Decoder.

- For example, if we need to implement the logic of a full adder, we need a 3:8 decoder and OR gates. The input to the full adder, first and second bits and carry bit, are used as input to the decoder. Let x, y and z represent these three bits. Sum and Carry outputs of a full adder have the following truth tables-

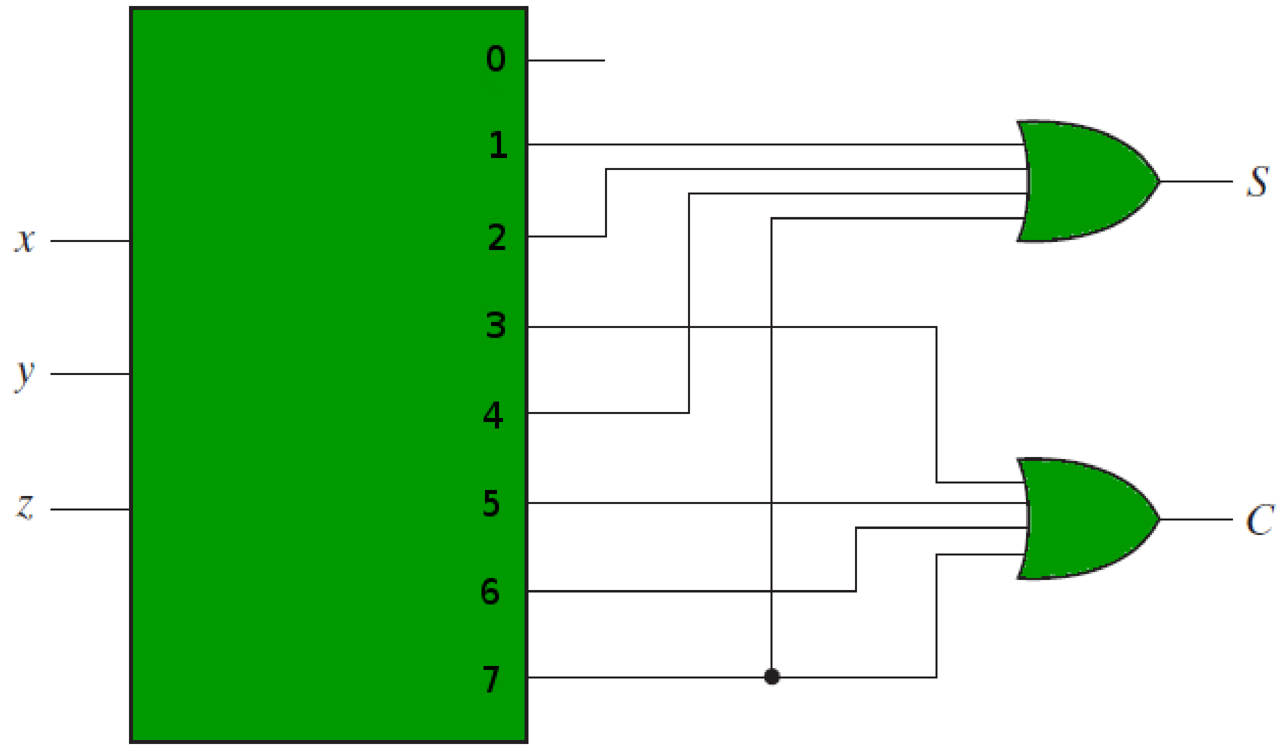
x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$S = \Sigma m(1,2,4,7)$$

$$C = \Sigma m(3,5,6,7)$$

Decoder

The following circuit diagram shows the implementation of Full adder using a 3:8 Decoder and OR gates.



Decoder

Implement the following two higher-order decoders using lower-order decoders 3*8 using 2*4 decoder .

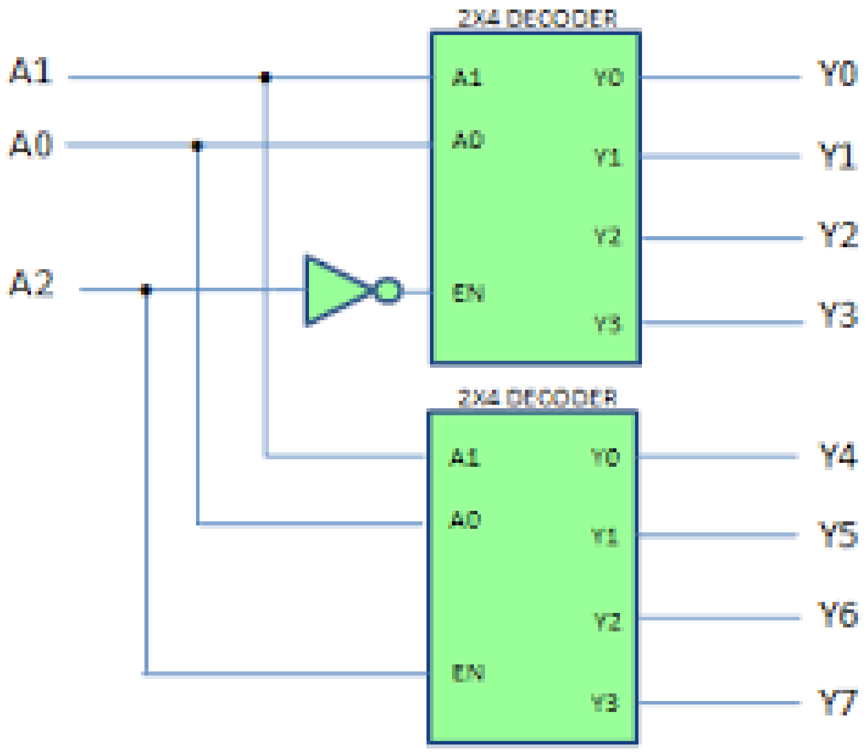
- – We can find the number of lower order decoders required for implementing higher order decoder using the following formula.
- – Required number of lower order decoders =m2 / m1
- – Where, m1 is the number of outputs of lower order decoder.
    - – m2 is the number of outputs of higher order decoder.

- Here, m1 = 4 and m2= 8. Substitute, these two values in the above formula.
- Required number of 2*4 decoders= 8/4 = 2

- Therefore, we require two 2 to 4 decoders for implementing one 3 to 8 decode

Decoder

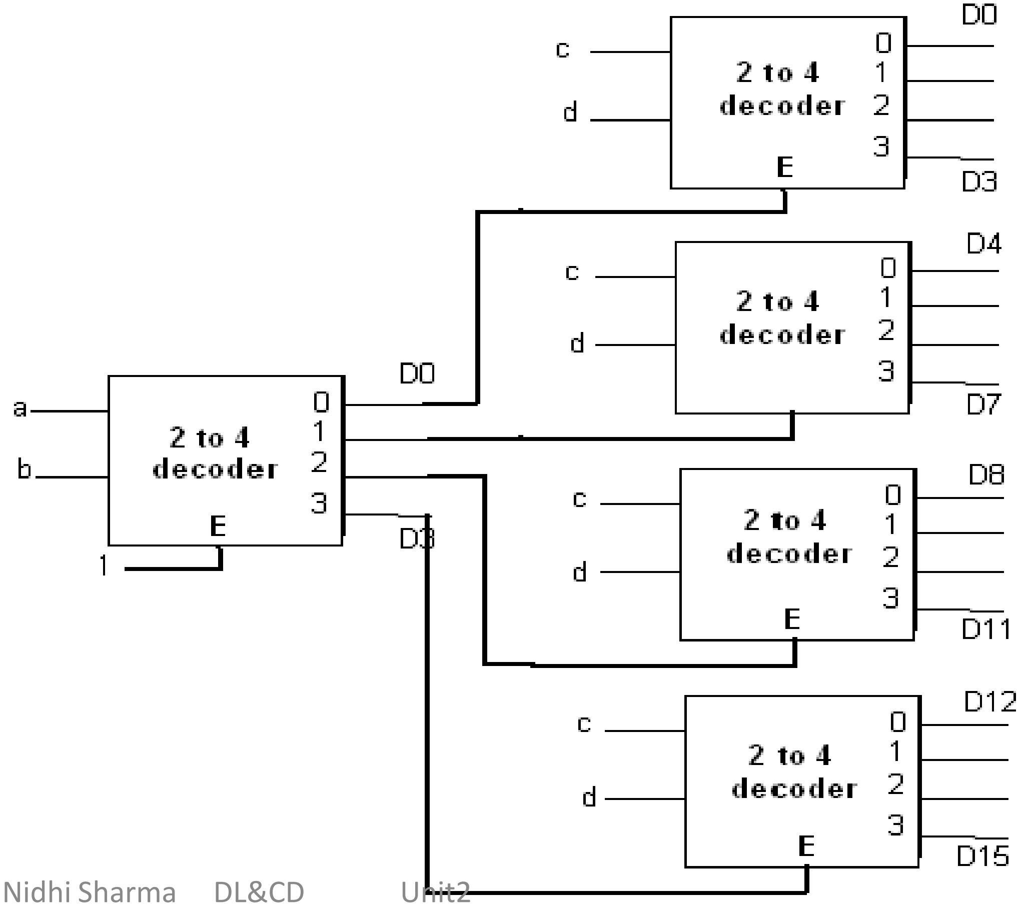
- The **block diagram** of 3 to 8 decoder using 2 to 4 decoders is shown in the following figure.
- Here MSB A_2 will act as enable line.



Decoder

Implement 4*16 decoder using 2 *4 decoders.

- m1 = 4 and m2 = 16 in the formula.
- Required number of 3*8 decoders = 16/4 = 4
- MSB a and b will act as input line for first decoder and c and d will act as input lines for rest of the decoders.



Daily Quiz

- The number of 2:4 decoders required to implement 6:64 decoder are:

  a) 21

  b) 20

  c) 16

  d) 18

- A decoder converts n inputs to ___________ outputs.

  a) n

  b) n^2

  c) 2^n

  d) n^n

- BCD to seven segment conversion is a __________________

  a) Decoding process

  b) Encoding process

  c) Comparing process

  d) None of the mentioned

Faculty Video Links, Youtube & NPTEL Video Links and Online Courses Details

- Youtube/other Video Links
  - https://www.youtube.com/watch?v=SzV4l0_1MCQ&list=PLBlnK6fEyqRjMH3mWf6kwqiTbT798eAOm&index=100
  - https://nptel.ac.in/courses/117/106/117106086/

Old Questions

- Implement 4:16 decoder using 2:4 decoders.
- Implement full adder using decoder.
- Implement 3:8 decoder using 2:4 decoder.
- Implement full subtractor using 3:8 decoder.

Recap

- A *decoder is* a combinational circuit that converts binary information from *n* input lines to a maximum of 2^n unique output lines.
- **Implement the following two higher-order decoders using lower-order decoders 3*8 using 2*4 decoder .**
  - We can find the number of lower order decoders required for implementing higher order decoder using the following formula.
  - Required number of lower order decoders =m2 / m1
  - Where, m1 is the number of outputs of lower order decoder.
    - m2 is the number of outputs of higher order decoder.

Encoders

Topic objective	Mapping with CO
To understand the working of Encoder.	CO2
To understand the circuit diagram of Encoder.	CO2

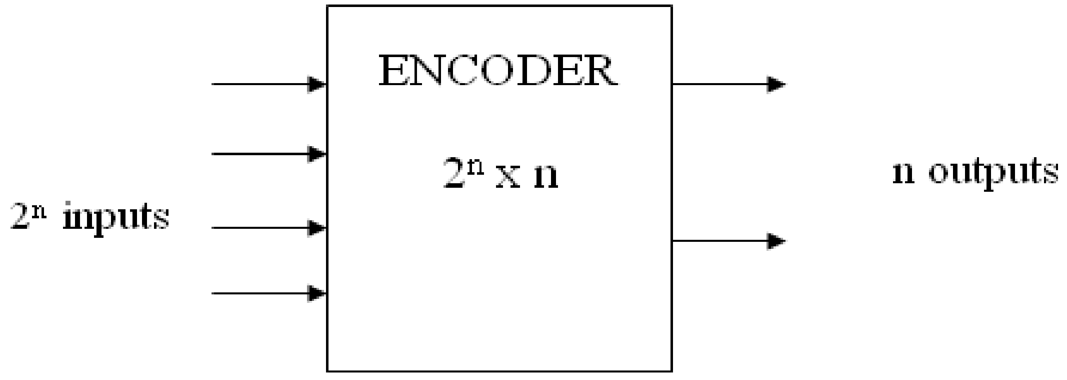
Decoders & Encoders

Prerequisite:

- Basic knowledge of logic gates.
- Basic knowledge of Boolean algebra.

Encoder

- An encoder is a device, circuit, software program, algorithm or person that converts information from one format or code to another. The purpose of encoder is standardization, speed, secrecy, security, or saving space by shrinking size. If a device output code has fewer bits than the input code has, the device is usually called an encoder.



No. of inputs =< 2^n
No. of outputs= n

Encoder Block diagram

Encoder

Uses of Encoders :

- Encoders are very common electronic circuits used in all digital systems.
- Encoders are used to translate the decimal values to the binary in order to perform the binary functions such as addition, subtraction, multiplication, etc.
- Other applications especially for Priority Encoders may include detecting interrupts in microprocessor applications.

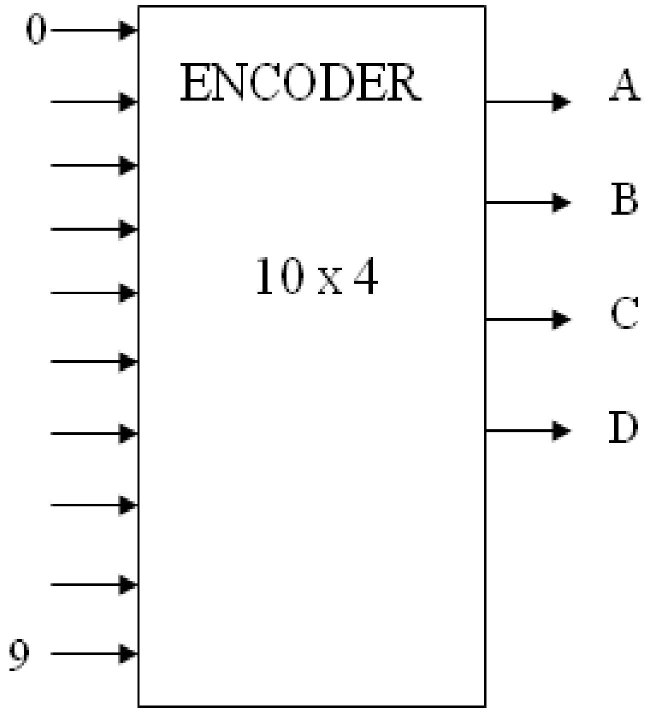
BCD Encoder

Example: Design a Decimal to BCD encoder?

Solution:

No. of inputs= 10
No. of outputs= 4

Input	Output			
	A	B	C	D
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1



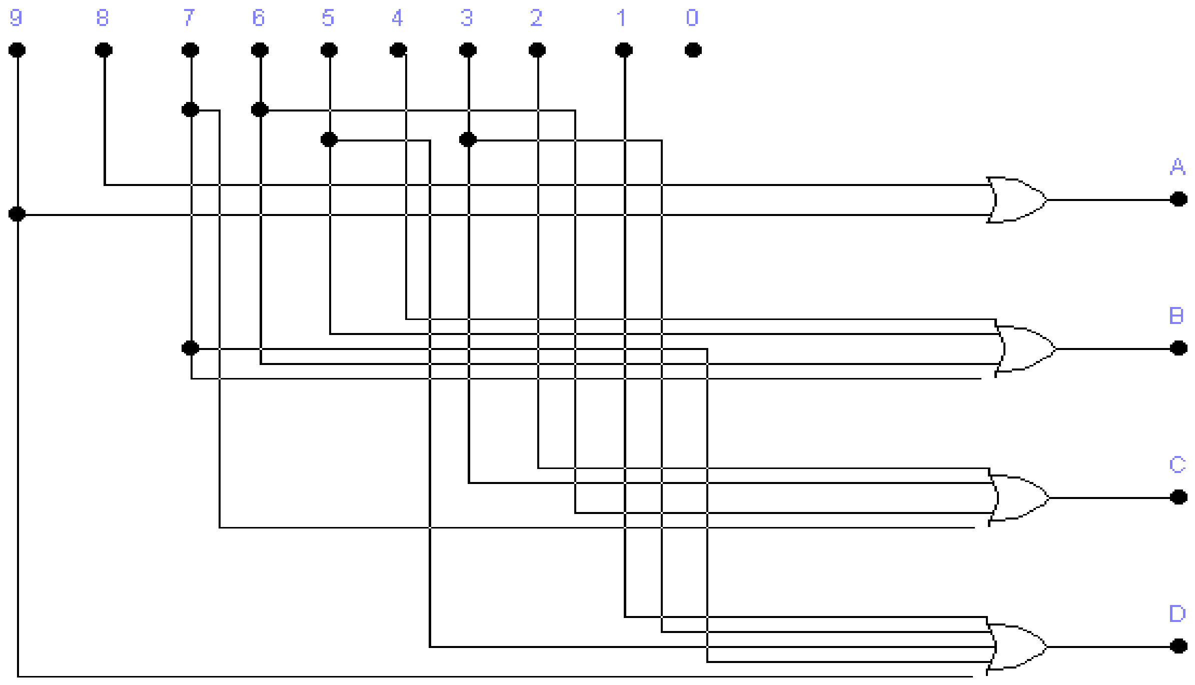
Encoder

From the truth table: $A = \sum m\,(8, 9)$

$B = \sum m\,(4, 5, 6, 7)$

$C = \sum m\,(2, 3, 6, 7)$

$D = \sum m\,(1, 3, 5, 7, 9)$



Decimal to BCD Encoder logical diagram

Encoder

Drawbacks of Normal Encoders –

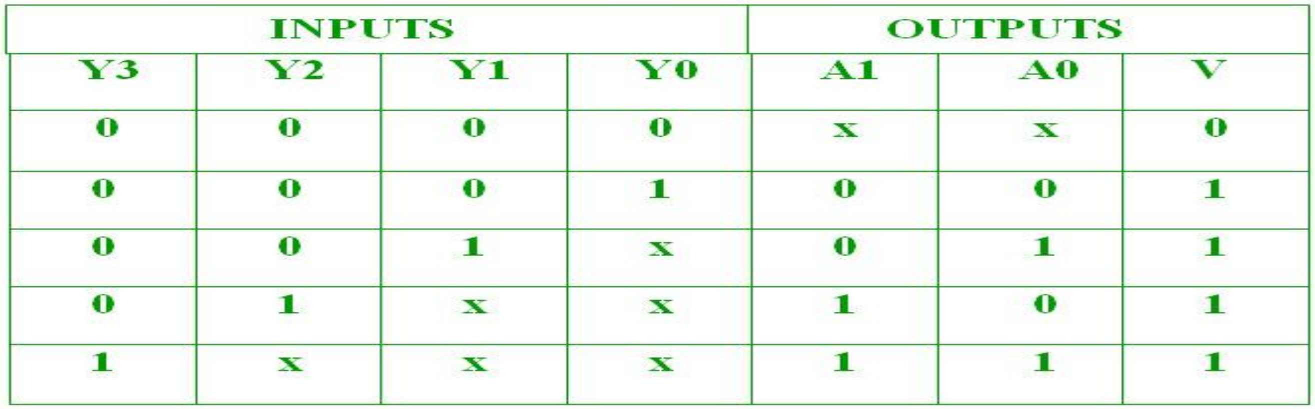
- There is an ambiguity, when all outputs of encoder are equal to zero.
- If more than one input is active High, then the encoder produces an output, which may not be the correct code.

So, to overcome these difficulties, we should assign priorities to each input of encoder. Then, the output of encoder will be the (code corresponding to the active High inputs, which has higher priority.

- The ***priority encoders*** output corresponds to the currently active input which has the highest priority. So when an input with a higher priority is present, all other inputs with a lower priority will be ignored.

Encoder

- A 4 to 2 priority encoder has **4 inputs** : Y3, Y2, Y1 & Y0 and **2 outputs** : A1 & A0. Here, the input, Y3 has the **highest priority**, whereas the input, Y0 has the **lowest priority**. In this case, even if more than one input is ‘1’ at the same time, the output will be the (binary) code corresponding to the input, which is having **higher priority**.
- The truth table for priority encoder is as follows :



INPUTS				OUTPUTS		
Y3	Y2	Y1	Y0	A1	A0	V
0	0	0	0	x	x	0
0	0	0	1	0	0	1
0	0	1	x	0	1	1
0	1	x	x	1	0	1
1	x	x	x	1	1	1

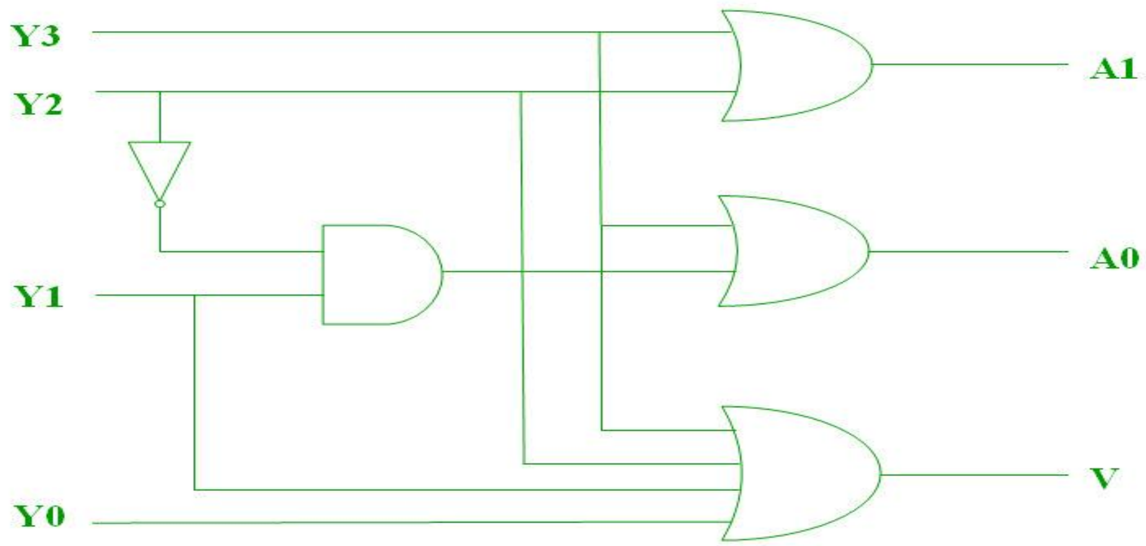
Encoder

Y3 Y2 \ Y1 Y0	00	01	11	10
00	X	0	0	0
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

A1=Y3 + Y2

Y3 Y2 \ Y1 Y0	00	01	11	10
00	X	0	1	1
01	0	0	0	0
11	X	X	X	X
10	1	1	1	1

A0=Y3 + Y2' Y1



Faculty Video Links, Youtube & NPTEL Video Links and Online Courses Details

- Youtube/other Video Links
  - https://www.youtube.com/watch?v=SzV4l0_1MCQ&list=PLBlnK6fEyqRjMH3mWf6kwqiTbT798eAOm&index=100
  - https://www.youtube.com/watch?v=Z_DYRgtAXfw&list=PLBlnK6fEyqRjMH3mWf6kwqiTbT798eAOm&index=103
  - https://www.youtube.com/watch?v=sUutDs7FFeA&ab_channel=nptelhrdnptelhrd

Daily Quiz

- If two inputs are active on a priority encoder which will be coded on the output:

  a) The higher priority

  b) The lower priority

- Which of the following are building blocks of encoders?

  a) NOT gate

  b) OR gate

  c) AND gate

  d) NAND gate

- Decoders and Encoders are doing reverse operation.

  a) True

  b) False

Old Questions

- What is the advantage of priority encoder?
- Design and explain priority encoder.
- Why priority encoder is used?

Weekly Assignment

- Implement 4:16 decoder using 2:4 decoders.
- Implement 3:8 decoder using 2:4 decoder.
- What is the advantage of priority encoder?
- Implement full subtractor using 3:8 decoder.
- Design 3:8 priority encoder.

Recap

- An encoder is a device, circuit, software program, algorithm or person that converts information from one format or code to another.
- Encoders are used to translate the decimal values to the binary in order to perform the binary functions such as addition, subtraction, multiplication, etc.
- The ***priority encoders*** output corresponds to the currently active input which has the highest priority.

Comparators

Topic objective	Mapping with CO
To understand magnitude comparators.	CO2

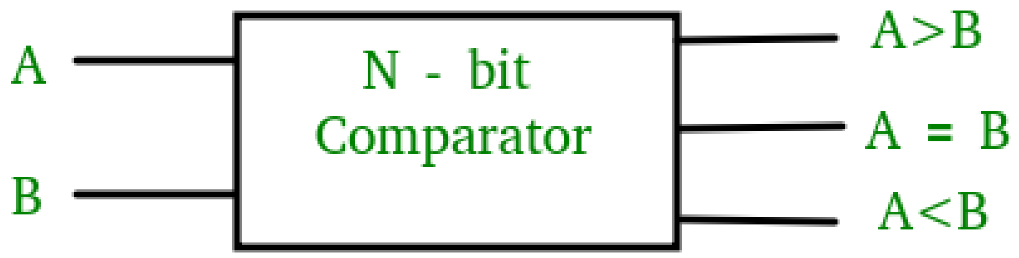
Comparators

Prerequisite:

- Basic knowledge of logic gates.
- Basic knowledge of Boolean algebra.

Magnitude Comparators

- A magnitude digital Comparator is a combinational circuit that **compares two digital or binary numbers** in order to find out whether one binary number is equal, less than or greater than the other binary number. We logically design a circuit for which we will have two inputs one for A and other for B and have three output terminals, one for A > B condition, one for A = B condition and one for A < B condition.

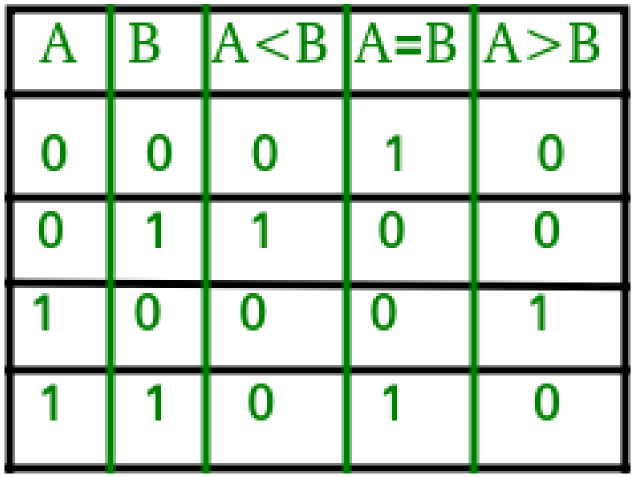


Magnitude Comparators

1-Bit Magnitude Comparator :

- A comparator used to compare two bits is called a single bit comparator. It consists of two inputs each for two single bit numbers and three outputs to generate less than, equal to and greater than between two binary numbers.

The truth table for a 1-bit comparator is given below:

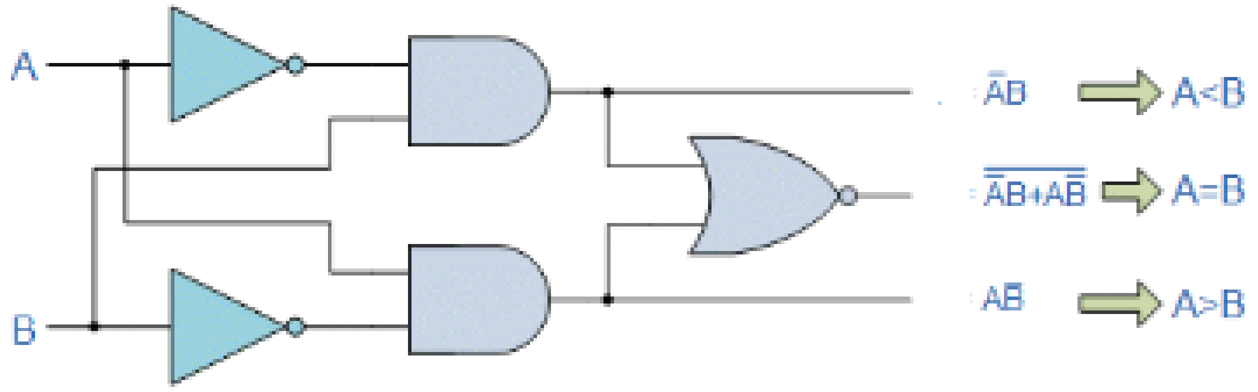


A	B	A<B	A=B	A>B
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

Magnitude Comparators

- From the above truth table logical expressions for each output can be expressed as follows:

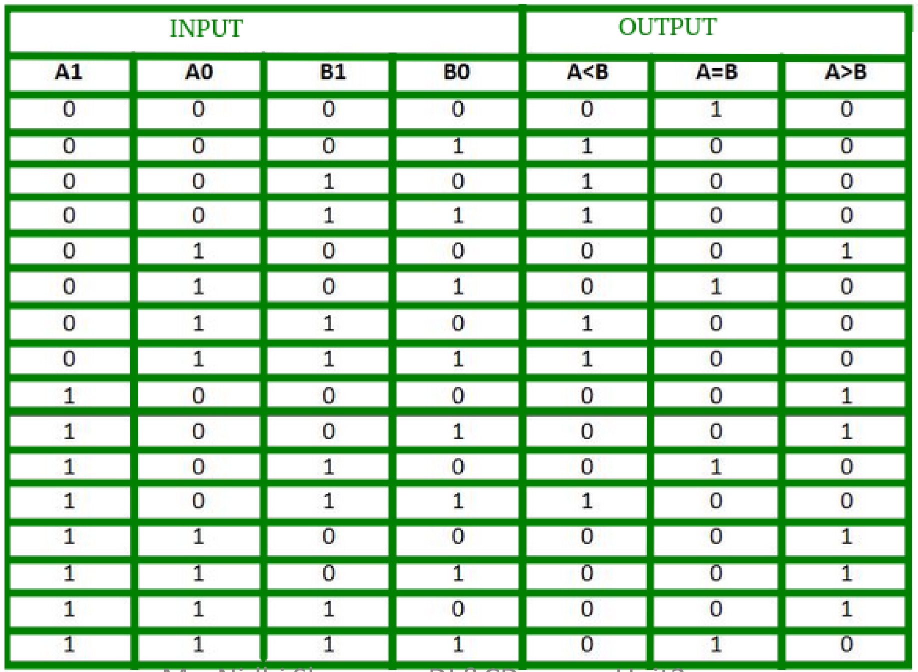
  $(A>B): AB'$, $(A<B) : A'B$, $(A=B) : = A'B' + AB^{\odot} = A \quad B$
- By using these Boolean expressions, we can implement a logic circuit for this comparator as given below



Magnitude Comparators

2-Bit Magnitude Comparator :

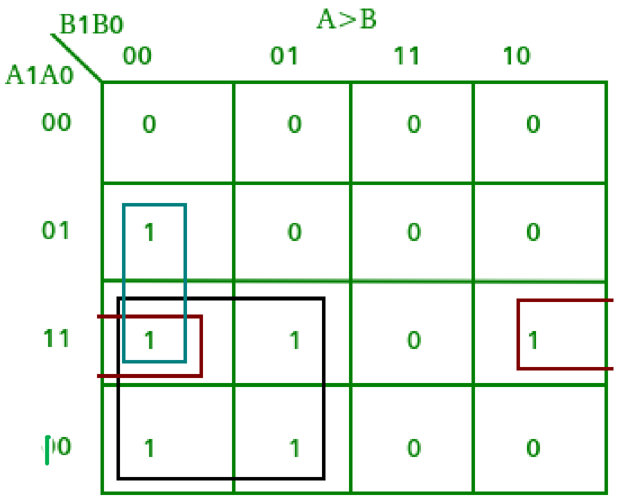
- A comparator used to compare two binary numbers each of two bits is called a 2-bit Magnitude comparator. It consists of four inputs and three outputs to generate less than, equal to and greater than between two binary numbers.
- The truth table for a 2-bit comparator is given below



INPUT				OUTPUT		
A1	A0	B1	B0	A<B	A=B	A>B
0	0	0	0	0	1	0
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	0

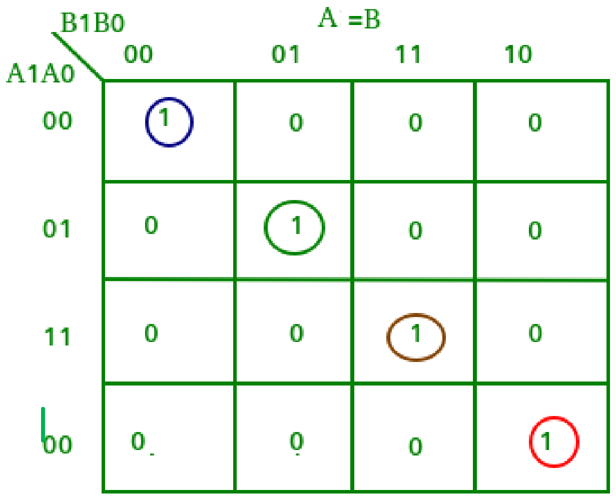
Magnitude Comparators

- From the truth table K-map for each output can be drawn as follows:



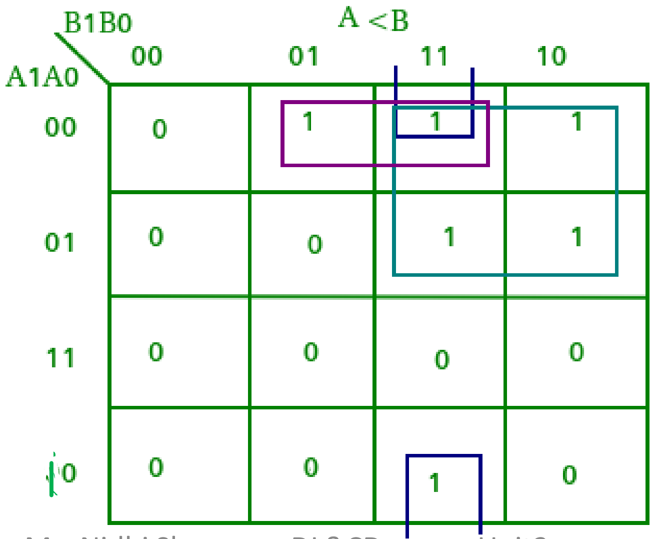
A>B

A1A0 \ B1B0	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	1	1	0	1
10	1	1	0	0



A =B

A1A0 \ B1B0	00	01	11	10
00	1	0	0	0
01	0	1	0	0
11	0	0	1	0
10	0	0	0	1



A <B

A1A0 \ B1B0	00	01	11	10
00	0	1	1	1
01	0	0	1	1
11	0	0	0	0
10	0	0	1	0

Magnitude Comparators

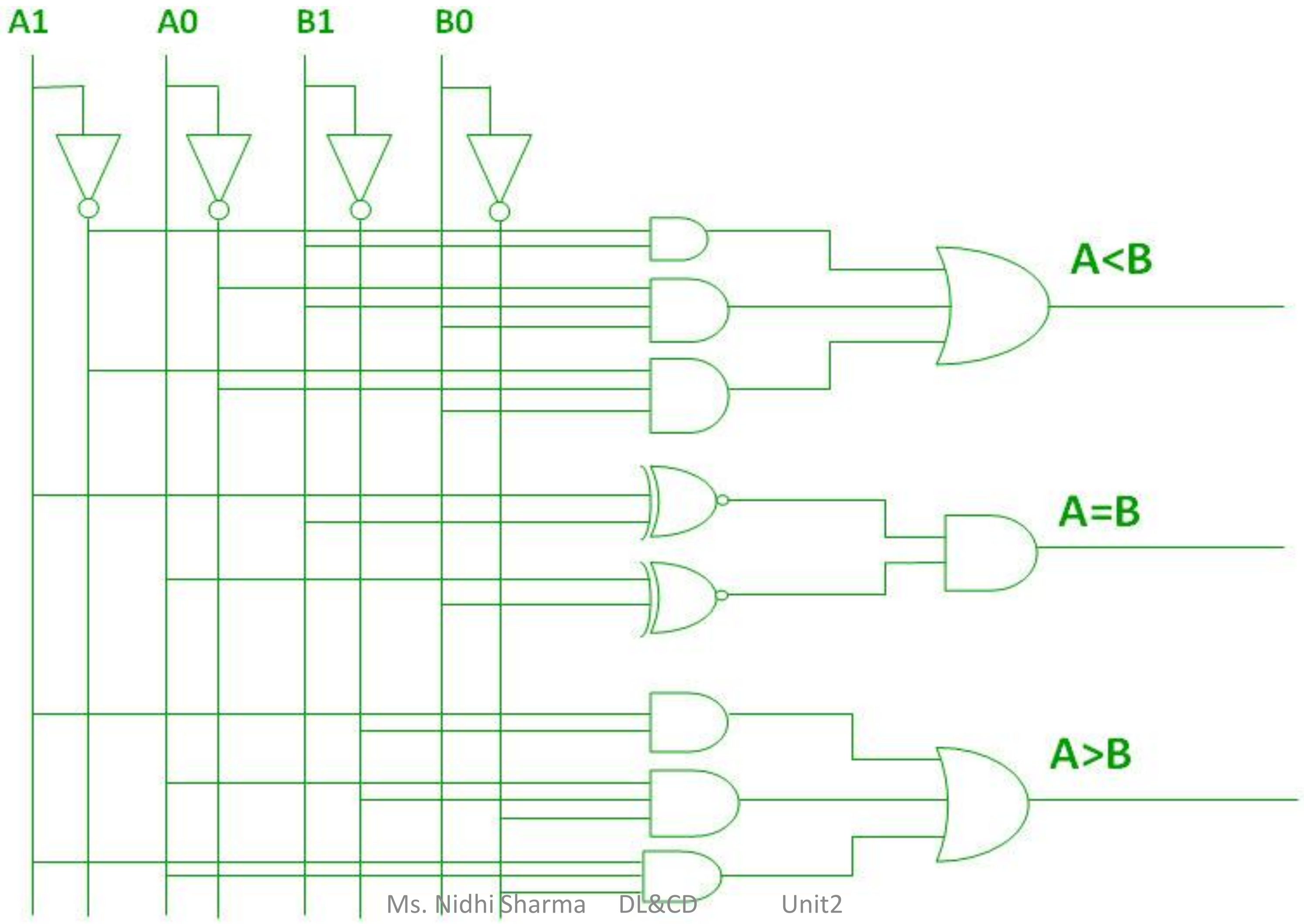
- From K-maps logical expressions for each output can be expressed as follows:
- (A>B): **$A_1B_1' + A_0B_1'B_0' + A_1A_0B_0'$**
  - $A_1B_1' + A_1'A_0B_1'B_0' + A_1A_0\,B_1\,B_0'$
  - $A_1B_1' + A_0B_0'[\,A_1`\,B_1' + A_1\,B_1\,]$
  - $A_1B_1' + X_1\,A_0\,B_0'$

- (A=B): **$A_1'A_0'B_1'B_0' + A_1'A_0B_1'B_0 + A_1A_0B_1B_0 + A_1A_0'B_1B_0'$**
  - $A_1'B_1'\,(A_0'B_0' + A_0B_0\,) + A_1B_1\,(A_0B_0 + A_0'B_0')$
  - $(A_0'B_0' + A_0B_0\,).\,(A_1'B_1' + A_1B_1\,)$
  - $(A_0 \quad B_0)\,(A_1 \odot B_1)$
  - $X_1 \odot X_0$

- (A<B): **$A_1'B_1 + A_0'B_1B_0 + A_1'A_0'B_0$**
  - $A_1'B_1 + A_1'A_0'\,B_1'B_0 + A_1A_0'\,B_1\,B_0$
  - $A_1'B_1 + A_0'\,B_0[\,A_1'\,B_1' + A_1\,B_1\,]$
  - $A_1'\,B_1 + X_1A_0'\,B_0$

Magnitude Comparators

- By using these Boolean expressions, we can implement a logic circuit for this comparator as given below:



Magnitude Comparators

4-Bit Magnitude Comparator:

- A comparator used to compare two binary numbers each of four bits is called a 4-bit magnitude comparator. It consists of eight inputs each for two four bit numbers and three outputs to generate less than, equal to and greater than between two binary numbers.
- The algorithm is a direct application of the procedure a person uses to compare the relative magnitudes of two numbers. Consider two numbers, A and B, with four digits each. Write the coefficients of the numbers with descending significance as follows:
- $A = A_3 A_2 A_1 A_0$ and $B = B_3 B_2 B_1 B_0$
- where each subscripted letter represents one of the digits in the number. The two numbers are equal if all pairs of significant digits are equal, i.e., if $A_3 = B_3$ and $A_2 = B_2$ and $A_1 = B_1$ and $A_0 = B_0$.

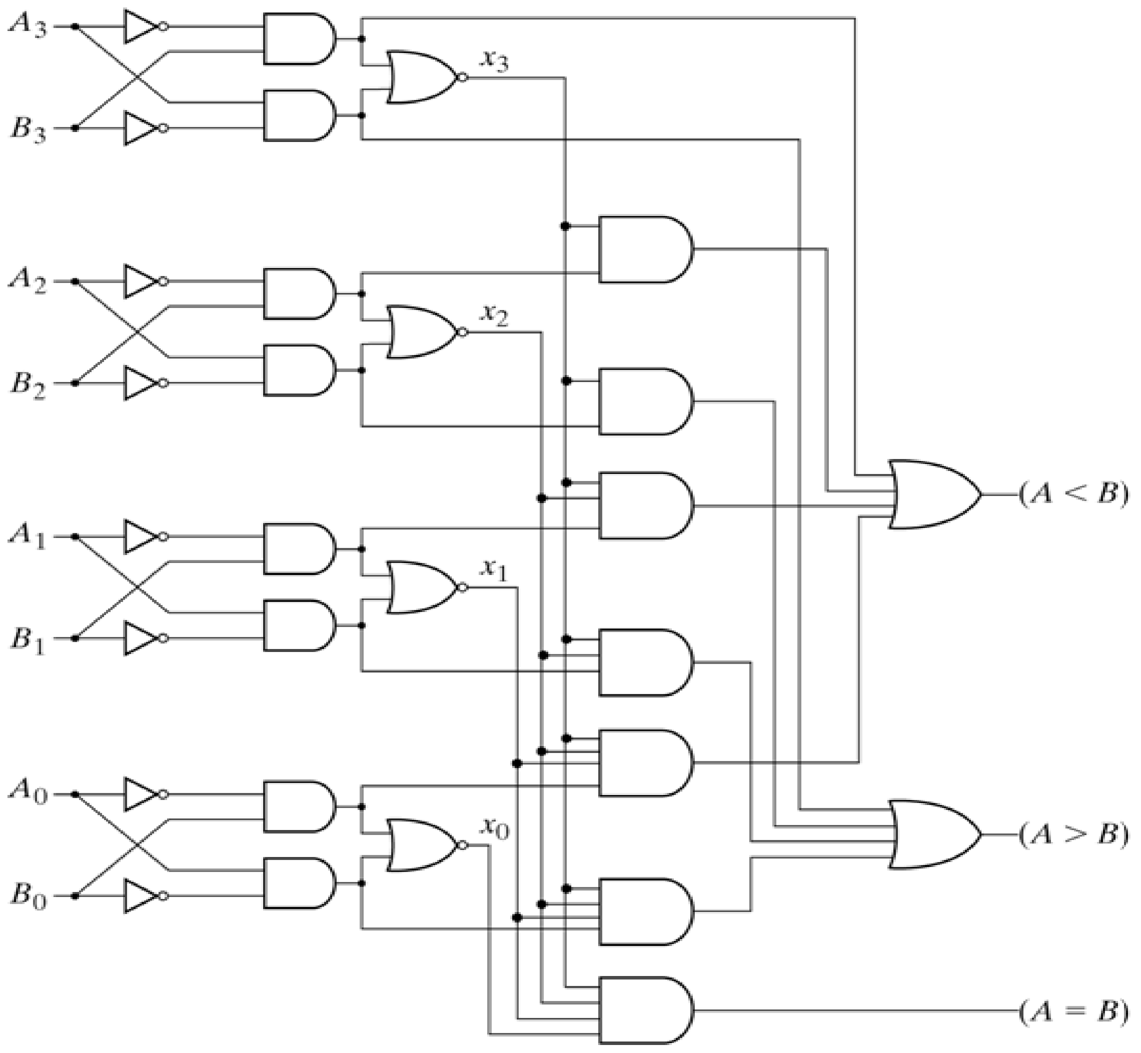
Magnitude Comparators

- When the numbers are binary, the digits are either 1 or 0 and the equality relation of each pair of bits can be expressed logically with an equivalence function:
- $X_i = A_iB_i + A_i'B_i' = 0, 1, 2, 3$ where $X_i = 1$ only if the pair of bits in position i are equal, i.e., if both are 1's or both are 0's.
- The equality of the two numbers, A and B, is displayed in a combinational circuit by an output binary variable that we designate by the symbol (A = B). This binary variable is equal to 1 if the input numbers, A and B, are equal, and it is equal to 0 otherwise.
- For the equality condition to exist, all Xi variables must be equal to 1. This dictates an AND operation of all variables:
- $(A = B) = X_3X_2X_1X_0$
- The binary variable (A = B) is equal to 1 only if all pairs of digits of the two numbers are equal.

Magnitude Comparators

- To determine if A is greater than or less than B, we inspect the relative magnitudes of pairs of significant digits starting from the most significant position. If the two digits are equal, we compare the next lower significant pair of digits. This comparison continues until a pair of unequal digits is reached.
- If the corresponding digit of A is 1 and that of B is 0, we conclude that A > B. If the corresponding digit of A is 0 and that of B is 1, we have that A < B. The sequential comparison can be expressed logically by the following two Boolean functions:
- $(A > B) = A_3B_3' + X_3A_2B_2' + X_3X_2A_1B_1' + X_3X_2X_1A_0B_0'$
- $(A < B) = A_3'B_3 + X_3A_2'B_2 + X_3X_2A_1'B_1 + X_3X_2X_1A_0'B_0$
- The symbols (A > B) and (A < B) are binary output variables that are equal to 1 when A > B or A < B, respectively.

Magnitude Comparators



Circuit Diagram of 4 bit comparator

Fig. 4-17 4-Bit Magnitude Comparator

Faculty Video Links, Youtube & NPTEL Video Links and Online Courses Details

- Youtube/other Video Links
  - ➢ https://nptel.ac.in/courses/117/106/117106086/
  - ➢ https://www.youtube.com/watch?v=BhUUmbz76P0

Daily Quiz

- Which of the following gate is used to check the equality condition?

  a) Ex-OR

  b) OR

  c) NOR

  d) Ex-NOR

- One that is not the outcome of magnitude comparator is

  a) a > b

  b) a – b

  c) a < b

  d) a = b

- If two numbers are not equal then binary variable will be

  a) 0

  b) 1

  c) A

  d) B

Old Questions

- Design a 4 bit magnitude comparator.
- Design a 1 bit magnitude comparator.

Recap

- A magnitude digital Comparator is a combinational circuit that **compares two digital or binary numbers** in order to find out whether one binary number is equal, less than or greater than the other binary number.
- We logically design a circuit for which we will have two inputs one for A and other for B and have three output terminals, one for A > B condition, one for A = B condition and one for A < B condition.

Code Converters

Topic objective	Mapping with CO
To understand the working of code converters.	CO2
To understand the circuit diagram of code converters.	CO2

Code Converters

Prerequisite:

- Basic knowledge of logic gates.
- Basic knowledge of Boolean algebra.

Code Converters

- Coding is the process of translating the input information which can be understandable by the machine or a particular device. Coding can be used for security purpose to protect the information from steeling or interrupting
- The code converters are used to convert the information into the code which we want. These are basically encoders and decoders which converts the data into an encoded form.
- There are various types of code converters :
  - Binary to Gray code converter
  - Gray to Binary code converter
  - Excess-3 to BCD
  - BCD to Excess-3 converter

Code Converter

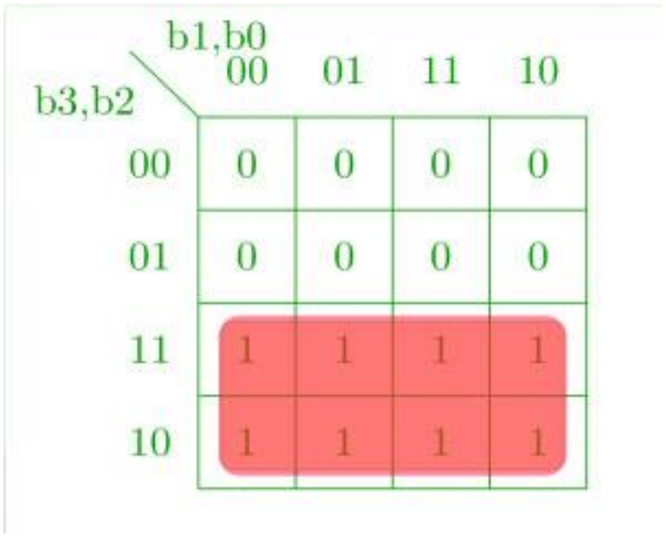
Example: Design a combinational logic circuit that converts 4 bit binary to gray code?

- Let g_3, g_2, g_1, g_0 be the bits representing the gray numbers, and b_3, b_2, b_1, b_0 be the bits representing the gray code of the binary numbers, where g_0 & b_0 is the LSB and g_3 , b_3 is the MSB

Binary				Gray Code			
b_3	b_2	b_1	b_0	g_3	g_2	g_1	g_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

Code Converter

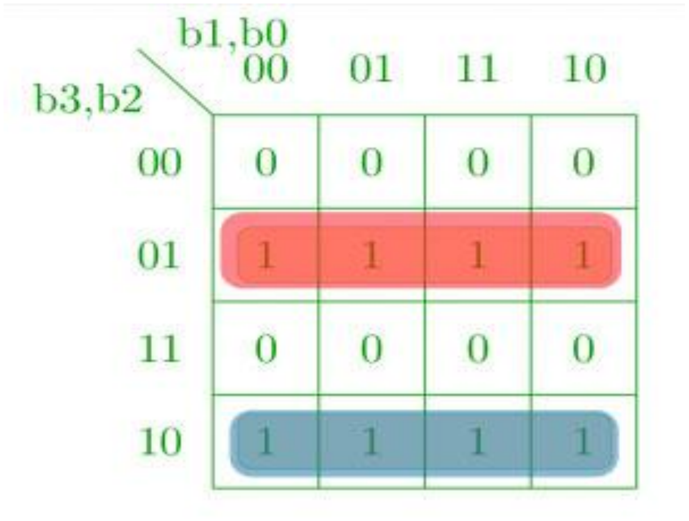
K-map for g_3 :



b3,b2 \ b1,b0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

$$g_3 = b_3$$

K-map for g_2 :

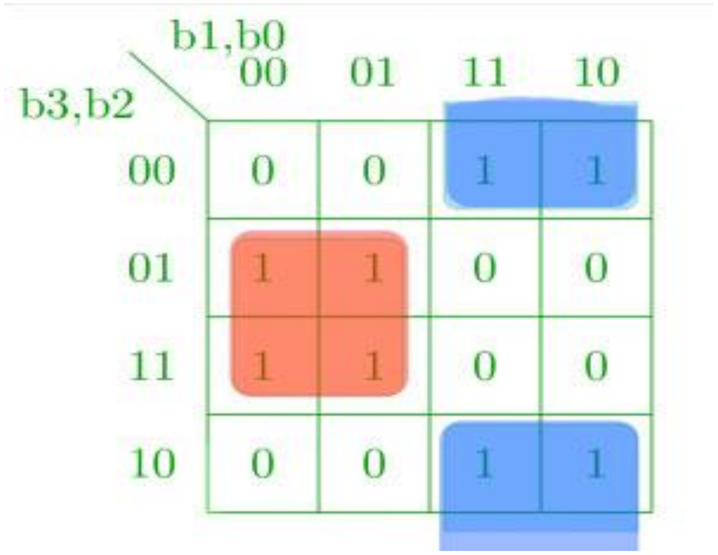


b3,b2 \ b1,b0	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

$$g_2 = b_2 b_3' + b_3 b_2' = b_2 \oplus b_3$$

Code Converter

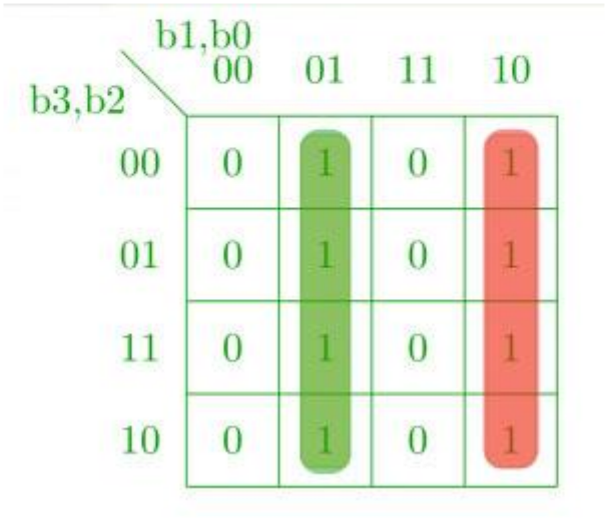
K-map for g_1 :



b3,b2 \ b1,b0	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	1	1	0	0
10	0	0	1	1

$$g_1 = b_2 b_1' + b_1 b_2' = b_1 \oplus b_2$$

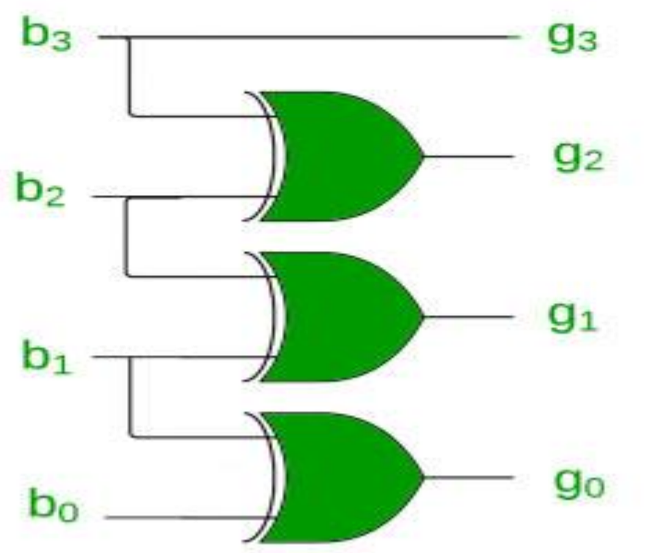
K-map for g_0 :



b3,b2 \ b1,b0	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

$$g_0 = b_0 b_1' + b_1 b_0' = b_0 \oplus b_1$$

Code Converter

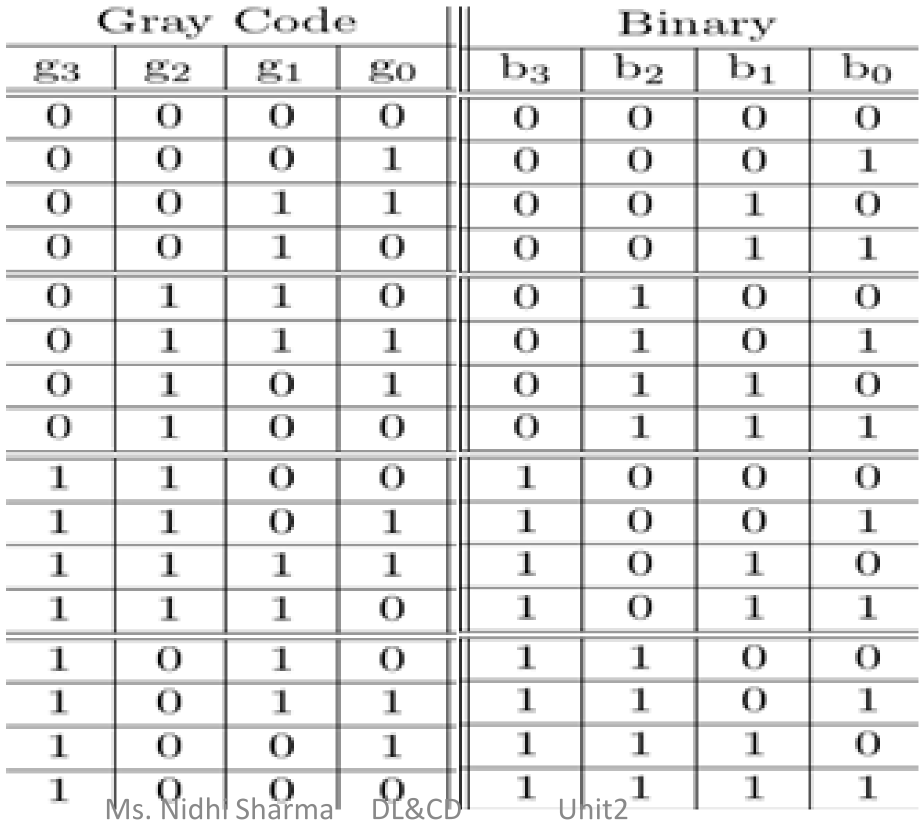


4bit binary to gray code converter

Code Converter

Eg. Design a combinational logic circuit that converts 4-bit gray to binary code?

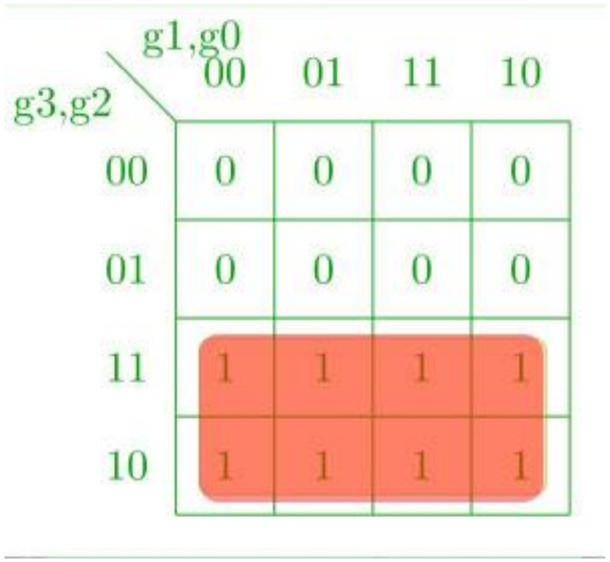
- Let g_3, g_2, g_1, g_0 be the bits representing the gray numbers, and b_3, b_2, b_1, b_0 be the bits representing the gray code of the binary numbers, where g_0 & b_0 is the LSB and g_3, b_3 is the MSB.



Gray Code				Binary			
g_3	g_2	g_1	g_0	b_3	b_2	b_1	b_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
0	1	0	1	0	1	1	0
0	1	0	0	0	1	1	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	1	1	0	1	0
1	1	1	0	1	0	1	1
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	0	0	1	1	1	1	0
1	0	0	0	1	1	1	1

Code Converter

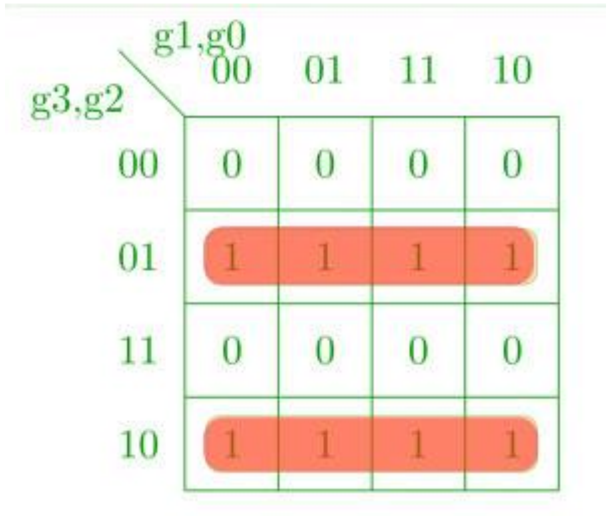
K-map for b_3:



g3,g2 \ g1,g0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

$$b_3 = g_3$$

K-map for b_2:

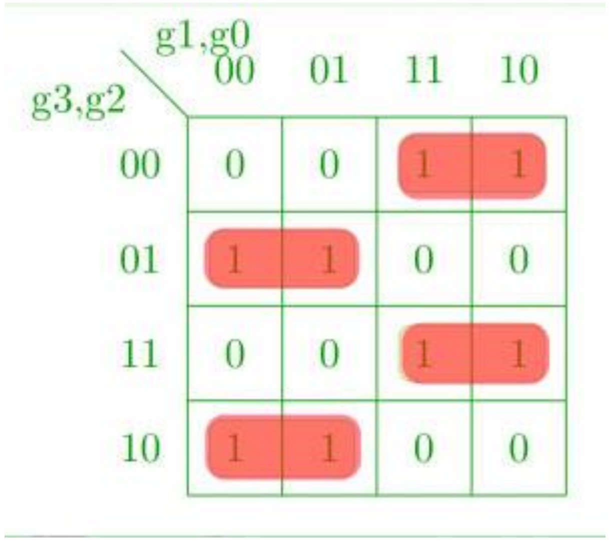


g3,g2 \ g1,g0	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

$$\begin{aligned} b_2 &= g_3' g_2 + g_3 g_2' \\ &= g_3 \oplus g_2 \end{aligned}$$

Code Converter

K-map for b_1: :

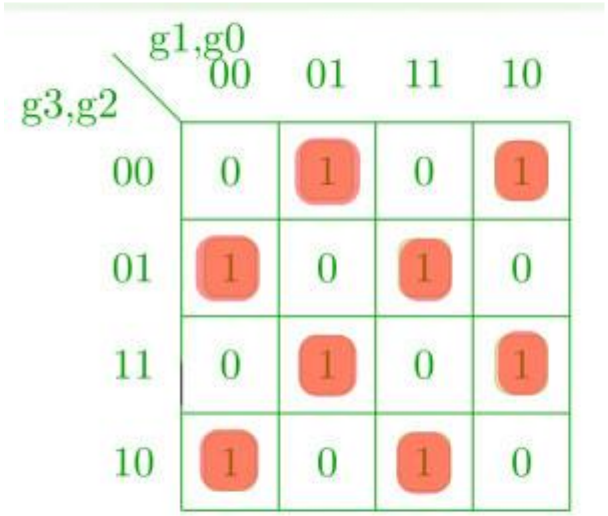


g3,g2 \ g1,g0	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	0	0	1	1
10	1	1	0	0

$$\begin{aligned} b_1 &= g_3'g_2'g_1 + g_3'g_2g_1' + g_3g_2g_1 + g_3g_2'g_1' \\ &= g_3'(g_2'g_1 + g_2g_1') + g_3(g_2g_1 + g_2'g_1') \\ &= g_3'(g_2 \oplus g_1) + g_3(g_2 \odot g_1) \\ &= g_3 \oplus g_2 \oplus g_1 \end{aligned}$$

Code Converter

K-map for b_0:

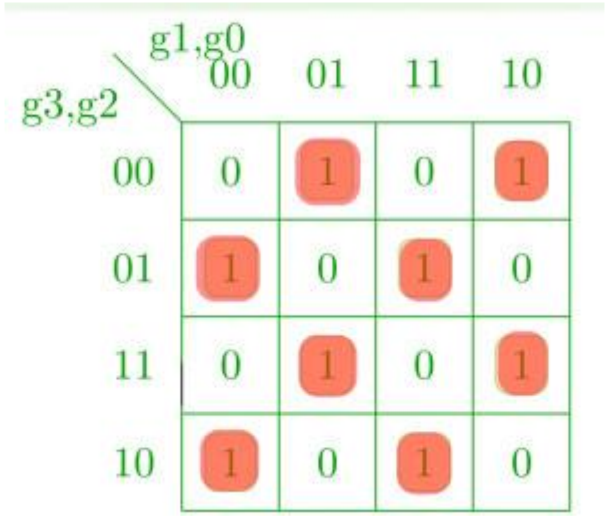


g3,g2 \ g1,g0	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0

$$\begin{aligned}
b_0 &= g_3'g_2'g_1'g_0 + g_3'g_2'g_1g_0' + g_3'g_2g_1'g_0' + g_3'g_2g_1g_0 + g_3g_2'g_1'g_0' + g_3g_2g_1g_0 \\
&\quad + g_3g_2g_1'g_0 + g_3g_2g_1g_0' \\
&= g_3'g_2'(g_1'g_0 + g_1g_0') + g_3'g_2(g_1'g_0' + g_1g_0) + g_3g_2'(g_1'g_0' + g_1g_0) \\
&\quad + g_3g_2(g_1'g_0 + g_1g_0') \\
&= g_3'g_2'(g_0 \oplus g_1) + g_3'g_2(g_0 \odot g_1) + g_3g_2'(g_0 \odot g_1) + g_3g_2(g_0 \oplus g_1) \\
&= (g_0 \oplus g_1)(g_2 \odot g_3) + (g_0 \odot g_1)(g_2 \oplus g_3) \\
&= g_3 \oplus g_2 \oplus g_1 \oplus g_0
\end{aligned}$$

Code Converter

K-map for b_0:

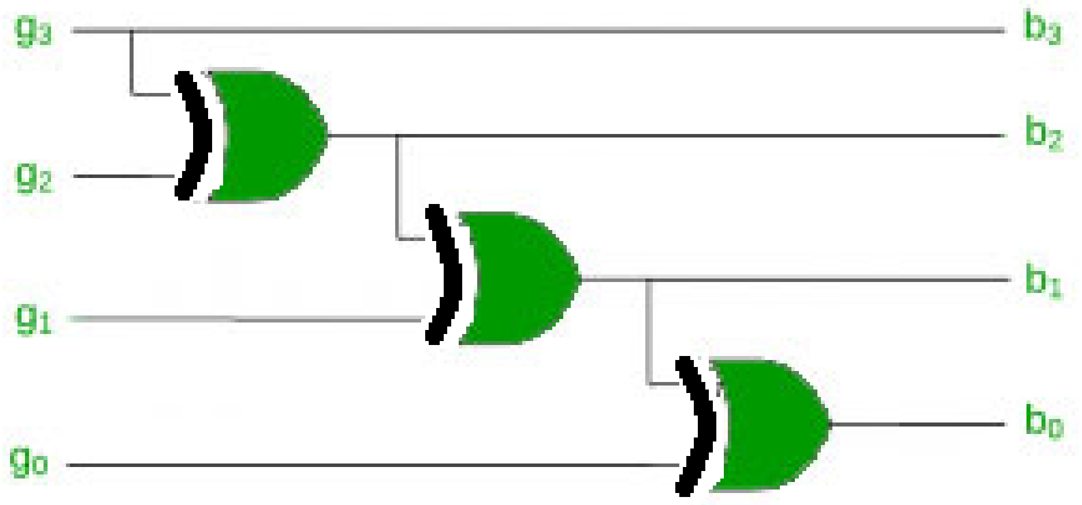


g3,g2 \ g1,g0	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0

$$\begin{aligned}
b_0 &= g_3'g_2'g_1'g_0 + g_3'g_2'g_1g_0' + g_3'g_2g_1'g_0' + g_3'g_2g_1g_0 + g_3g_2'g_1'g_0' + g_3g_2'g_1g_0 \\
&\quad + g_3g_2g_1'g_0 + g_3g_2g_1g_0' \\
&= g_3'g_2'(g_1'g_0 + g_1g_0') + g_3'g_2(g_1'g_0' + g_1g_0) + g_3g_2'(g_1'g_0' + g_1g_0) \\
&\quad + g_3g_2(g_1'g_0 + g_1g_0') \\
&= g_3'g_2'(g_0 \oplus g_1) + g_3'g_2(g_0 \odot g_1) + g_3g_2'(g_0 \odot g_1) + g_3g_2(g_0 \oplus g_1) \\
&= (g_0 \oplus g_1)(g_2 \odot g_3) + (g_0 \odot g_1)(g_2 \oplus g_3) \\
&= g_3 \oplus g_2 \oplus g_1 \oplus g_0
\end{aligned}$$

Code Converter

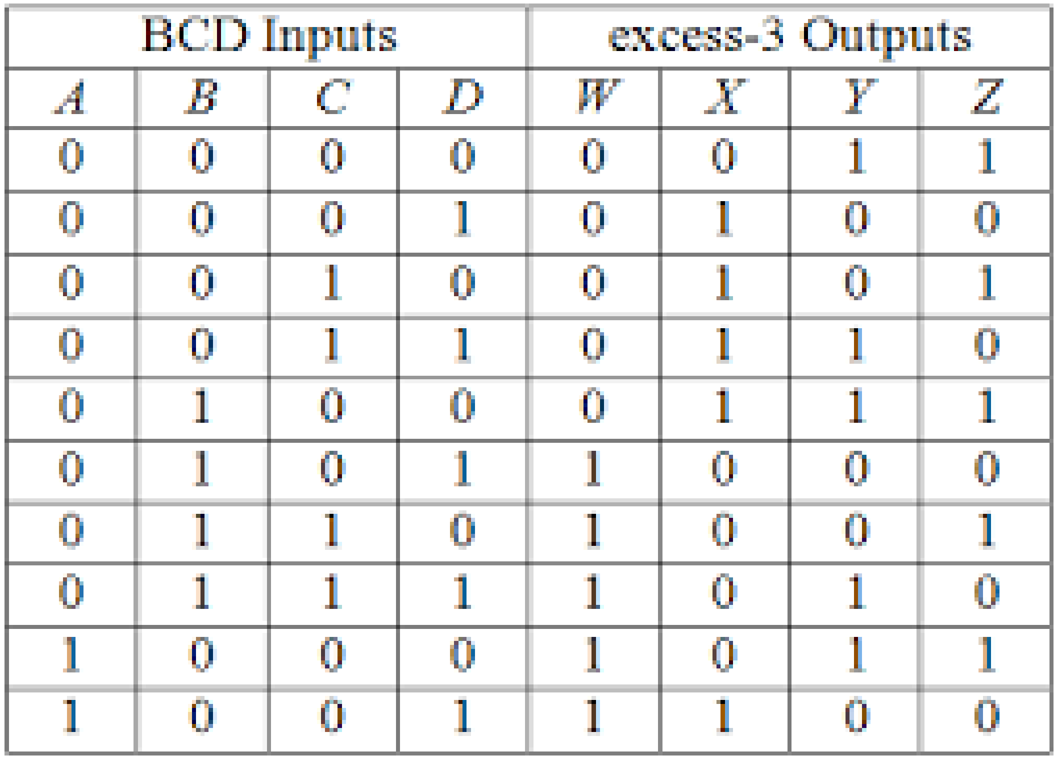
Circuit diagram of 4-bit gray to binary converter :



Code Converter

Design a combinational circuit which converts the BCD to Excess-3 code.

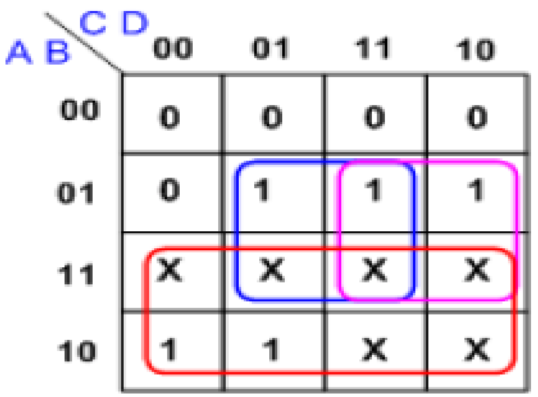
Let A,B,C,D be the bits representing the binary numbers and w,x,y,z be the bits representing the gray code of the binary numbers, where is the A , w are LSB and D ,z is the MSB.



BCD Inputs				excess-3 Outputs			
A	*B*	*C*	*D*	*W*	*X*	*Y*	*Z*
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

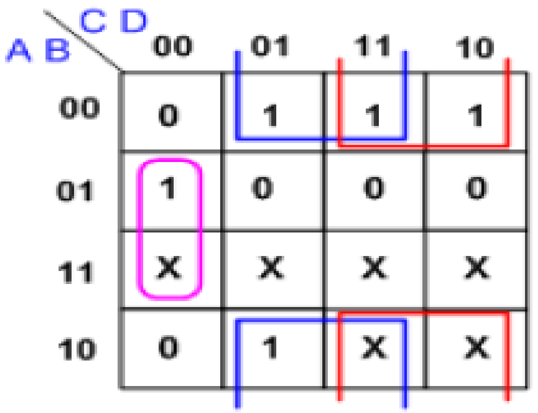
Code Converter

.



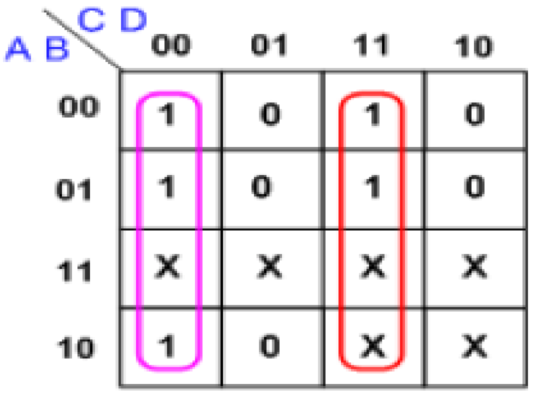
AB \ CD	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	X	X	X	X
10	1	1	X	X

$W = A + BC + BD$



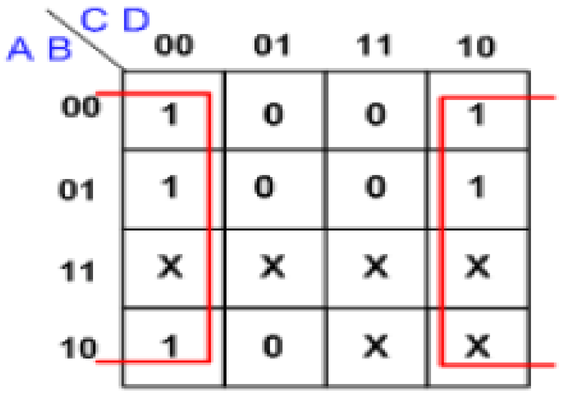
AB \ CD	00	01	11	10
00	0	1	1	1
01	1	0	0	0
11	X	X	X	X
10	0	1	X	X

$X = B'C + BC'D' + B'D$



AB \ CD	00	01	11	10
00	1	0	1	0
01	1	0	1	0
11	X	X	X	X
10	1	0	X	X

$Y = CD + C'D'$



AB \ CD	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	X	X	X	X
10	1	0	X	X

$Z = D'$

Code Converter

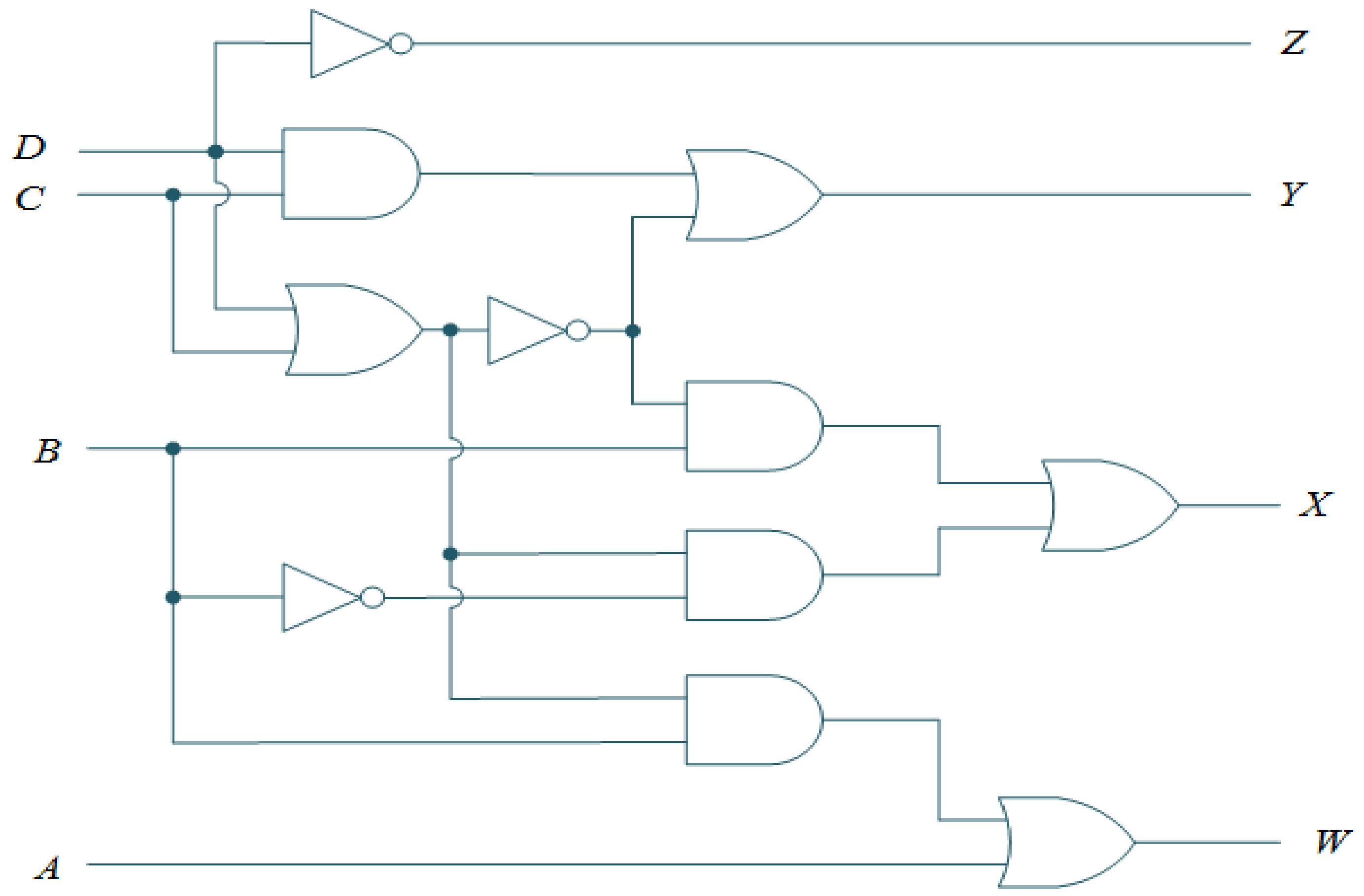
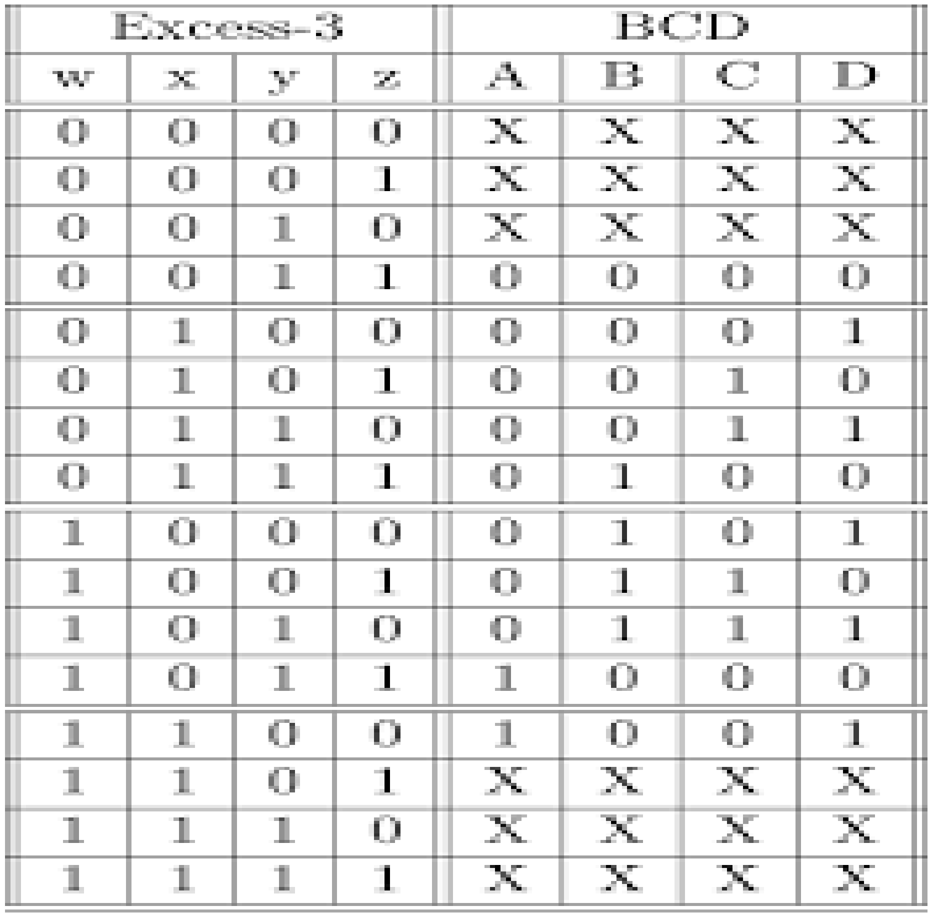


Figure : *BCD to Excess-3 Code Converter logic diagram.*

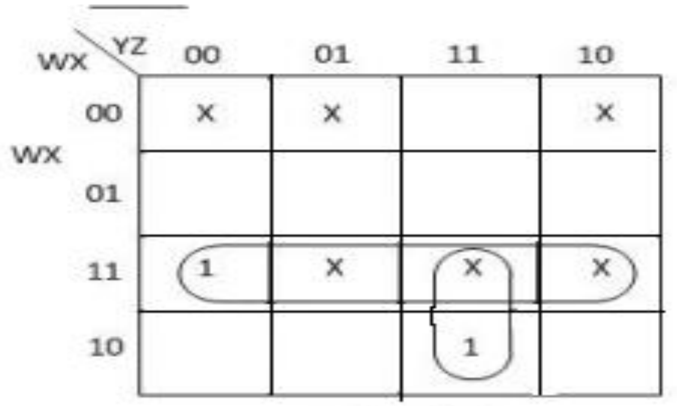
Code Converter

- Design a combinational circuit which converts the Excess-3 code to BCD.



Excess-3				BCD			
w	x	y	z	A	B	C	D
0	0	0	0	X	X	X	X
0	0	0	1	X	X	X	X
0	0	1	0	X	X	X	X
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	1
0	1	0	1	0	0	1	0
0	1	1	0	0	0	1	1
0	1	1	1	0	1	0	0
1	0	0	0	0	1	0	1
1	0	0	1	0	1	1	0
1	0	1	0	0	1	1	1
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	1
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

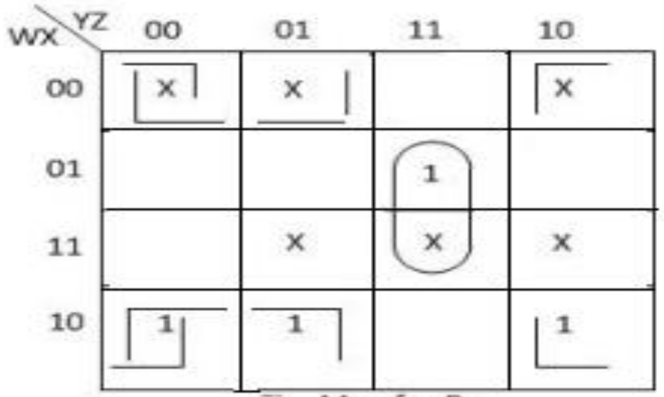
Code Converter



WX \ YZ	00	01	11	10
00	X	X		X
01				
11	1	X	X	X
10			1	

Map for A

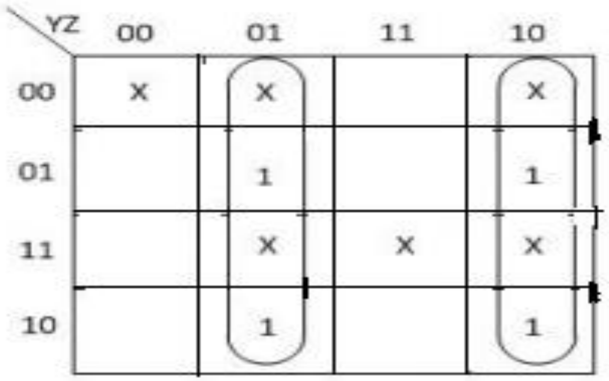
$A = WX + WYZ$



WX \ YZ	00	01	11	10
00	X	X		X
01			1	
11		X	X	X
10	1	1		1

Map for B

$B = X'Y' + XYZ + Z'$



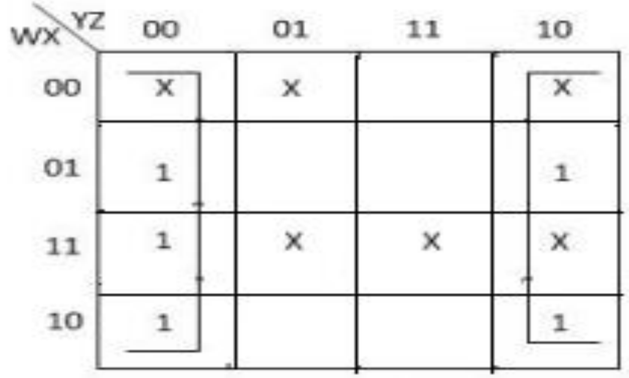
WX \ YZ	00	01	11	10
00	X	X		X
01		1		1
11		X	X	X
10		1		1

Map for C

$C = Y'Z + YZ'$

OR

$C = Y \oplus Z$



WX \ YZ	00	01	11	10
00	X	X		X
01	1			1
11	1	X	X	X
10	1			1

Map for D

$D = Z'$

Code Converter

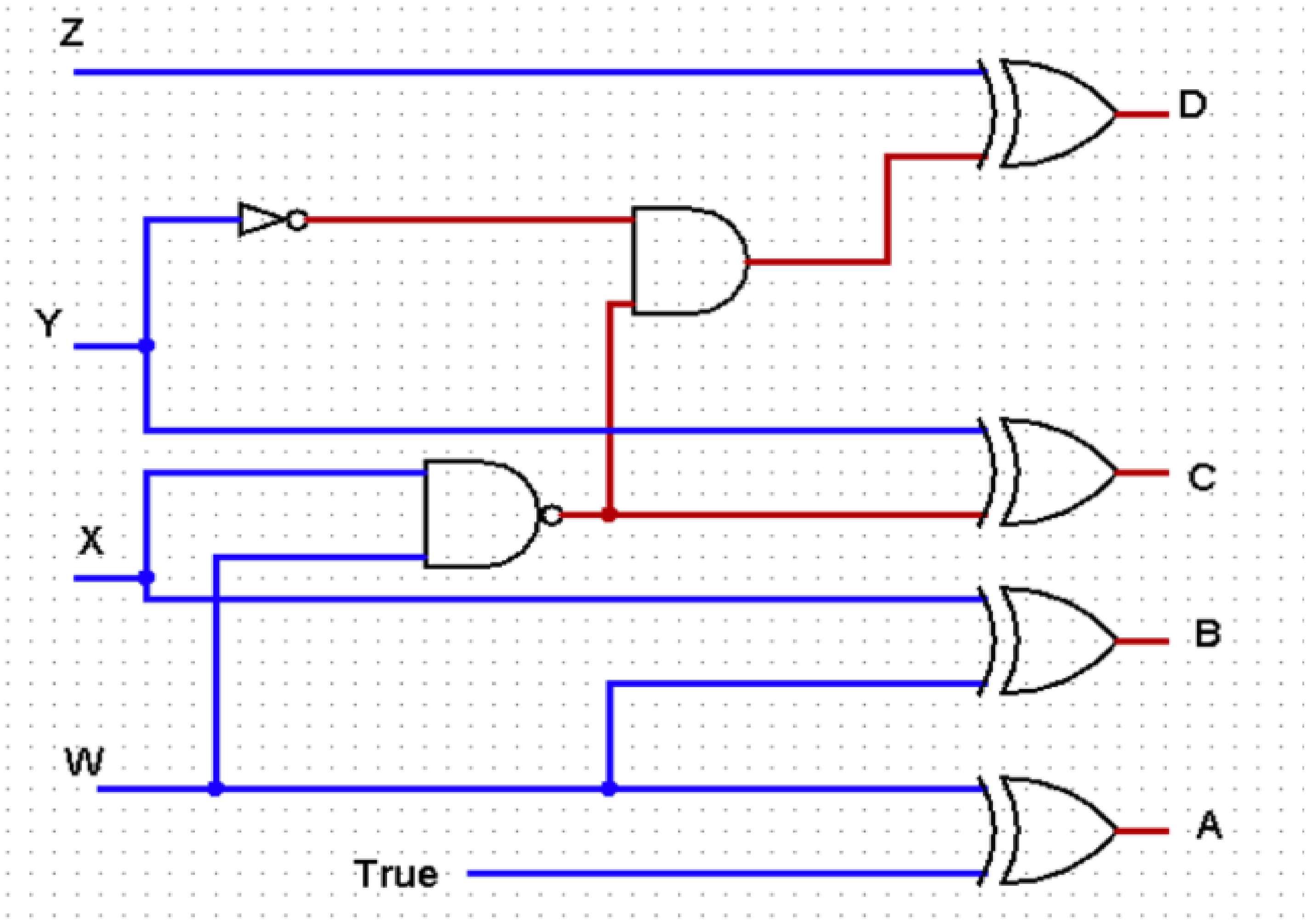


Figure : Excess-3 to BCD code converter

Faculty Video Links, Youtube & NPTEL Video Links and Online Courses Details

- Youtube/other Video Links
  - https://www.youtube.com/watch?v=2bcWI9zCMj4
  - https://nptel.ac.in/courses/117/106/117106086/

Daily Quiz

- A device which converts 2421 code into 8421 code is called:

  a) Decoder

  b) Code converter

  c) Multiplexer

  d) Encoder

- A code converter is a logic circuit that _______________
  a) Inverts the given input
  b) Converts into decimal number
  c) Converts data of one type into another type
  d) Converts to octal

- The binary representation of BCD number 00101001 (decimal 29) is
  a) 0011101
  b) 0110101
  c) 1101001
  d) 0101011

Old Questions

- Design excess 3 to binary code converter.
- Design 4 bit binary to 8,4,-2,-1 code converter.
- Implement Gray to Binary code converter.

Recap

- The code converters are used to convert the information into the code which we want. These are basically encoders and decoders which converts the data into an encoded form.
- There are various types of code converters :
  - Binary to Gray code converter
  - Gray to Binary code converter
  - Excess-3 to BCD
  - BCD to Excess-3 converter

Serial adder & ALU

Topic objective	Mapping with CO
To understand working of serial adder.	CO2
To understand the concept of ALU.	CO2

Serial adder & ALU

Prerequisite:

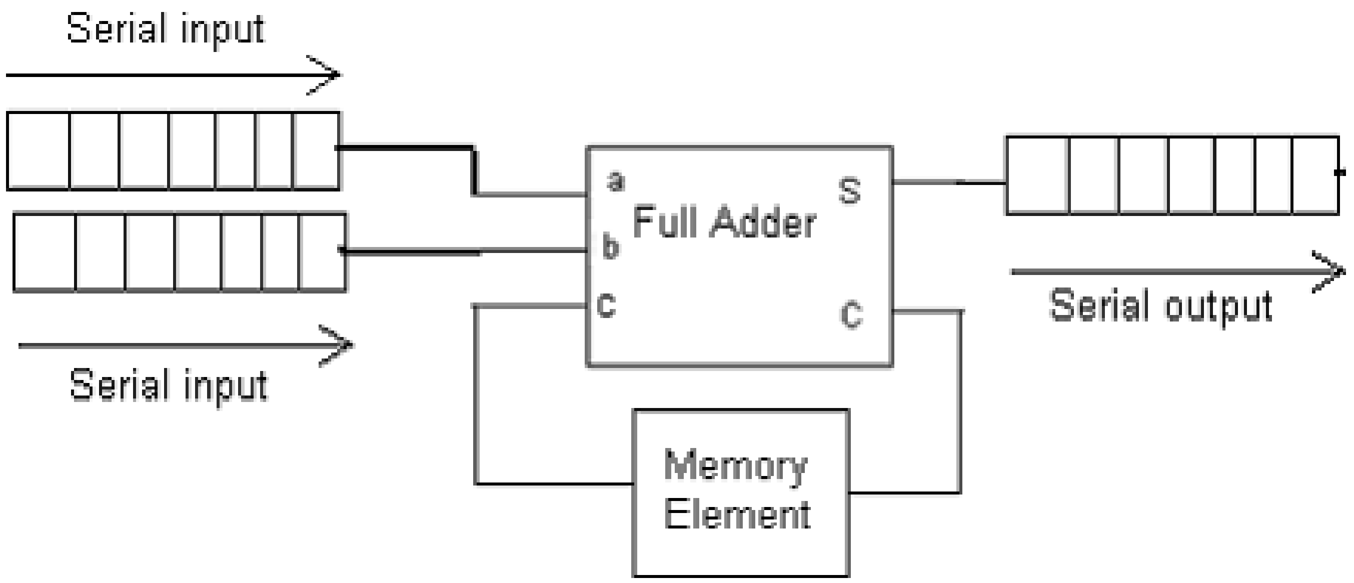
- Basic knowledge of logic gates.
- Basic knowledge of Boolean algebra.

Serial adder

Serial adder:

- The **serial binary adder** or **bit-serial adder** is a digital circuit that performs binary addition bit by bit. The serial full adder has three single-bit inputs for the numbers to be added and the carry in. There are two single-bit outputs for the sum and carry out. The carry-in signal is the previously calculated carry-out signal. The addition is performed by adding each bit, lowest to highest, one per clock cycle.
- Serial binary addition is done by a flip-flop and a full adder. The flip-flop takes the carry-out signal on each clock cycle and provides its value as the carry-in signal on the next clock cycle. After all of the bits of the input operands have arrived, all of the bits of the sum have come out of the sum output.

Serial adder



Serial adder

ALU

- In computing, an **arithmetic logic unit** (**ALU**) is a combinational digital circuit that performs arithmetic and bitwise operations on integer binary numbers.
- It is a fundamental building block of many types of computing circuits, including the central processing unit (CPU) of computers, FPUs, and graphics processing units (GPUs).
- The inputs to an ALU are the data to be operated on, called operands, and a code indicating the operation to be performed; the ALU's output is the result of the performed operation.
- In many designs, the ALU also has status inputs or outputs, or both, which convey information about a previous operation or the current operation, respectively, between the ALU and external status registers.

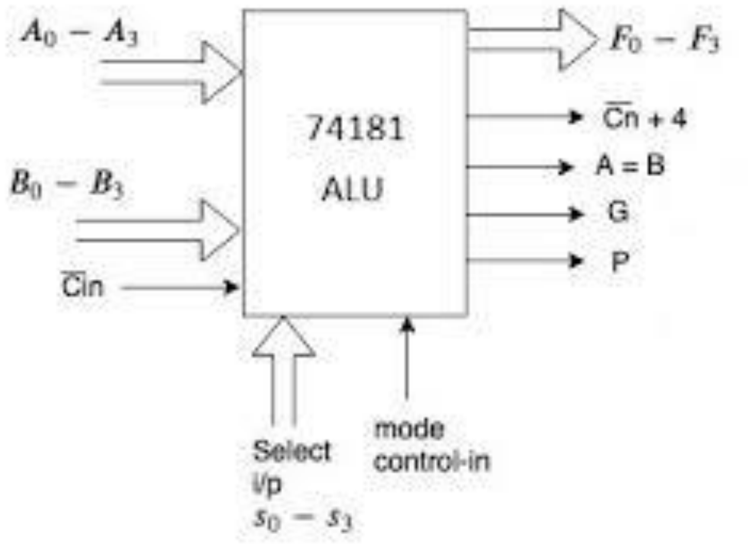
ALU

74181 ALU:

- The **74181** is a 4-bit arithmetic logic unit (ALU), implemented as a 74 series TTL integrated circuit.
- The first complete ALU on a single chip, it was used as the arithmetic/logic core in the CPUs of many historically significant minicomputers and other devices.
- The 4-bit wide ALU can perform all the traditional add / subtract / decrement operations with or without carry, as well as AND / NAND, OR / NOR, XOR, and shift. Many variations of these basic functions are available, for a total of 16 arithmetic and 16 logical operations on two four-bit words

ALU

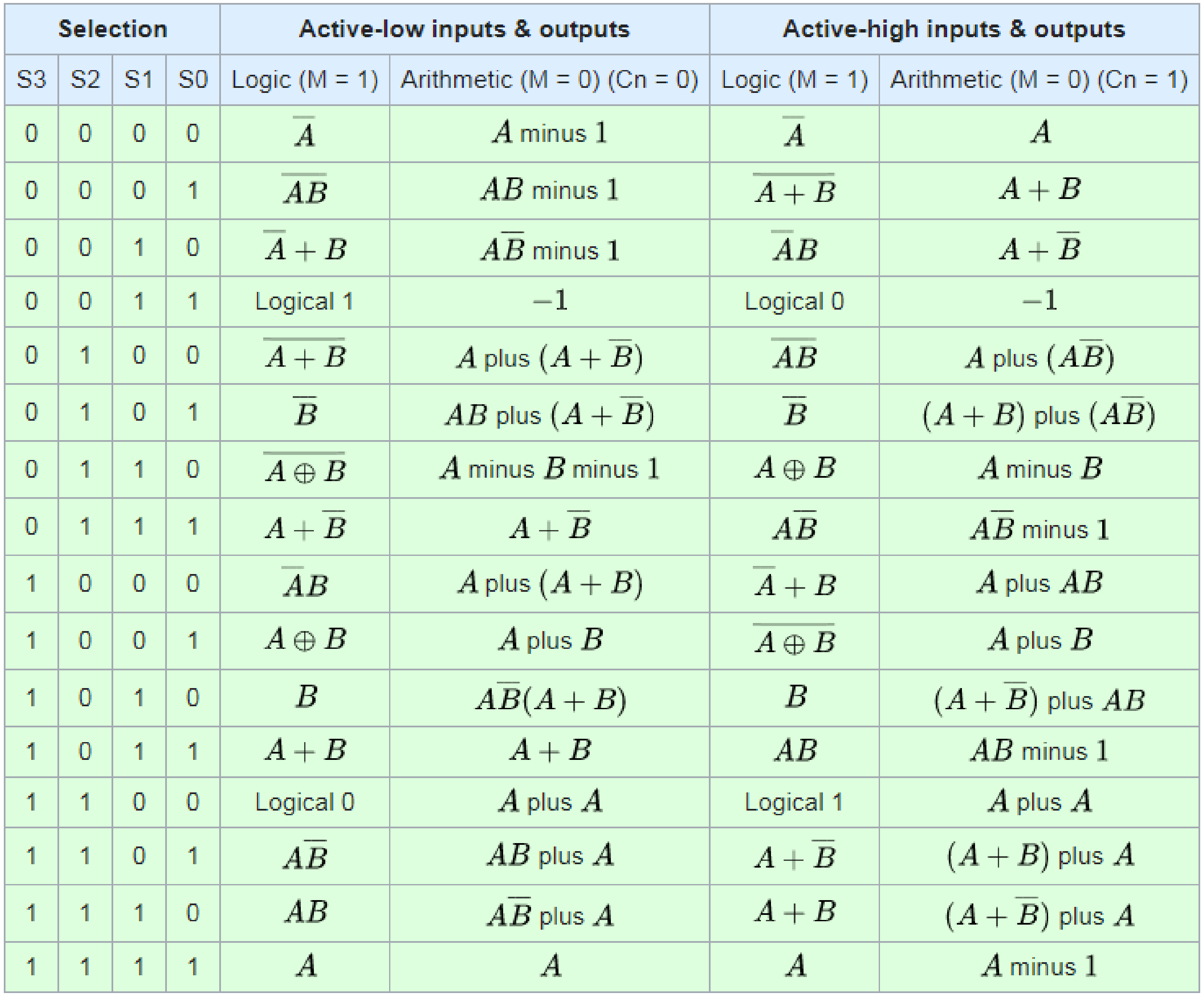
- **Pin diagram:**



- A & B are 4 bit data inputs and F is 4 bit data output.
- G is carry generate and P is carry propagate.
- S is select input used to select operation.
- M is mode control . M = 0 for Arithmetic operation and M = 1 for logic operation.
- Carry output is active low signal. For subtraction if it is 0 means result is positive and 1 means result is negative.

ALU

Function Table



Selection				Active-low inputs & outputs		Active-high inputs & outputs	
S3	S2	S1	S0	Logic (M = 1)	Arithmetic (M = 0) (Cn = 0)	Logic (M = 1)	Arithmetic (M = 0) (Cn = 1)
0	0	0	0	$\overline{A}$	A minus 1	$\overline{A}$	A
0	0	0	1	$\overline{AB}$	AB minus 1	$\overline{A+B}$	$A+B$
0	0	1	0	$\overline{A}+B$	$A\overline{B}$ minus 1	$\overline{A}B$	$A+\overline{B}$
0	0	1	1	Logical 1	-1	Logical 0	-1
0	1	0	0	$\overline{A+B}$	A plus $(A+\overline{B})$	$\overline{AB}$	A plus $(A\overline{B})$
0	1	0	1	$\overline{B}$	AB plus $(A+\overline{B})$	$\overline{B}$	$(A+B)$ plus $(A\overline{B})$
0	1	1	0	$\overline{A\oplus B}$	A minus B minus 1	$A\oplus B$	A minus B
0	1	1	1	$A+\overline{B}$	$A+\overline{B}$	$A\overline{B}$	$A\overline{B}$ minus 1
1	0	0	0	$\overline{A}B$	A plus $(A+B)$	$\overline{A}+B$	A plus AB
1	0	0	1	$A\oplus B$	A plus B	$\overline{A\oplus B}$	A plus B
1	0	1	0	B	$A\overline{B}(A+B)$	B	$(A+\overline{B})$ plus AB
1	0	1	1	$A+B$	$A+B$	AB	AB minus 1
1	1	0	0	Logical 0	A plus A	Logical 1	A plus A
1	1	0	1	$A\overline{B}$	AB plus A	$A+\overline{B}$	$(A+B)$ plus A
1	1	1	0	AB	$A\overline{B}$ plus A	$A+B$	$(A+\overline{B})$ plus A
1	1	1	1	A	A	A	A minus 1

Recap

- The **serial binary adder** or **bit-serial adder** is a digital circuit that performs binary addition bit by bit. The serial full adder has three single-bit inputs for the numbers to be added and the carry in. There are two single-bit outputs for the sum and carry out. The carry-in signal is the previously calculated carry-out signal.
- In computing, an **arithmetic logic unit (ALU)** is a combinational digital circuit that performs arithmetic and bitwise operations on integer binary numbers.

Daily Quiz

- What is a disadvantage of using a serial adder?
- ALU is the place where the actual executions of instructions take place during the processing operation.
  a) True
  b) False
- The ALU gives the output of the operations and the output is stored in the _________
  a) Memory Devices
  b) Registers
  c) Flags
  d) Output Unit
- Which adder has highest speed of operation?
  a) Serial adder
  b) Parallel adder
  c) Carry look ahead adder

Old Questions

- What is the use of ALU? Explain its working.
- Explain the operation of serial adder. Write its advantages over parallel adder.

Faculty Video Links, Youtube & NPTEL Video Links and Online Courses Details

- Youtube/other Video Links
  - https://www.youtube.com/watch?v=H073mLyvoUI&ab_channel=LearnByWatchElectronicsLearnByWatchElectronics
  - https://nptel.ac.in/courses/117/106/117106086/
  - https://www.youtube.com/watch?v=MJi2jHYM-EY&t=33s

Weekly Assignment

- How a serial adder is different from serial adder?
- Design a combinational circuit that converts a decimal digit from 8, 4, -2, -1 code to BCD.
- Write a short note on the followings:

a) 2-bit Magnitude Comparator

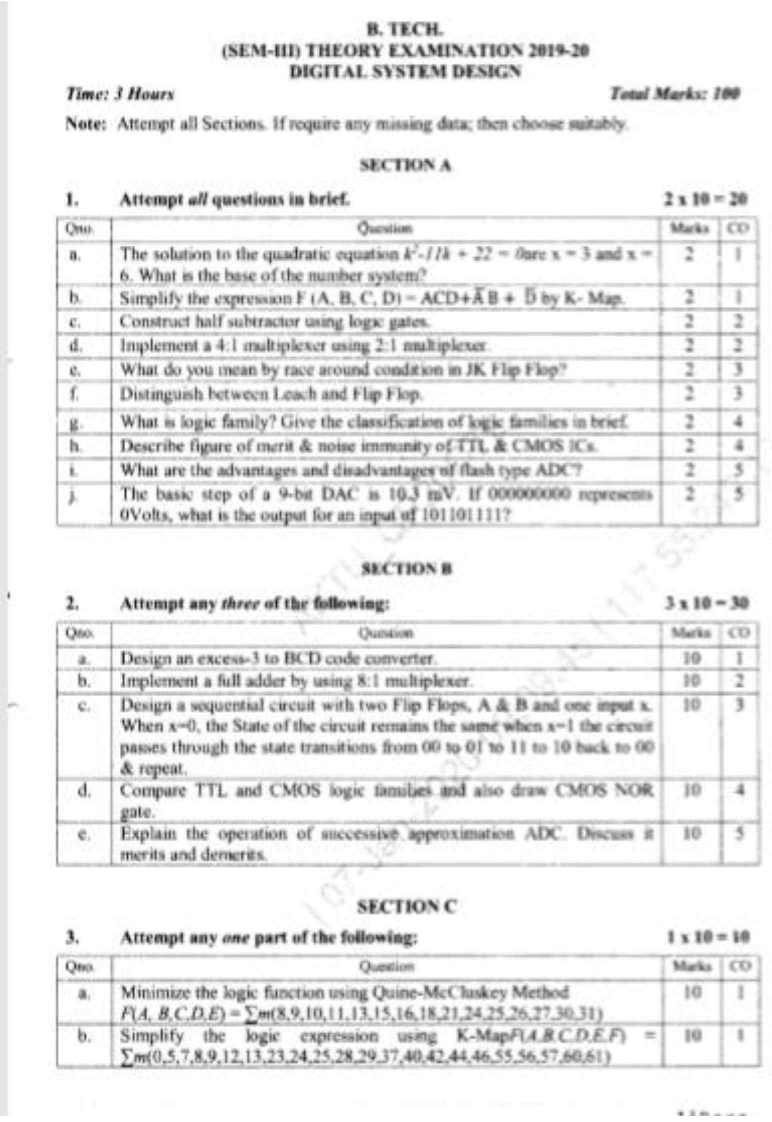
b) Priority Encoder

c) Serial Adder

d) 2- digit BCD Adder

- What is the use of ALU?
- Implement Gray to Binary code converter.

Old Question Papers



B. TECH.
(SEM-III) THEORY EXAMINATION 2019-20
DIGITAL SYSTEM DESIGN

Time: 3 Hours | *Total Marks: 100*

Note: Attempt all Sections. If require any missing data; then choose suitably.

SECTION A

1. **Attempt *all* questions in brief.** 2 x 10 = 20

Qno.	Question	Marks	CO
a.	The solution to the quadratic equation $k^2-11k+22=0$ are x = 3 and x = 6. What is the base of the number system?	2	1
b.	Simplify the expression F (A, B, C, D) = ACD+$\bar{A}$B + $\bar{D}$ by K- Map.	2	1
c.	Construct half subtractor using logic gates.	2	2
d.	Implement a 4:1 multiplexer using 2:1 multiplexer.	2	2
e.	What do you mean by race around condition in JK Flip Flop?	2	3
f.	Distinguish between Leach and Flip Flop.	2	3
g.	What is logic family? Give the classification of logic families in brief.	2	4
h.	Describe figure of merit & noise immunity of TTL & CMOS ICs.	2	4
i.	What are the advantages and disadvantages of flash type ADC?	2	5
j.	The basic step of a 9-bit DAC is 10.3 mV. If 000000000 represents 0Volts, what is the output for an input of 101101111?	2	5

SECTION B

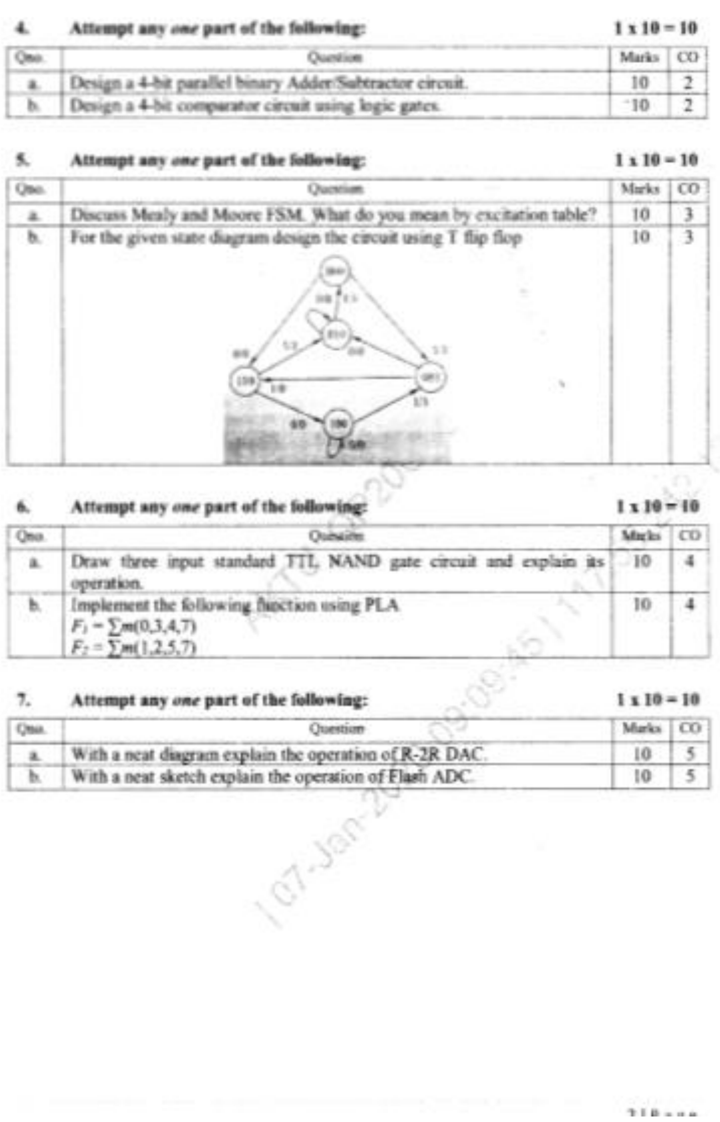
2. **Attempt any *three* of the following:** 3 x 10 = 30

Qno.	Question	Marks	CO
a.	Design an excess-3 to BCD code converter.	10	1
b.	Implement a full adder by using 8:1 multiplexer.	10	2
c.	Design a sequential circuit with two Flip Flops, A & B and one input x. When x=0, the State of the circuit remains the same when x=1 the circuit passes through the state transitions from 00 to 01 to 11 to 10 back to 00 & repeat.	10	3
d.	Compare TTL and CMOS logic families and also draw CMOS NOR gate.	10	4
e.	Explain the operation of successive approximation ADC. Discuss it merits and demerits.	10	5

SECTION C

3. **Attempt any *one* part of the following:** 1 x 10 = 10

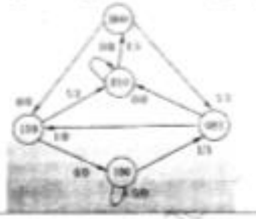
Qno.	Question	Marks	CO
a.	Minimize the logic function using Quine-McCluskey Method $F(A,B,C,D,E)=\sum m(8,9,10,11,13,15,16,18,21,24,25,26,27,30,31)$	10	1
b.	Simplify the logic expression using K-Map$F(A,B,C,D,E,F)=\sum m(0,5,7,8,9,12,13,23,24,25,28,29,37,40,42,44,46,55,56,57,60,61)$	10	1



4. **Attempt any *one* part of the following:** 1 x 10 = 10

Qno.	Question	Marks	CO
a.	Design a 4-bit parallel binary Adder/Subtractor circuit.	10	2
b.	Design a 4-bit comparator circuit using logic gates.	10	2

5. **Attempt any *one* part of the following:** 1 x 10 = 10

Qno.	Question	Marks	CO
a.	Discuss Mealy and Moore FSM. What do you mean by excitation table?	10	3
b.	For the given state diagram design the circuit using T flip flop 	10	3

6. **Attempt any *one* part of the following:** 1 x 10 = 10

Qno.	Question	Marks	CO
a.	Draw three input standard TTL NAND gate circuit and explain its operation.	10	4
b.	Implement the following function using PLA $F_1=\sum m(0,3,4,7)$ $F_2=\sum m(1,2,5,7)$	10	4

7. **Attempt any *one* part of the following:** 1 x 10 = 10

Qno.	Question	Marks	CO
a.	With a neat diagram explain the operation of R-2R DAC.	10	5
b.	With a neat sketch explain the operation of Flash ADC.	10	5

REFERENCE

- R.P. Jain, “Modern Digital Electronics,” Tata McGraw Hill, 4th edition, 2009.
- A. Anand Kumar, “Fundamental of Digital Circuits,” PHI 4th edition, 2018.
- W.H. Gothmann, “Digital Electronics- An Introduction to Theory and Practice,” PHI, 2nd edition, 2006.
- D.V. Hall, “Digital Circuits and Systems,” Tata McGraw Hill, 1989.
- A. K. Singh, “Foundation of Digital Electronics & Logic Design,” New Age Int. Publishers.
- Subrata Ghosal, “Digital Electronics,” Cengage publication, 2nd edition, 2018

Thank You