Half adder

- A half-adder is a combinational circuit with two binary inputs (augund and addend bits) and two binary outputs (sum and carry bits).
- It adds the two inputs (single bit words A and B) and produces the sum (S) and the carry (C) bits.
- The truth table of a half-adder are shown below:

Inputs		Outputs	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

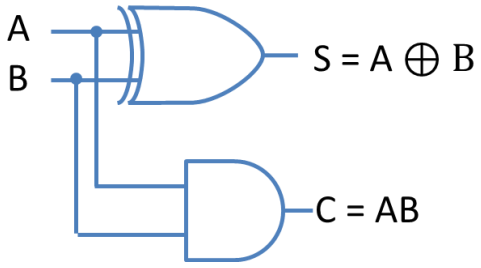
- The Sum (S) is the X-OR of A and B (It represent the LSB of the sum). Therefore,

$$S = AB' + BA' = A \oplus B$$

- The carry (C) is the AND of A and B (It is 0 unless both the inputs are 1). Therefore,

$$C = AB$$

- A half-adder can, therefore, be realized by using one X-OR gate and one AND gate as shown in figure below.



Full adder

- A full-adder is a combinational circuit that adds two bits and a carry and outputs a sum bit and a carry bit.
- When we want to add two binary numbers, each having two or more bits, the LSBs can be added by using a half-adder.
- The carry resulted from the addition of the LSBs is carried over to the next significant column and added to the two bits in that column.
- The full-adder adds the bits A and B and the carry from the previous column called the carry-in C_{in} and outputs the sum bit S and the carry bit called the carry-out C_{out}.
- The variable S gives the value of the least significant bit of the sum.
- The variable C_{out} gives the output carry.
- The truth table of a full-adder are shown in figure below.

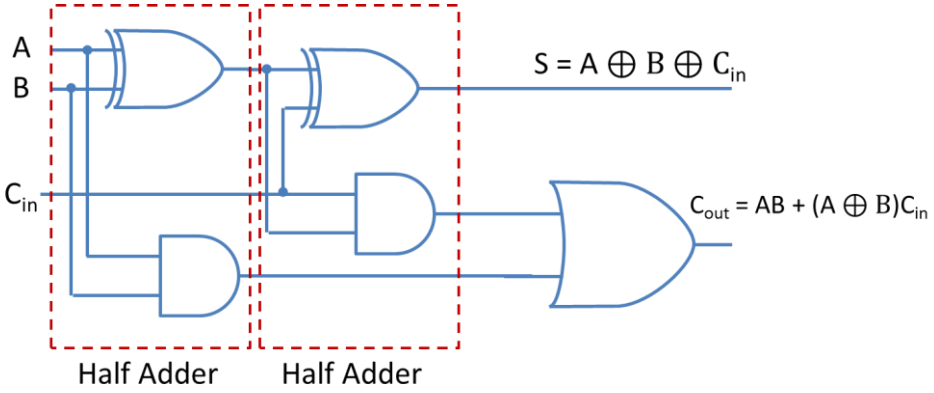
Inputs			Outputs	
A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

- The eight rows under the input variables designate all possible combinations of 1s and 0s that these variables may have.
- When all the bits are 0s, the output is 0.
- The S output is equal to 1 when only 1 input is equal to 1 or when all the inputs are equal to 1.
- The C_{out} has a carry of 1 if two or three inputs are equal to 1.
- From the truth table, a circuit that will produce the correct sum and carry bits in response to every possible combination of A, B, and C_{in} is described by

$$\begin{aligned} S &= A'B'C_{in} + A'BC_{in}' + AB'C_{in}' + ABC_{in} \\ &= (AB' + A'B)C_{in}' + (AB + A'B')C_{in} \\ &= (A \oplus B)C_{in}' + (A \oplus B)'C_{in} \\ &= A \oplus B \oplus C_{in} \end{aligned}$$

$$\begin{aligned} C_{out} &= A'BC_{in} + AB'C_{in} + ABC_{in}' + ABC_{in} \\ &= AB + (A \oplus B)C_{in} \end{aligned}$$

- The sum term of the full-adder is the X-OR of A, B and C_{in}, i.e., the sum bit is the modulo sum of the data bits in that column and the carry from the previous column.
- The logic diagram of the full-adder using two X-OR gates and two AND gates (i.e., two half-adders) and one OR gate is shown in figure below.



Half-Subtractor

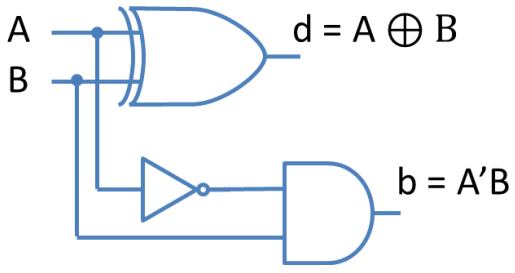
- A half-subtractor is a combinational circuit that subtracts one bit from the other and produces the difference.
- It also has an output to specify if a 1 has been borrowed.
- It is used to subtract the LSB of the subtrahend from the LSB of the minuend when one binary number is subtracted from the other.
- A half-subtractor is a combinational circuit with two inputs A and B and two outputs *d* and *b*.
- *d* indicates the difference and *b* is the output signal generated that informs the next stage that a 1 has been borrowed.
- We know that, when a bit B is subtracted from another bit A, a difference bit (d) and a borrow bit (b) result according to the rules given as follows.

Inputs		Outputs	
A	B	d	b
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

- A circuit that produces the correct difference and borrow bits in response to every possible combination of the two 1-bit numbers is, therefore, described by

$$d = AB' + BA' = A \oplus B \text{ and } d = A'B$$

- That is, the difference bit is obtained by X-ORing the two inputs, and the borrow bit is obtained by ANDing the complement of the minuend with the subtrahend.
- Figure below shows logic diagrams of a half-subtractor.



Full Subtractor

- The half-subtractor can be used only for LSB subtraction.
- If there is a borrow during the subtraction of the LSBs, it affects the subtraction in the next higher column; the subtrahend bit is subtracted from the minuend bit, considering the borrow from that column used for the subtraction in the preceding column.
- Such a subtraction is performed by a full-subtractor.
- It subtracts one bit (B) from another bit (A), when already there is a borrow b_i from this column for the subtraction in the preceding column, and outputs the difference bit (d) and the borrow bit (b) required from the next column.
- So a full-subtractor is a combinational circuit with three inputs (A, B, bi) and two outputs d and b.
- The 1s and 0s for the output variables are determined from the subtraction of A - B - b_i.
- The truth table of a full-subtractor are shown in figure.

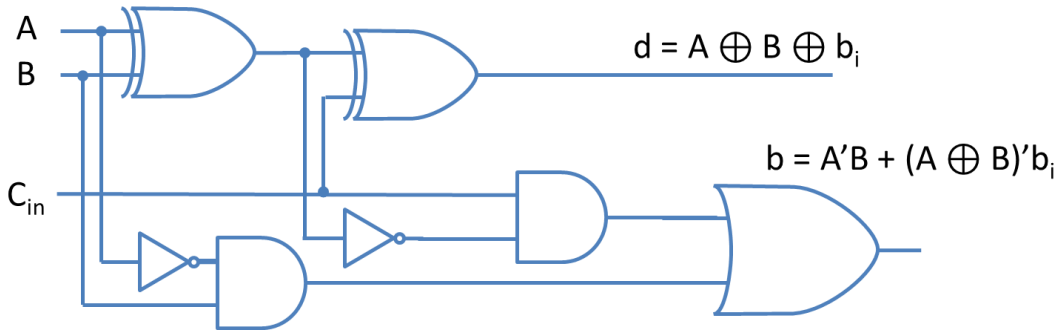
Inputs			Outputs	
A	**B**	**b_i**	**d**	**b**
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

- From the truth table, a circuit that will produce the correct difference and borrow bits in response to every possible combination of A, B, and b_i is described by

$$\begin{aligned} d &= A'B'b_i + A'Bb_i' + AB'b_i' + ABb_i \\ &= (AB' + A'B)b_i' + (AB + A'B')b_i \\ &= (A \oplus B)b_i' + (A \oplus B)'b_i \\ &= A \oplus B \oplus b_i \end{aligned}$$

$$\begin{aligned} b &= A'B'b_i + A'Bb_i' + A'Bb_i + ABb_i \\ &= A'B(b_i + b_i') + (AB + A'B')b_i \\ &= A'B + (A \oplus B)'b_i \end{aligned}$$

- A full-subtractor can, therefore, be realized using X-OR gates as shown below.



Design BCD to Excess-3 code converter circuit.

- BCD means 8421 BCD.
- The 4-bit input BCD code (B4 B3 B2 B1) and the corresponding output XS-3 code (X4 X3 X2 X1) numbers are shown in the conversion table in figure.

8421 code				XS-3 code			
B_4	B_3	B_2	B_1	X_4	X_3	X_2	X_1
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

- The input combinations 1010, 1011, 1100, 1101, 1110, and 1111 are invalid in BCD. So they are treated as don't cares.
- From the above truth table, function can be realized as follows:

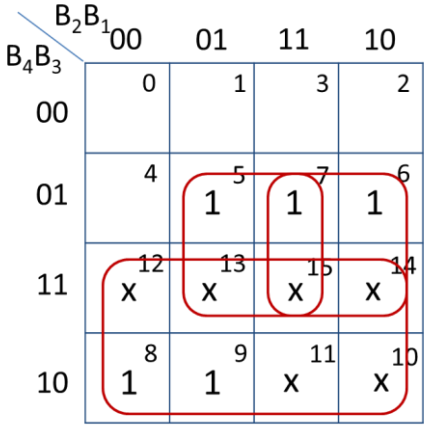
$X_4 = \Sigma\, m(5, 6, 7, 8, 9) + d(10, 11, 12, 13, 14, 15)$

$X_3 = \Sigma\, m(1, 2, 3, 4, 9) + d(10, 11, 12, 13, 14, 15)$

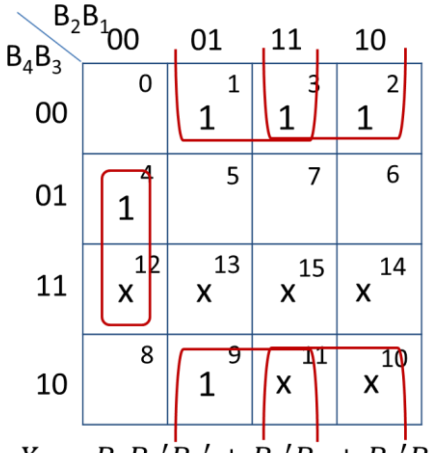
$X_2 = \Sigma\, m(0, 3, 4, 7, 8) + d(10, 11, 12, 13, 14, 15)$

$X_1 = \Sigma\, m(0, 2, 4, 6, 8) + d(10, 11, 12, 13, 14, 15)$

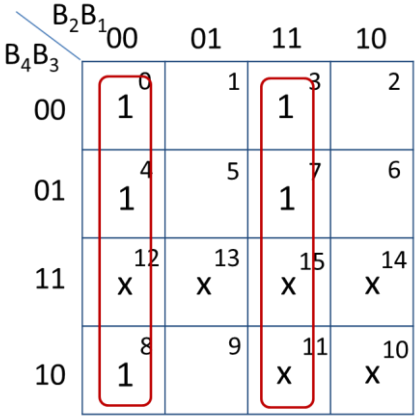
- Drawing K-maps for the outputs X_4, X_3, X_2, and X_1 in terms of the inputs B_4, B_3, B_2, and B_1 and simplifying them, as shown.



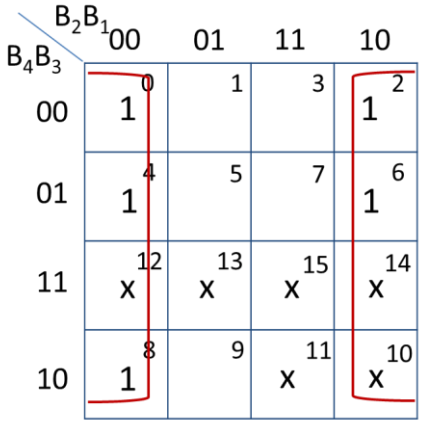
$X_4 = B_4 + B_3B_2 + B_3B_1$



$X_3 = B_3B_2'B_1' + B_3'B_1 + B_3'B_2$



$X_2 = B_2'B_1' + B_2B_1$



$X_2 = B_1'$

- The minimal expressions are

$X_4 = B_4 + B_3B_2 + B_3B_1$
$X_3 = B_3B_2'B_1' + B_3'B_1 + B_3'B_2$
$X_2 = B_2'B_1' + B_2B_1$
$X_1 = B_1'$

Design 4 bit binary to gray code converter

- The input to the 4-bit binary-to-Gray code converter circuit is a 4-bit binary and the output is a 4-bit Gray code.
- There are 16 possible combinations of 4-bit binary input and all of them are valid. Hence no don't cares.
- The 4-bit binary and the corresponding Gray code are shown in the conversion table below.

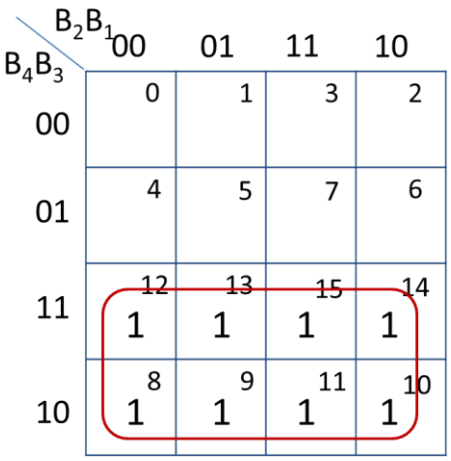
4-bit binary				4-bit Gray			
B_4	B_3	B_2	B_1	G_4	G_3	G_2	G_1
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1

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

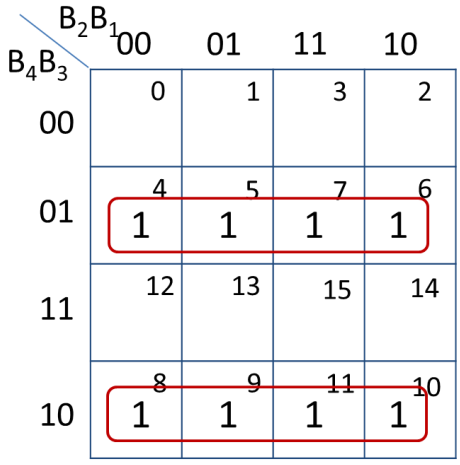
- From the conversion table, we observe that the expressions for the outputs G4, G3, G2, and G1 are as follows:

G4 = Σ m(8, 9, 10, 11, 12, 13, 14, 15)
G3 = Σ m(4, 5, 6, 7, 8, 9, 10, 11)
G2 = Σ m(2, 3, 4, 5, 10, 11, 12, 13)
G1 = Σ m(1, 2, 5, 6, 9, 10, 13, 14)

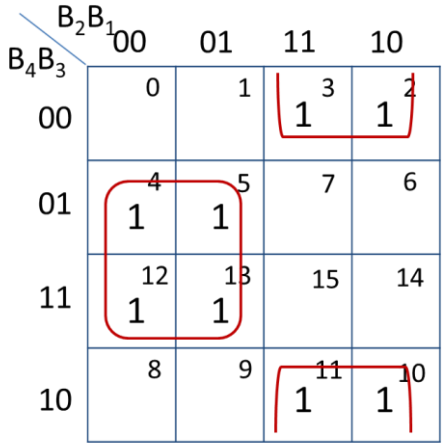
- The K-maps for G4, G3, G2, and G1 and their minimization are shown in figure below.



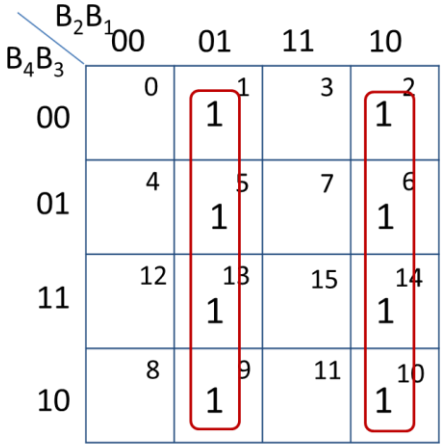
$G_4 = B_4$



$G_3 = B_4 \oplus B_3$



$G_2 = B_3 \oplus B_2$



$G_1 = B_2 \oplus B_1$

- The minimal expressions for the outputs obtained from the K-map are:

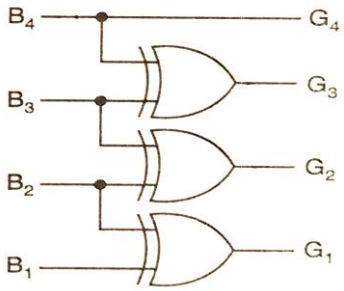
G_4 = B4

G3 = B4'B3 + B4B3' = B4 ⊕ B3

G2 = B3'B2 + B3B2' = B3 ⊕ B2

G1 = B2'B1 + B2B1' = B2 ⊕ B1

- Logic diagram for the above is as follows.



Design a circuit for 2-bit magnitude comparator.

- The logic for a 2-bit magnitude comparator: Let the two 2-bit numbers be A = A_1A_0 and B = B_1B_0.
  1. If A_1 = 1 and B_1 = 0, then A > B or
  2. If A_1 and B_1 coincide and A_0 = 1 and B_0 = 0, then A > B. So the logic expression for A > B is

$$A > B : G = A_1B_1' + (A1 \odot B1)\, A_0B_0'$$

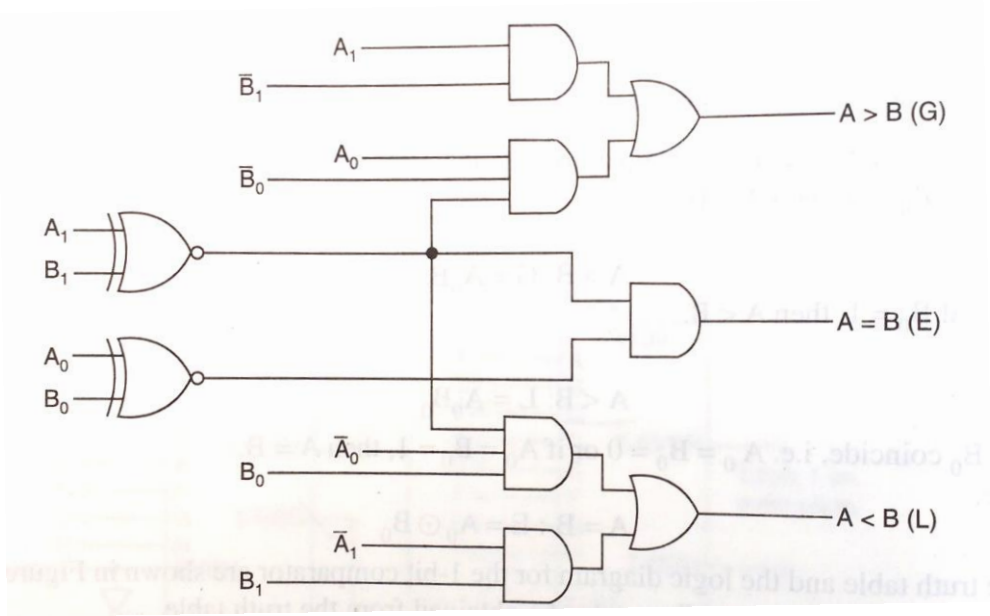
  1. If A_1 = 0 and B_1 = 1, then A < B or
  2. If A_1 and B_1 coincide and A_0 = 0 and B_0 = 1, then A < B. So the expression for A < B is

$$A < B : L = A_1'B_1 + (A_1 \odot B_1)\, A_0'B_0$$

  If A_1 and B_1 coincide and if A_0 and B_0 coincide then A = B. So the expression for A = B is

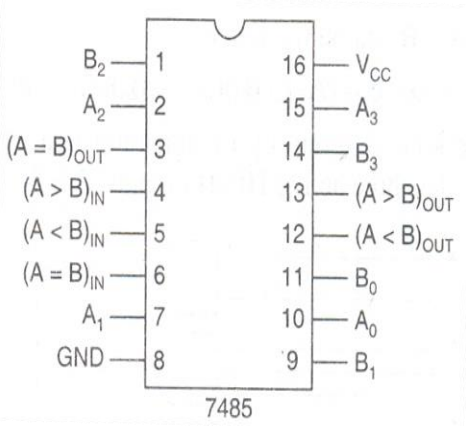
$$A = B : E = (A_1 \odot B_1)(A_0 \odot B_0)$$

- The logic diagram for a 2-bit comparator is as shown below:

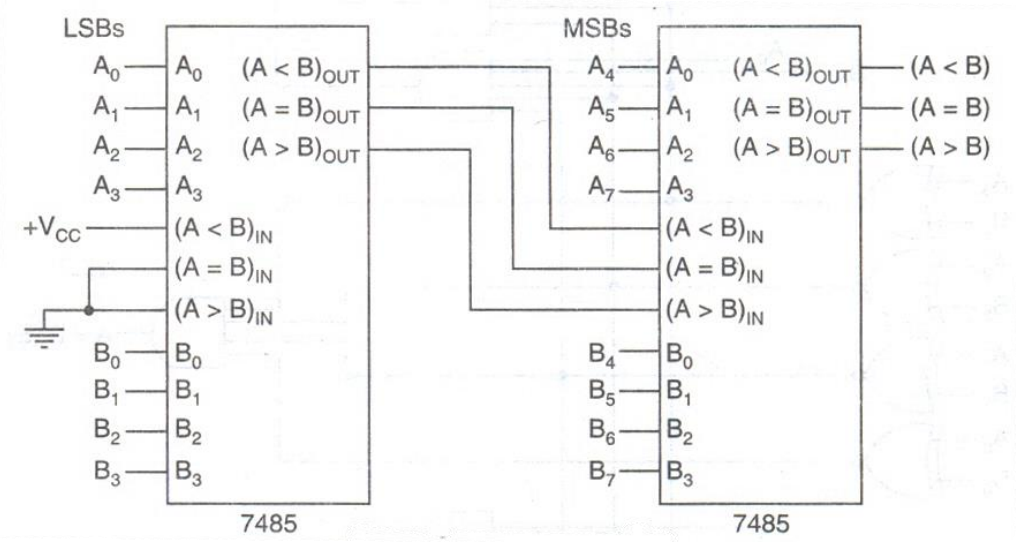


Draw & explain in brief pin diagram of 7485 four-bit magnitude comparator.

- Figure below shows the pin diagram of IC 7485, a 4-bit comparator.



- Pins labelled $(A < B)_{IN}$, $(A = B)_{IN}$, and $(A > B)_{IN}$ are used for cascading.
- Figure shows how two 4-bit comparator are cascaded to perform 8-bit comparisons.



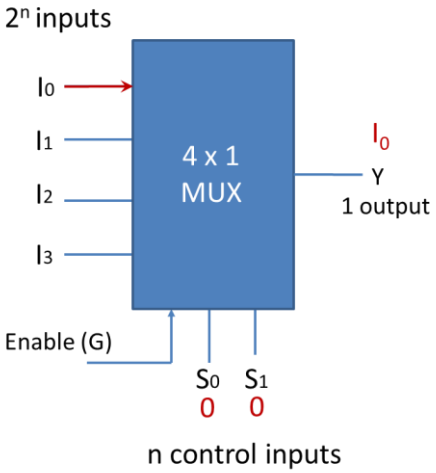
- The $(A < B)_{OUT}$, $(A = B)_{OUT}$ and $(A > B)_{OUT}$ outputs from the lower order comparator used for the least significant 4 bits, are connected to the $(A < B)_{IN}$, $(A = B)_{IN}$, and $(A > B)_{IN}$ inputs of the higher-order comparator.
- Note that, $(A < B)_{IN}$ input of the lower order comparator is connected to V_{CC}, and $(A = B)_{IN}$ and $(A > B)_{IN}$ inputs of the lower order comparator are connected to ground.

What is multiplexer? With logic circuit and function table explain the working of 4 to 1 line multiplexer.

- A multiplexer (MUX) is a device that allows digital information from several sources to be routed onto a single line for transmission over that line to a common destination.
- Consider an integer 'm', which is constrained by the following relation:

  $m = 2^n$, where m and n are both integers.
- A **m-to-1** Multiplexer has
  - m Inputs: $I_0, I_1, I_2, \ldots\ldots\ldots\ldots I_{(m-1)}$
  - One Output: Y
  - n Control inputs: $S_0, S_1, S_2, \ldots\ldots S_{(n-1)}$
  - One (or more) Enable input(s)

  such that Y may be equal to one of the inputs, depending upon the control inputs.
- The block diagram of 4 x 1 multiplexer is as follows.



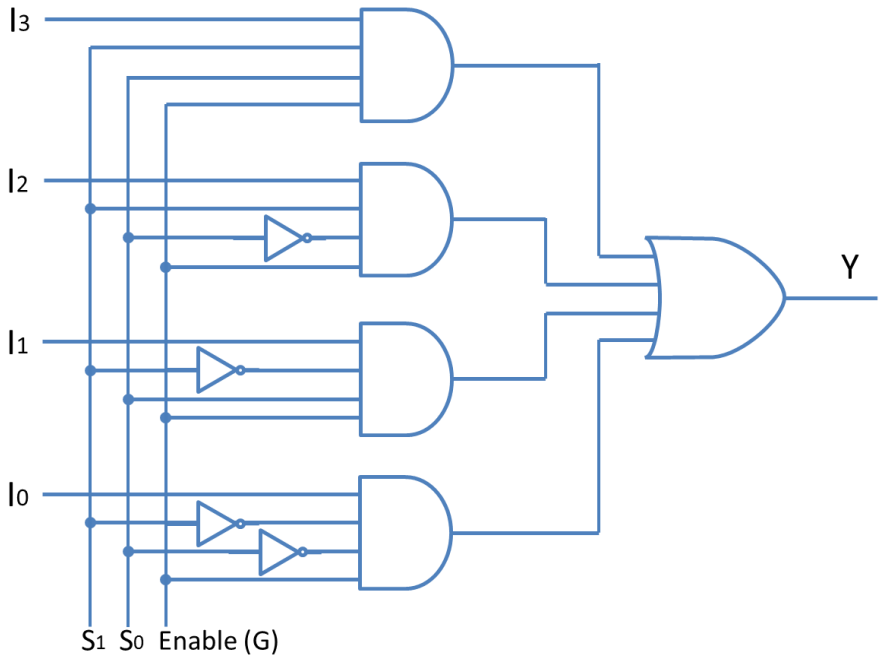
- The function table for the 4 x 1 multiplexer can be stated as below.

Select Inputs		Output
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

- The following logic function describes the above function table.

$$Y = S_1'S_0'I_0 + S_1'S_0I_1 + S_1S_0'I_2 + S_1S_0I_3$$

- The following figure describes the logic circuit for 4 x 1 multiplexer.



- Applications of Multiplexer is as follows:
  1. Logic function generation
  2. Data selection
  3. Data routing
  4. Operation sequencing
  5. Parallel-to-serial conversion
  6. Waveform generation

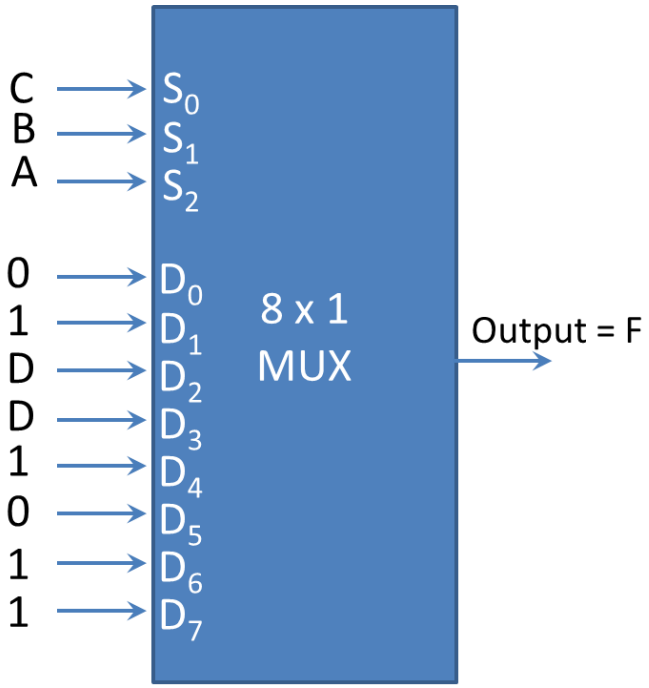
Implement following Boolean function using 8 : 1 multiplexer.

F(A,B,C,D) = Σ(2,3,5,7,8,9,12,13,14,15)

- The truth table for the above function is as follows:

S_2	S_1	S_0	D	F	
A	B	C			
0 0	0 0	0 0	0 1	0 0	F = 0
0 0	0 0	1 1	0 1	1 1	F = 1
0 0	1 1	0 0	0 1	0 1	F = D
0 0	1 1	1 1	0 1	0 1	F = D
1 1	0 0	0 0	0 1	1 1	F = 1
1 1	0 0	1 1	0 1	0 0	F = 0
1 1	1 1	0 0	0 1	1 1	F = 1
1 1	1 1	1 1	0 1	1 1	F = 1

- Based on the above truth table, the logic function can be implemented using 8 x 1 Multiplexer as follows:



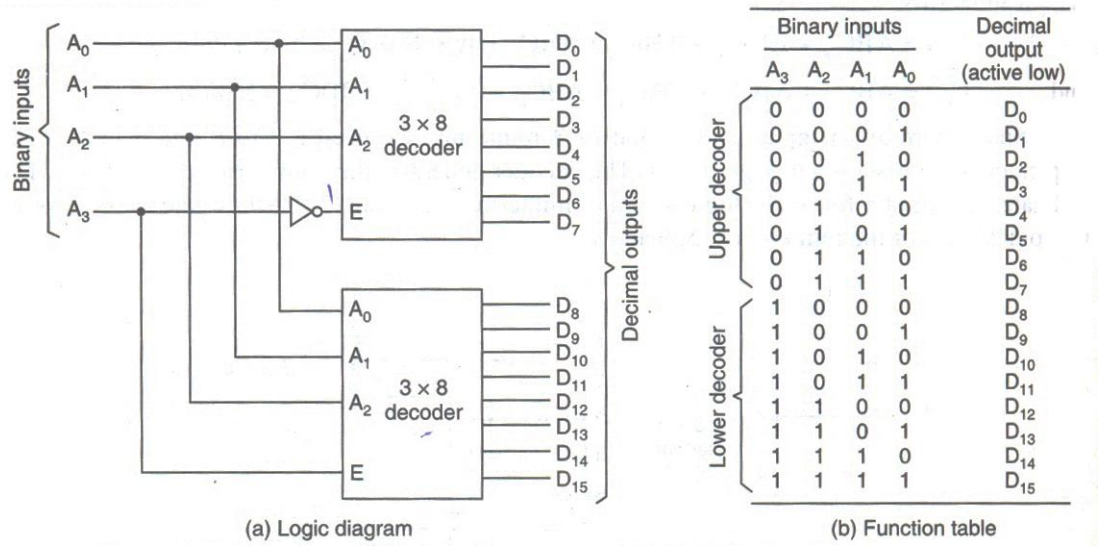
Exercise

Implement following Boolean function using 8 : 1 multiplexer.

1) F(A, B, C) = ∑m (1, 3, 5, 6)
2) F(A, B, C) = ∑m (1,2,4,7)
3) F(A, B, C, D) = ∑m (0, 1, 3, 5, 7, 11, 13, 14, 15)
4) F(A, B, C, D) = ∑m (0,1,2,3,5,8,9,11,14)
5) F(A, B, C, D) = ∑m (0,1,3,4,8,9,15)

Design 4 x 16 decoder using two 3 x 8 decoder.

- Decoders with enable inputs can be connected together to form a larger decoder circuit.
- Figure shows the arrangement for using two 3-to-8 decoders, to obtain a 4-to-16 decoder.
- The most significant input bit A_3 is connected through an inverter to E on the upper decoder (for D_0 through D_7) and directly to E on the lower decoder (for D_8 through D_{15}).
- Thus, when A_3 is LOW, the upper decoder is enabled and the lower decoder is disabled.
- The bottom decoder outputs all 0s, and top 8 outputs generate minterms.
- When A_3 is HIGH, the lower decoder is enabled and the upper decoder is disabled. The bottom decoder outputs generate minterms 1000 to 1111 while the outputs of the top decoder are all 0s.

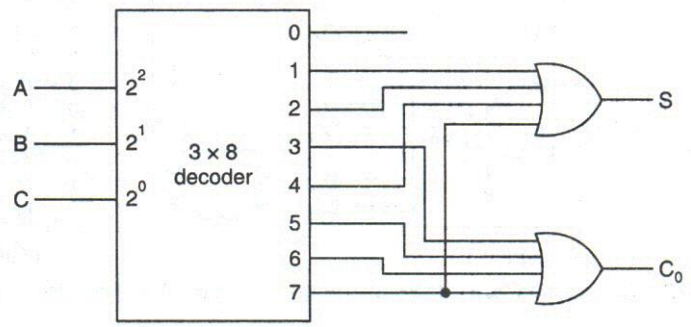


Explain full adder and design a full adder circuit using 3 to 8 decoder and two OR gates.

- A full-adder is a combinational circuit that adds two bits and a carry and outputs a sum bit and a carry bit.
- When we want to add two binary numbers, each having two or more bits, the LSBs can be added by using a half-adder.
- The carry resulted from the addition of the LSBs is carried over to the next significant column and added to the two bits in that column.
- The full-adder adds the bits A and B and the carry from the previous column called the carry-in C_{in} and outputs the sum bit S and the carry bit called the carry-out C_{out}.
- The variable S gives the value of the least significant bit of the sum.
- The variable C_{out} gives the output carry.
- The truth table of a full-adder are shown in figure below.

Inputs			Outputs	
A	**B**	**C_{in}**	**S**	**C_{out}**
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

- The eight rows under the input variables designate all possible combinations of 1s and 0s that these variables may have.
- When all the bits are 0s, the output is 0.
- The S output is equal to 1 when only 1 input is equal to 1 or when all the inputs are equal to 1.
- The C_{out} has a carry of 1 if two or three inputs are equal to 1.
- From the truth table, a circuit that will produce the correct sum and carry bits in response to every possible combination of A, B, and C_{in} is described by
- The function S and C_{out} can be represented in form of minterms as,
  S = Σ m(1, 2, 4, 7)
  C_{out} = Σ m(3, 5, 6, 7)
- The full adder can be implemented using decoder is as follows.



Implement Full Subtractor Circuit with the help of Decoder and logic gates.

- The half-subtractor can be used only for LSB subtraction.
- If there is a borrow during the subtraction of the LSBs, it affects the subtraction in the next higher column; the subtrahend bit is subtracted from the minuend bit, considering the borrow from that column used for the subtraction in the preceding column.
- Such a subtraction is performed by a full-subtractor.
- It subtracts one bit (B) from another bit (A), when already there is a borrow b_i from this column for the subtraction in the preceding column, and outputs the difference bit (d) and the borrow bit (b) required from the next column.
- So a full-subtractor is a combinational circuit with three inputs (A, B, bi) and two outputs d and b.
- The 1s and 0s for the output variables are determined from the subtraction of A - B - b_i.
- The truth table of a full-subtractor are shown in figure.

Inputs			Outputs	
A	**B**	**b_i**	**d**	**b**
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

- From the truth table, a circuit that will produce the correct difference and borrow bits in response to every possible combination of A, B, and b_i is described by
- The function d and b can be represented in form of minterms as,
  d = Σ m(1, 2, 4, 7)
  b = Σ m(1, 2, 3, 7)
- The full subtractor can be implemented using decoder is as follows.