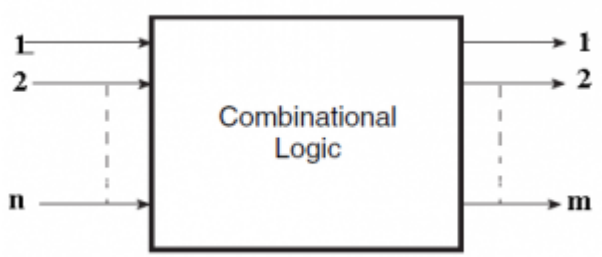
UNIT 3 COMBINATIONAL LOGIC

Introduction to combinational circuits:

A combinational circuit is the digital logic circuit in which the output depends on the combination of inputs at that point of time with total disregard to the past state of the inputs. The digital logic gate is the
building block of combinational circuits. The function implemented by combinational circuit is depend upon the Boolean expressions. On the other hand, sequential logic circuits, consists of both logic gates and memory elements such as flip-flops. Figure below shows the combinational circuit having n inputs and and m outputs. The n number of inputs shows that there are 2^n possible combinations of bits at the input. Therefore, the output is expressed in terms m Boolean expressions.



Analysis Procedure

- **To obtain the output Boolean functions from a logic diagram, proceed as follows:**

1. **Label all gate outputs that are a function of input variables with arbitrary symbols. Determine the Boolean functions for each gate output.**
2. **Label the gates that are a function of input variables and previously labeled gates with other arbitrary symbols. Find the Boolean functions for these gates.**
3. **Repeat the process outlined in step 2 until the outputs of the circuit are obtained.**
4. **By repeated substitution of previously defined functions, obtain the output Boolean functions in terms of input variables.**

   Example:

   $F_2 = AB + AC + BC$; $T_1 = A + B + C$; $T_2 = ABC$; $T_3 = F_2'T_1$;

   $F_1 = T_3 + T_2$

$F_1 = T_3 + T_2 = F_2'T_1 + ABC = A'BC' + A'B'C + AB'C' + ABC$

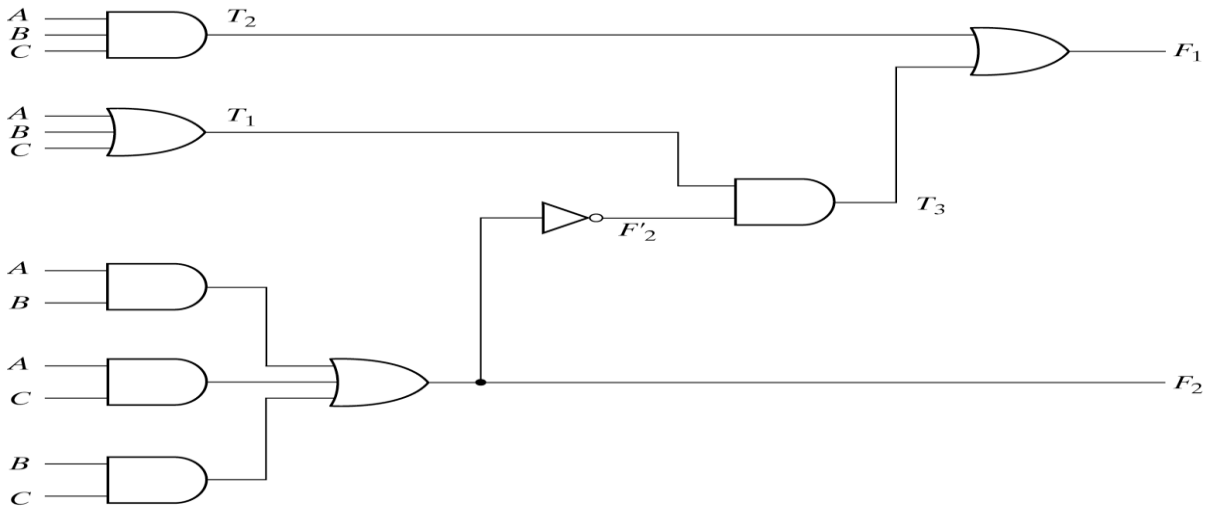


Fig. 4-2 Logic Diagram for Analysis Example

Derive truth table from logic diagram :

- **We can derive the truth table in Table 4-1 by using the circuit of Fig.4-2.**

Table 4-1
Truth Table for the Logic Diagram of Fig. 4-2

A	B	C	F_2	F_2	T_1	T_2	T_3	F_1
0	0	0	0	1	0	0	0	0
0	0	1	0	1	1	0	1	1
0	1	0	0	1	1	0	1	1
0	1	1	1	0	1	0	0	0
1	0	0	0	1	1	0	1	1
1	0	1	1	0	1	0	0	0
1	1	0	1	0	1	0	0	0
1	1	1	1	0	1	1	0	1

Design procedure :

1. **Table4-2 is a Code-Conversion example, first, we can list the relation of the BCD and Excess-3 codes in the truth table.**

Table 4-2
Truth Table for Code-Conversion Example

Input BCD				Output Excess-3 Code			
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

Karnaugh map:

For each symbol of the Excess-3 code, we use 1's to draw the map for simplifying Boolean function :

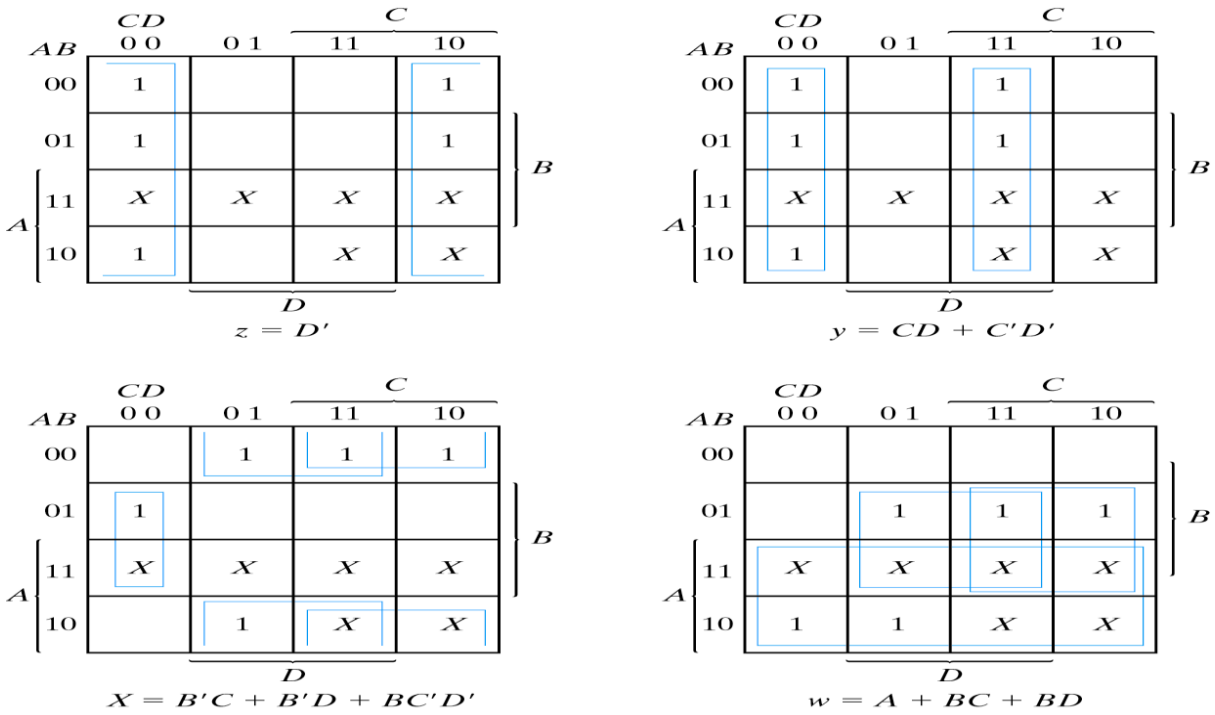


Fig. 4-3 Maps for BCD to Excess-3 Code Converter

Circuit implementation:

z = D'; y = CD + C'D' = CD + (C + D)'

x = B’C + B’D + BC’D’ = B’(C + D) + B(C + D)’

w = A + BC + BD = A + B(C + D)

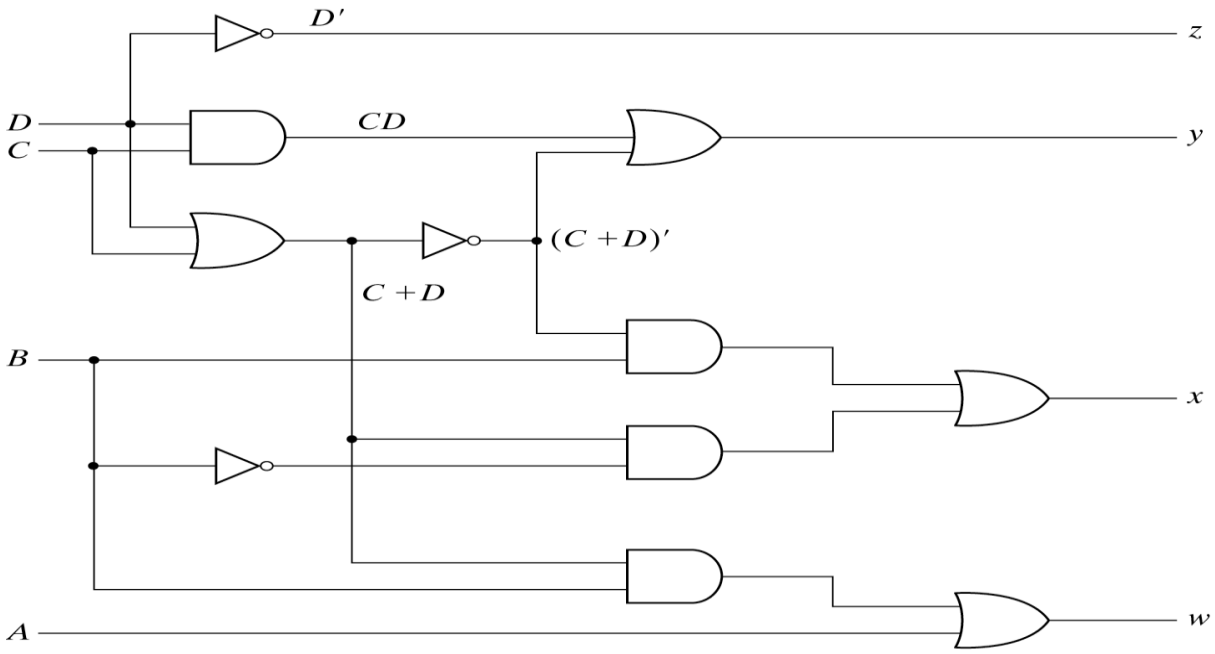


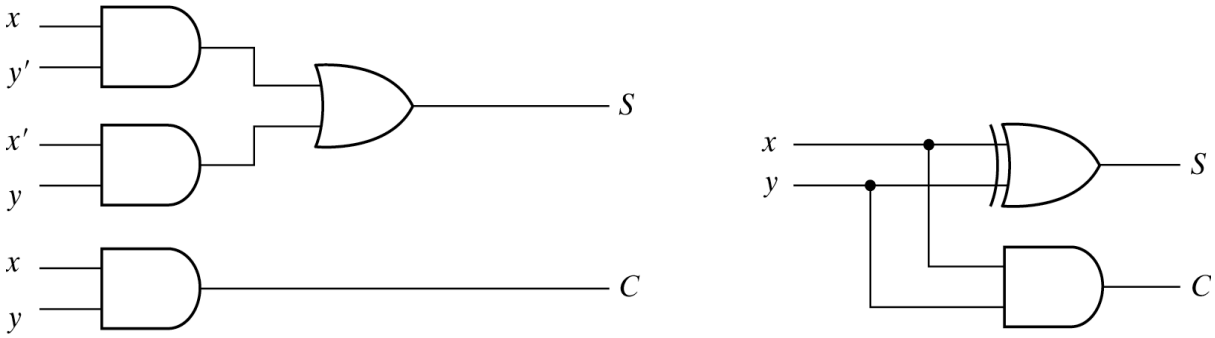
Fig. 4-4 Logic Diagram for BCD to Excess-3 Code Converter

Binary Adder-Subtractor:

- HALF-ADDER: **a combinational circuit that performs the addition of two bits is called a half adder.**

Half-adder is used to add two bits. Therefore, half-adder has two inputs and two outputs, with SUM and CARRY. Figure shows the truth table of a half-adder. The Boolean expressions for SUM and CARRY are,

SUM = AB’+A’B
CARRY = AB
These expressions shows`that, SUM output is EX-OR gate and the CARRY output is AND gate. Figure shows the be implementation of half-adder with all the combinations including the implementation using NAND gates only.



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

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

Fig. 4-5 Implementation of Half-Adder

- FULL ADDER : one that performs the addition of three bits(two significant bits and a previous carry) is a full adder.

Table 4-4
Full Adder

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

Simplified Expressions :

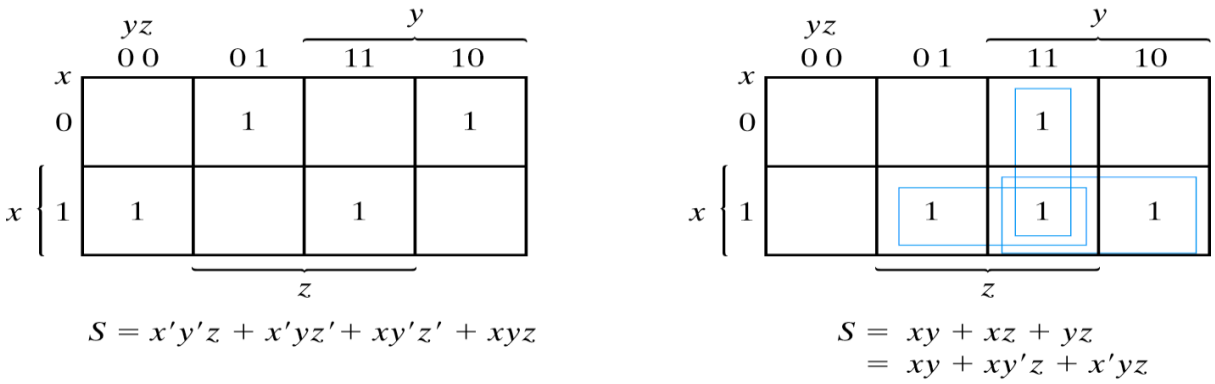


Fig. 4-6 Maps for Full Adder

S = x'y'z + x'yz' + xy'z' + xyz

C = xy + xz + yz

Full adder implemented in SOP :

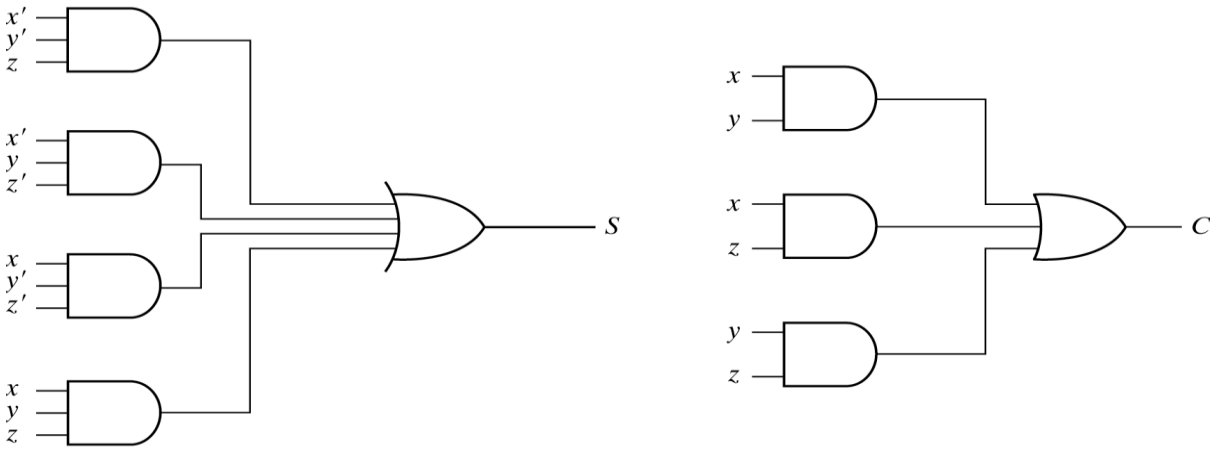


Fig. 4-7 Implementation of Full Adder in Sum of Products

Another implementation :

- Full-adder can also implemented with two half adders and one OR gate (Carry Look-Ahead adder).

S = z ⊕ (x ⊕ y)

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

= xy'z' + x'yz' + xyz + x'y'z

C = z(xy' + x'y) + xy = xy'z + x'yz + xy

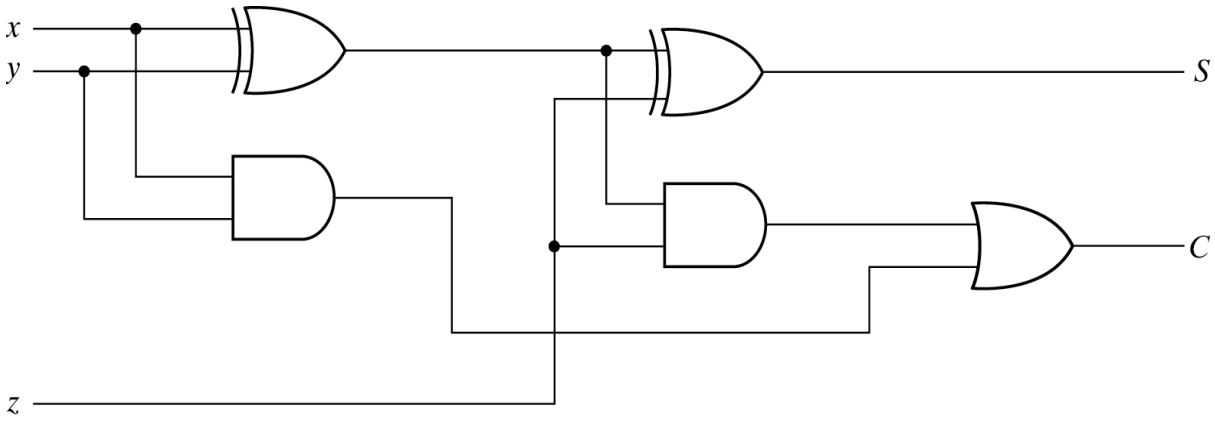


Fig. 4-8 Implementation of Full Adder with Two Half Adders and an OR Gate

Binary adder :

- This is also called Ripple Carry Adder ,because of the construction with full adders are connected in cascade.

Subscript i:	3	2	1	0	
Input carry	0	1	1	0	C_i
Augend	1	0	1	1	A_i
Addend	0	0	1	1	B_i
Sum	1	1	1	0	S_i
Output carry	0	0	1	1	C_{i+1}

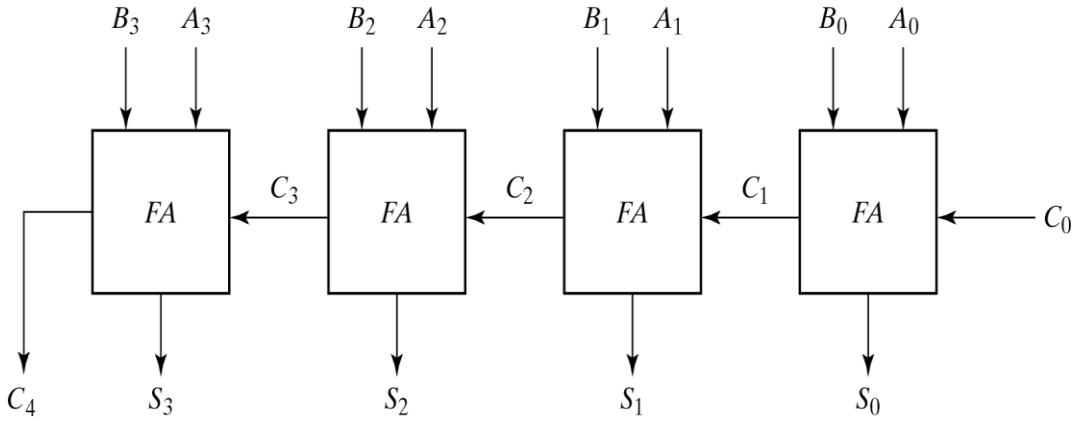


Fig. 4-9 4-Bit Adder

Carry Propagation :

- Fig.4-9 causes a unstable factor on carry bit, and produces a longest propagation delay.

- The signal from C_i to the output carry C_{i+1}, propagates through an AND and OR gates, so, for an n-bit RCA, there are 2n gate levels for the carry to propagate from input to output.
- Because the propagation delay will affect the output signals on different time, so the signals are given enough time to get the precise and stable outputs.

The most widely used technique employs the principle of carry look-ahead to improve the speed of the algorithm.

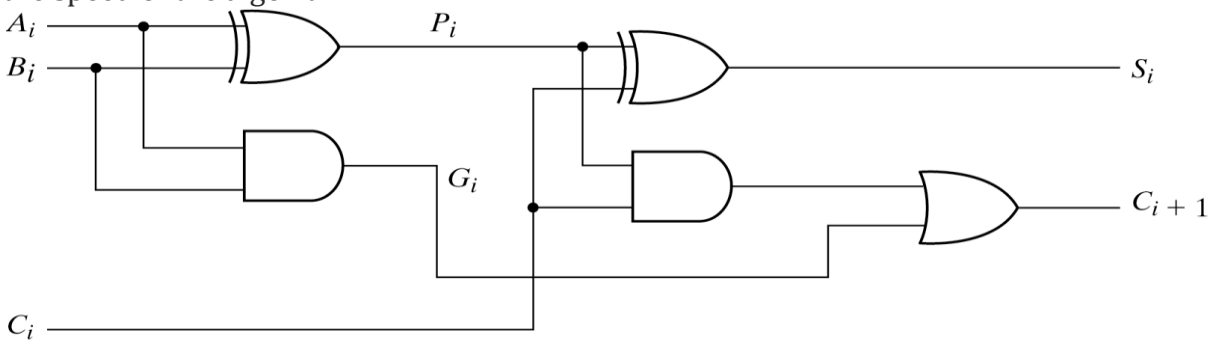


Fig. 4-10 Full Adder with P and G Shown

Boolean functions :

$P_i = A_i \oplus B_i$ steady state value

$G_i = A_iB_i$ steady state value

Output sum and carry

$S_i = P_i \oplus C_i$

$C_{i+1} = G_i + P_iC_i$

G_i : carry generate P_i : carry propagate

C_0 = input carry

$C_1 = G_0 + P_0C_0$

$C_2 = G_1 + P_1C_1 = G_1 + P_1G_0 + P_1P_0C_0$

$C_3 = G_2 + P_2C_2 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$

C_3 does not have to wait for C_2 and C_1 to propagate

Logic diagram of carry look-ahead generator :

- C_3 is propagated at the same time as C_2 and C_1.

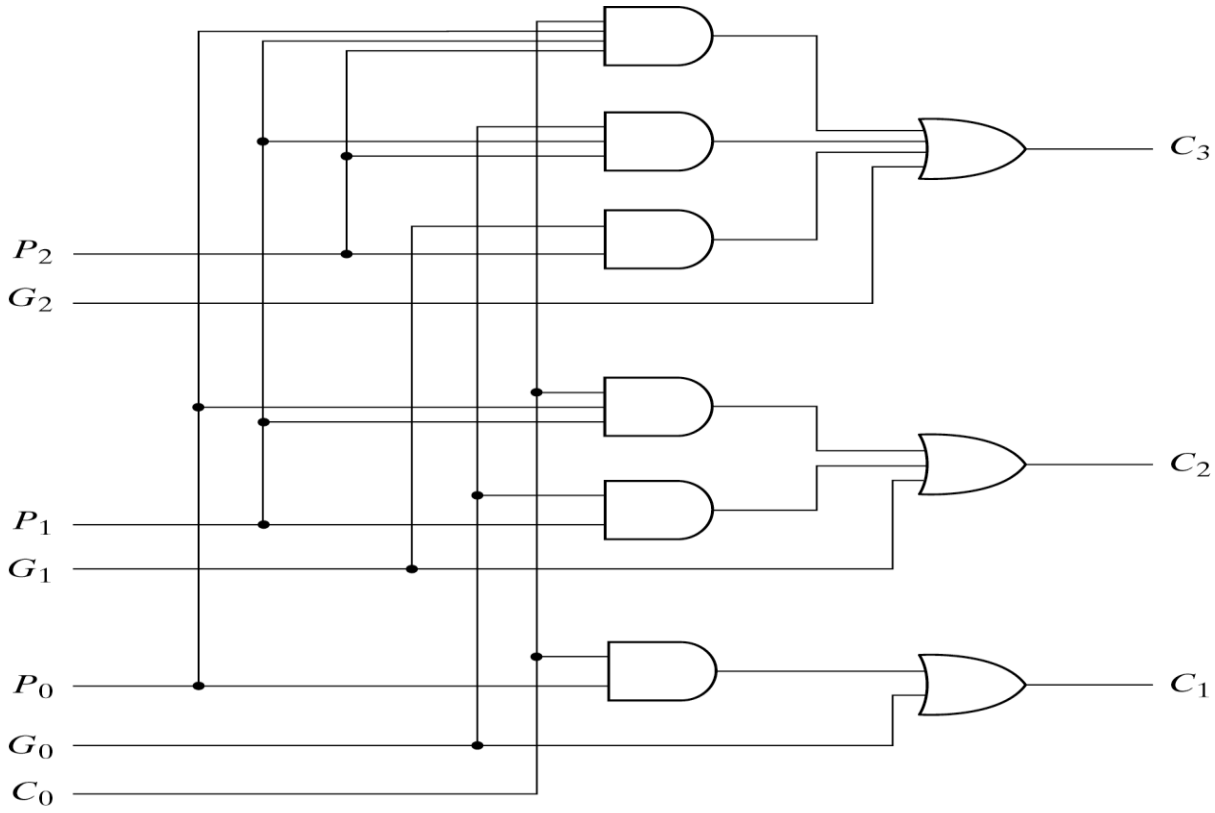


Fig. 4-11 Logic Diagram of Carry Lookahead Generator

4-bit adder with carry lookahead:

- Delay time of n-bit CLAA = XOR + (AND + OR) + XOR

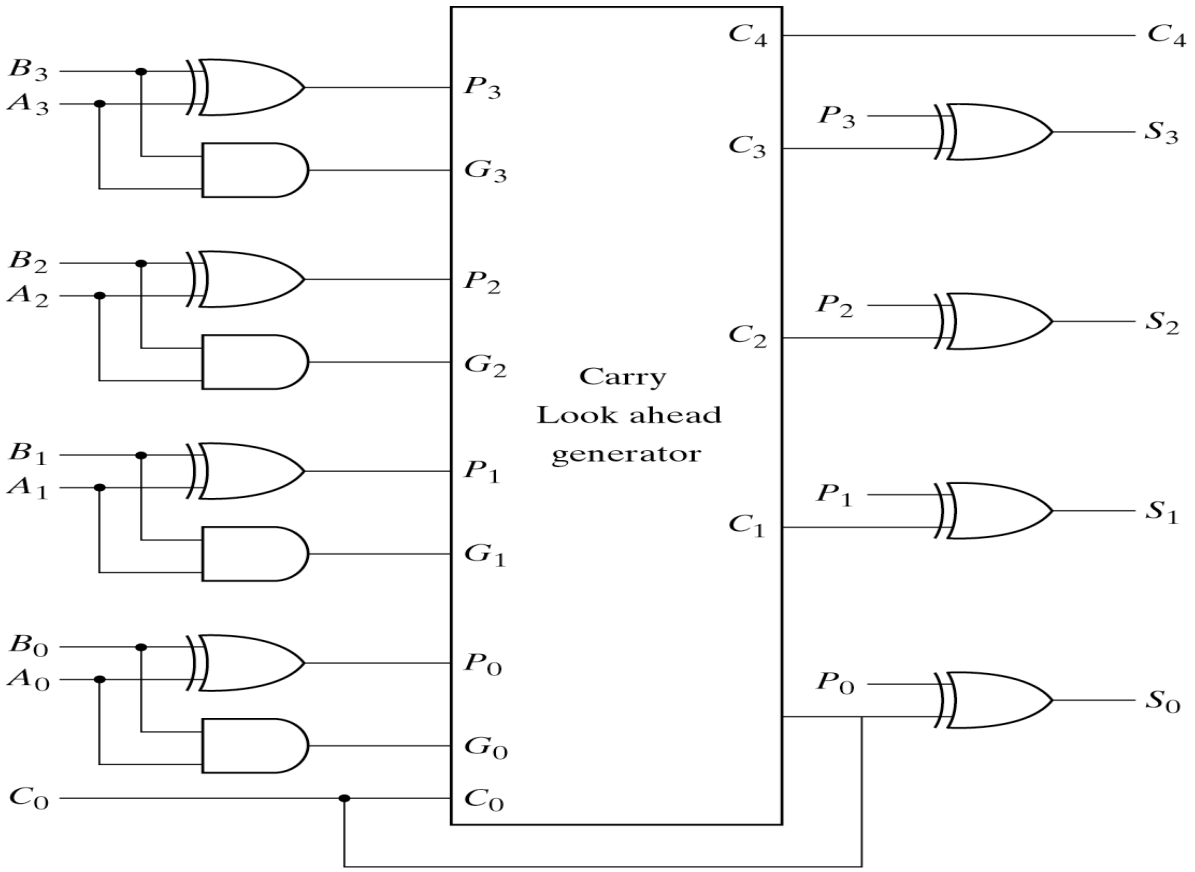


Fig. 4-12 4-Bit Adder with Carry Lookahead

Binary Adder -subtractor :

M = 1→subtractor ; M = 0→adder

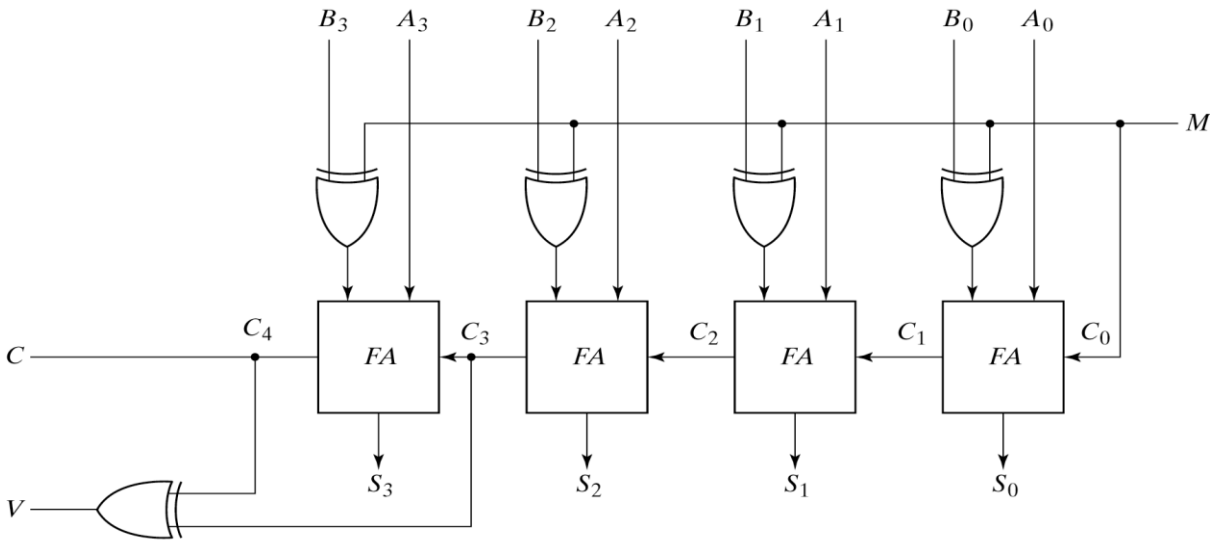


Fig. 4-13 4-Bit Adder Subtractor

- It is worth noting Fig.4-13 that binary numbers in the signed-complement system are added and subtracted by the sae basic addition and subtraction rules as unsigned numbers.

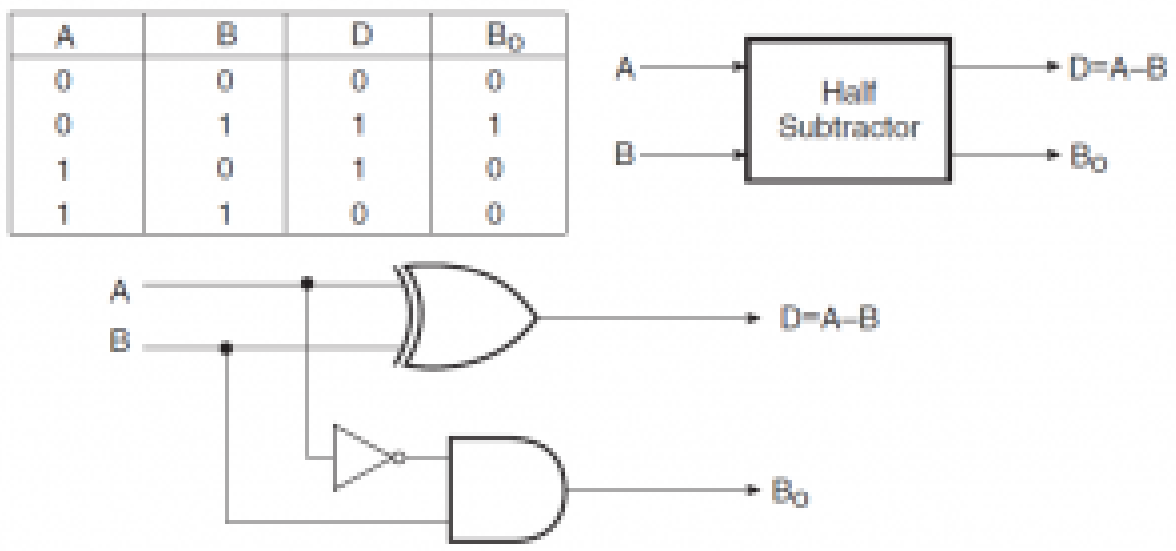
- Overflow is a problem in digital computers because the number of bits that hold the number is finite and a result that contains n+1 bits cannot be accommodated.

Boolean expression of sum can be implemented by using two-input EX-OR gate in which one of the input is Carry in and the other input is the output of another two-input EX-OR gate with A and B as its inputs. On the other hand Carry output can implemented by ORing the outputs of AND gates.

HALF-SUBTRACTOR

Half-subtractor is used to subtract one binary digit from another
to give DIFFERENCE output and a BORROW output. The truth table of a half-subtractor is shown in figure. The Boolean expressions for half-subtractor are,
D='B+AB' and Bo=A'B
Here, the DIFFERENCE i.e. the D output is an EX-OR gate and the BORROW i.e. Bo is AND gate with complemented input A. Figure shows the logic implementation of a half-subtractor. Comparing a half-subtractor with a half-adder, it can be seen that, the expressions for SUM and DIFFERENCE outputs are same. The expression for BORROW in the case of the half-subtractor is more or less same with CARRY of the half-adder. However, the case of BORROW output the the minuend is complemented and then ANDing is done.

A	B	D	B_O
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0



FULL SUBTRACTOR

Full subtractor performs subtraction of two bits, one is minuend and other is subtrahend. In full subtractor '1' is borrowed by the previous adjacent lower minuend bit. Hence there are three bits are considered at the input of a full subtractor. There are two outputs, that are DIFFERENCE output D and BORROW output Bo. The BORROW output indicates`that the minuend bit requires borrow '1' from the next minuend bit. Figure shows the truth table of a full
subtractor. The K-maps for the two outputs are shown in figure. If we compare DIFFERENCE output D and BORROW output Bo with full adder`it can be seen that the DIFFERENCE output D is the same as that for the SUM output. Further, the BORROW

output Bo is similar to CARRY-OUT. In the case of a half-subtractor, A input is complemented similar things are carried out in full subtractor.

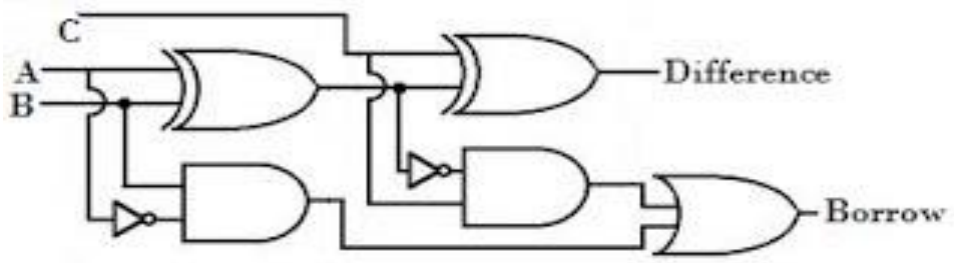
The Truth Table of Full Subtractor is Shown Below.

Full Subtractor-Truth Table				
Input			Output	
A	B	C	Difference	Borrow
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
www.flintgroups.com				

From the Truth Table The Difference and Borrow will written as

Difference=A'B'C+A'BB'+AB'C'+ABC
Reduce it like adder
Then We got
Difference=A⊕B⊕C
Borrow=A'B'C+A'BC'+A'BC+ABC
=A'B'C+A'BC'+A'BC+A'BC+A'BC+ABC ----------> A'BC=A'BC+A'BC+A'BC
=A'C(B'+B)+A'B(C'+C)+BC(A'+A)
Borrow=A'C+A'B+BC

The logic diagram of Full Subtractor is Shown below



Full Subtractor-Logic Diagram
www.flintgroups.com

N-Bit Parallel Adder

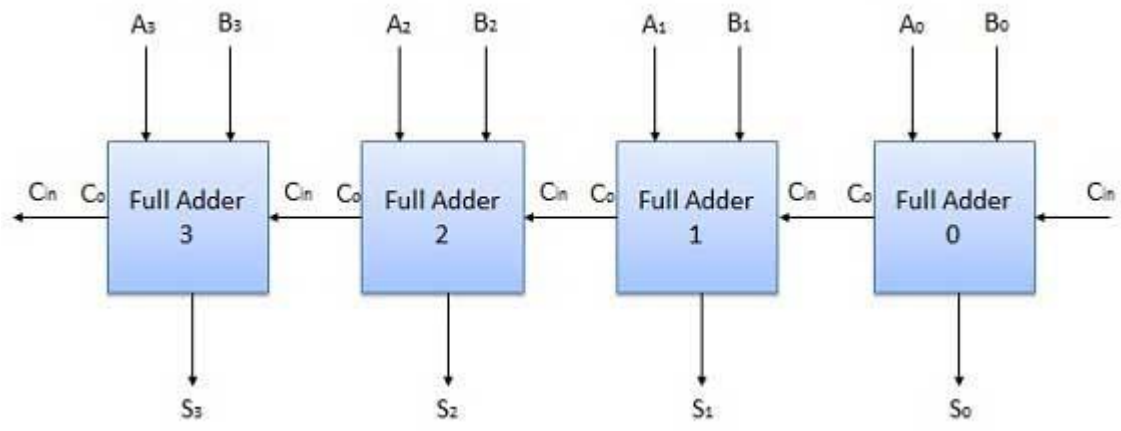
The Full Adder is capable of adding only two single digit binary number along with a carry input. But in practical we need to add binary numbers

which are much longer than just one bit. To add two n-bit binary numbers we need to use the n-bit parallel adder. It uses a number of full adders in cascade. The carry output of the previous full adder is connected to carry input of the next full adder.

4 Bit Parallel Adder

In the block diagram, A_0 and B_0 represent the LSB of the four bit words A and B. Hence Full Adder-0 is the lowest stage. Hence its C_{in} has been permanently made 0. The rest of the connections are exactly same as those of n-bit parallel adder is shown in fig. The four bit parallel adder is a very common logic circuit.

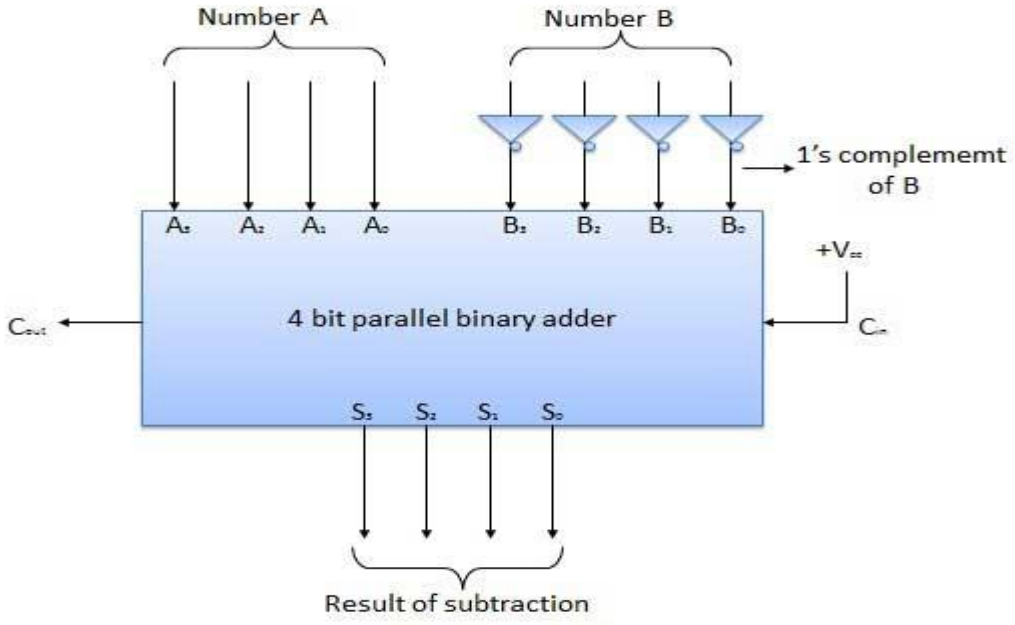
Block diagram



4 Bit Parallel Subtractor

The number to be subtracted (B) is first passed through inverters to obtain its 1's complement. The 4-bit adder then adds A and 2's complement of B to produce the subtraction. S_3 S_2 S_1 S_0 represents the result of binary subtraction (A-B) and carry output C_{out} represents the polarity of the result. If A > B then Cout = 0 and the result of binary form (A-B) then C_{out} = 1 and the result is in the 2's complement form.

Block diagram



BINARY ADDER/SUBTRACTOR

Subtraction of binary numbers can be carried out by using the addition of 2's complement of subtrahend to the minuend. In this operation If the MSB of addition is a '0', then the answer is correct and if MSB is '1', then answer is having negative sign. Hence, by using full adders subtraction can be carried out.

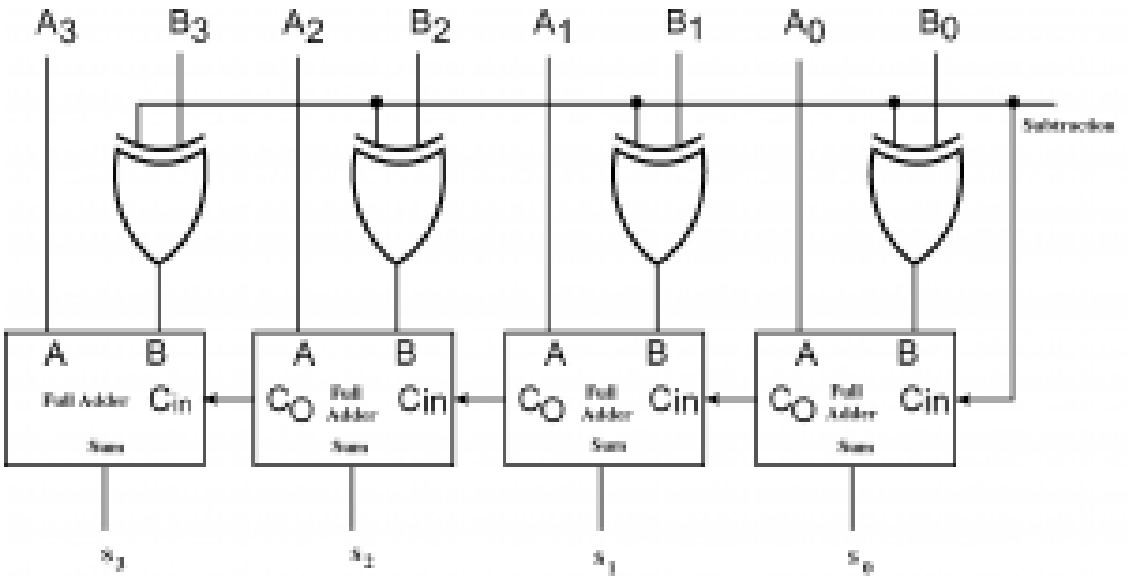


Figure above the realization of 4 bit adder-subtractor. From the figure it can be seen that, the bits of the binary numbers are given to full adder through the XOR gates. The control input is controls the addition or subtraction operation.

When the SUBTRACTION input is logic '0' , the B3 B2 B1 B0 are passed to the full adders. Hence, the output of the full adders is the addition of the two numbers.

When the SUBTRACTION input is logic '1' , the B3 B2 B1 B0 are complemented. Further, the SUBTRACTION logic '1' input is also work as Cin for the LSB full adder, due to which 2's complement addition can be carried out. Hence, the outputs of the full adders in this case is the subtraction of two numbers.

Magnitude comparator:

- The equality relation of each pair of bits can be expressed logically with an exclusive-NOR function as:

$$A = A_3A_2A_1A_0 \; ; \; B = B_3B_2B_1B_0$$

$$x_i = A_iB_i + A_i'B_i' \quad \text{for } i = 0, 1, 2, 3$$

$$(A = B) = x_3x_2x_1x_0$$

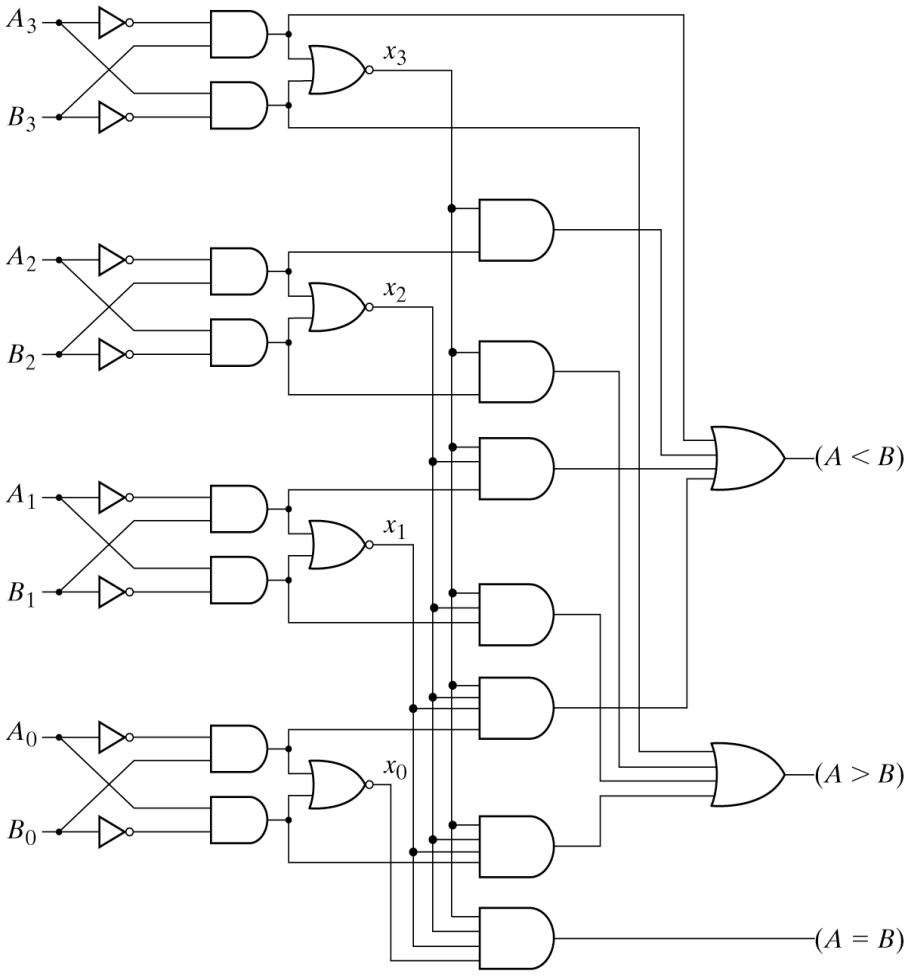


Fig. 4-17 4-Bit Magnitude Comparator

- We inspect the relative magnitudes of pairs of MSB. If equal, we compare the next lower significant pair of digits 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.

(A>B)=

$A_3B'_3+x_3A_2B'_2+x_3x_2A_1B'_1+x_3x_2x_1A_0B'_0$

(A<B)=

$A'_3B_3+x_3A'_2B_2+x_3x_2A'_1B_1+x_3x_2x_1A'_0B_0$

Decoders :

- The decoder is called n-to-m-line decoder, where $m \leq 2^n$.
- the decoder is also used in conjunction with other code converters such as a BCD-to-seven_segment decoder.
- 3-to-8 line decoder: For each possible input combination, there are seven outputs that are equal to 0 and only one that is equal to 1.

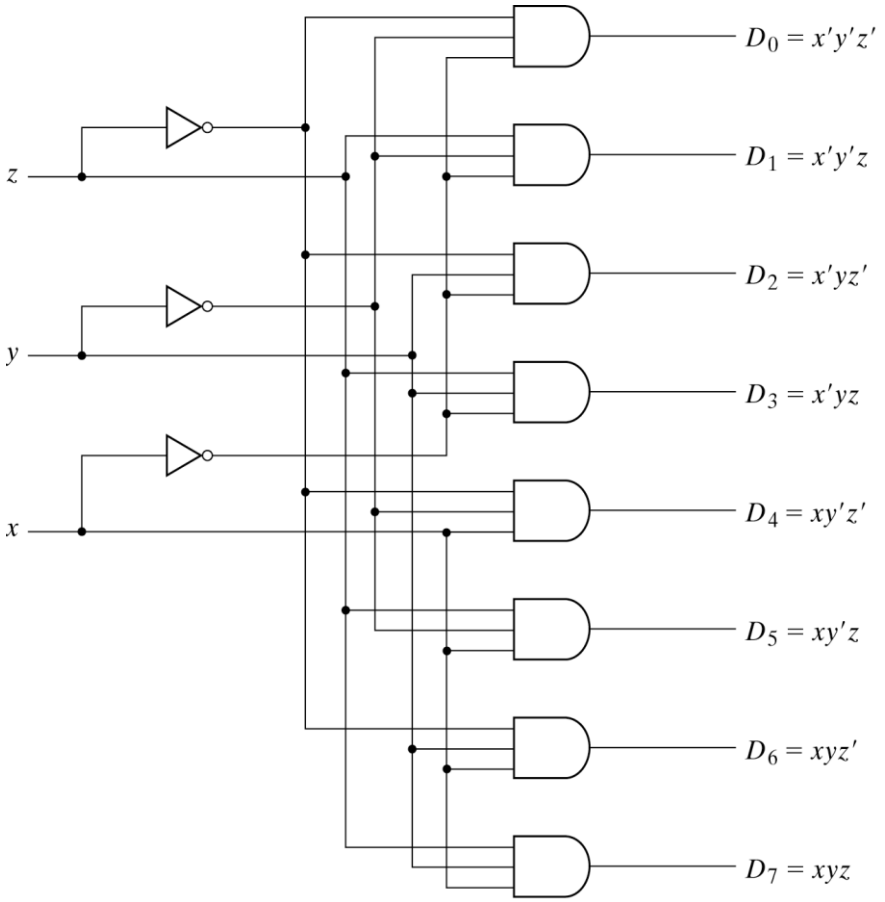


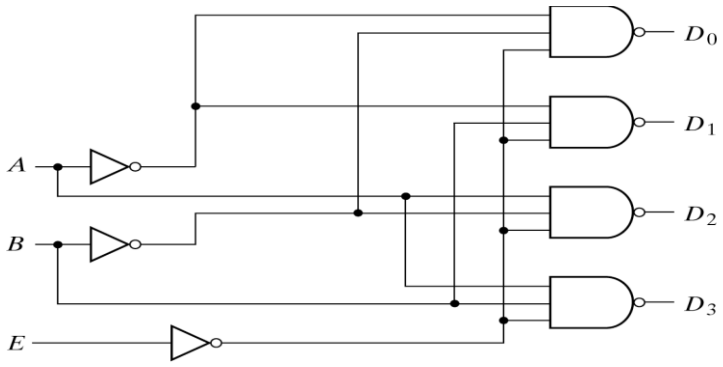
Fig. 4-18 3-to-8-Line Decoder

Table 4-6
Truth Table of a 3-to-8-Line Decoder

Inputs			Outputs							
x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Decoder with enable input :

- Some decoders are constructed with NAND gates, it becomes more economical to generate the decoder minterms in their complemented form.
- As indicated by the truth table , only one output can be equal to 0 at any given time, all other outputs are equal to 1.



(a) Logic diagram

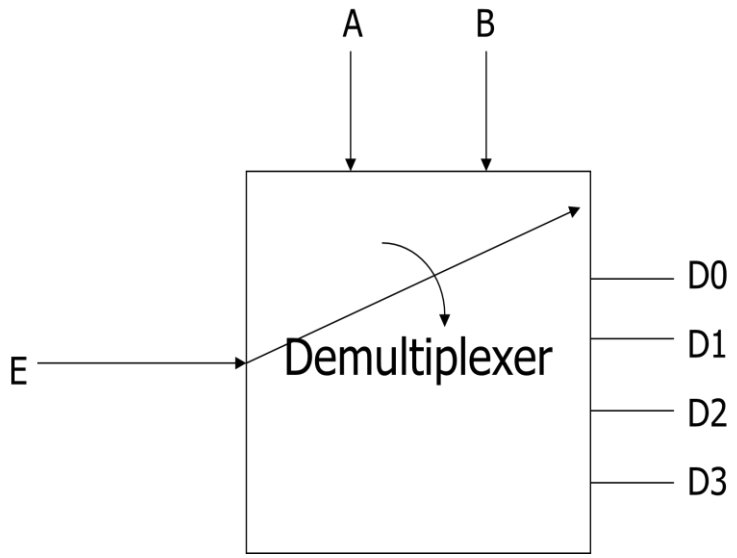
E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

(b) Truth table

Fig. 4-19 2-to-4-Line Decoder with Enable Input

Demultiplexer:

- **A decoder with an enable input is referred to as a decoder/demultiplexer.**
- **The truth table of demultiplexer is the same with decoder.**



3-to-8 decoder with enable implement the 4-to-16 decoder :

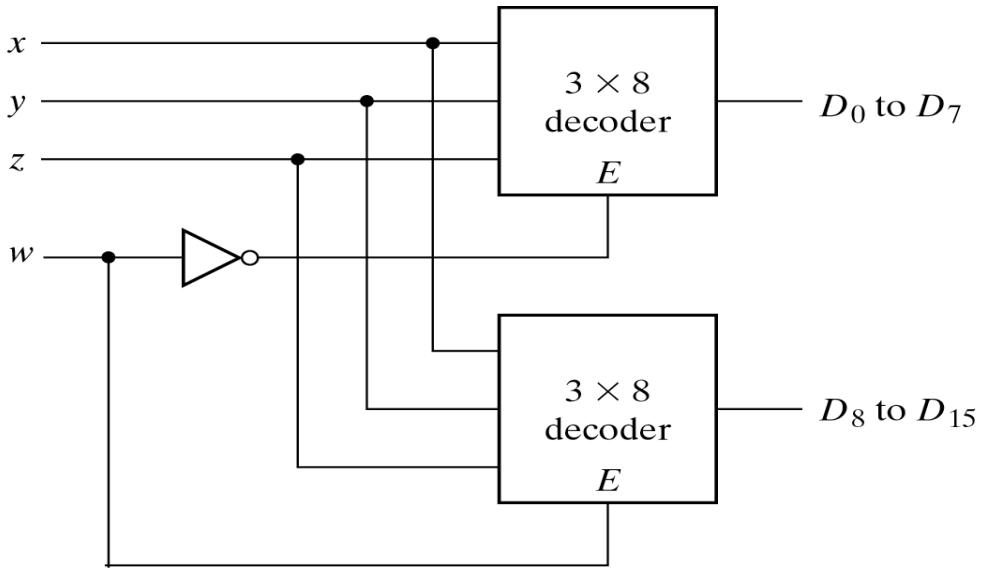


Fig. 4-20 4 × 16 Decoder Constructed with Two 3 × 8 Decoders

Implementation of a Full Adder with a Decoder:

- **From table 4-4, we obtain the functions for the combinational circuit in sum of minterms:**

$$S(x, y, z) = \sum(1, 2, 4, 7)$$

$$C(x, y, z) = \sum(3, 5, 6, 7)$$

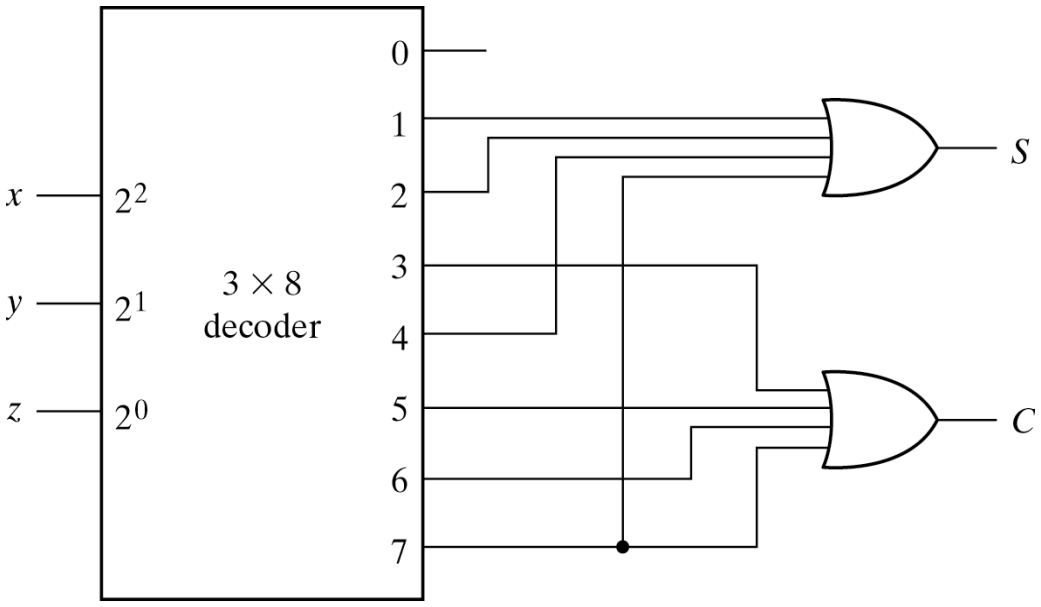


Fig. 4-21 Implementation of a Full Adder with a Decoder

Encoders:

- **An encoder is the inverse operation of a decoder.**
- **We can derive the Boolean functions by table 4-7**

$$z = D_1 + D_3 + D_5 + D_7$$

$$y = D_2 + D_3 + D_6 + D_7$$

$$x = D_4 + D_5 + D_6 + D_7$$

Table 4-7
Truth Table of Octal-to-Binary Encoder

Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

Priority encoder :

- **If two inputs are active simultaneously, the output produces an undefined combination. We can establish an input priority to ensure that only one input is encoded.**
- **Another ambiguity in the octal-to-binary encoder is that an output with all 0's is generated when all the inputs are 0; the output is the same as when D_0 is equal to 1.**
- **The discrepancy tables on Table 4-7 and Table 4-8 can resolve aforesaid condition by providing one more output to indicate that at least one input is equal to 1.**
- **V=0→no valid inputs**
- **V=1→valid inputs**
- **X's in output columns represent**
- **don't-care conditions**
- **X's in the input columns are**
- **useful for representing a truth**
- **table in condensed form.**
- **Instead of listing all 16**
- **minterms of four variables**

Table 4-8
Truth Table of a Priority Encoder

Inputs				Outputs		
D_0	D_1	D_2	D_3	x	y	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

- **4-input priority encoder :**

  Implementation of table 4-8

  x = D_2 + D_3

  y = D_3 + $D_1D'_2$

  V = D_0 + D_1 + D_2 + D_3

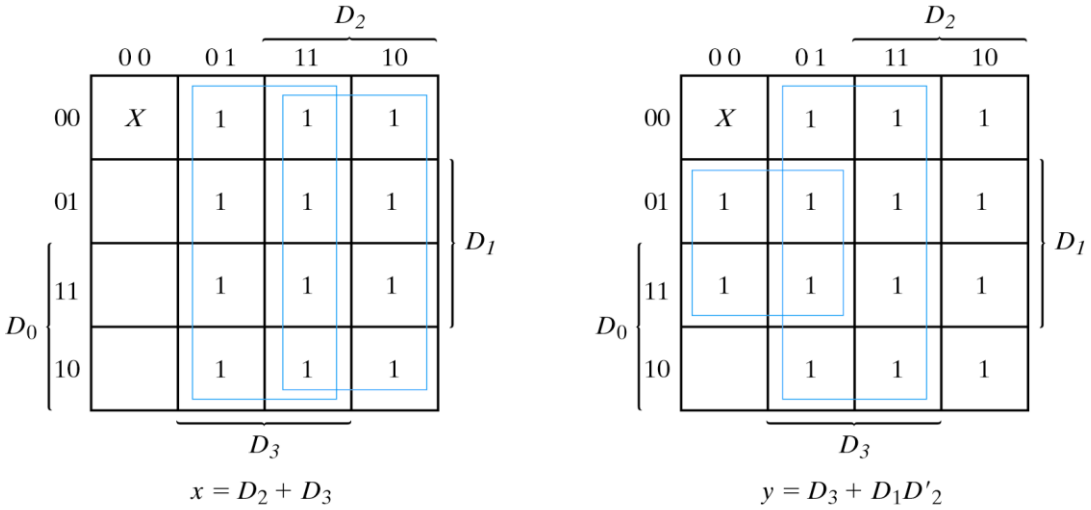


Fig. 4-22 Maps for a Priority Encoder

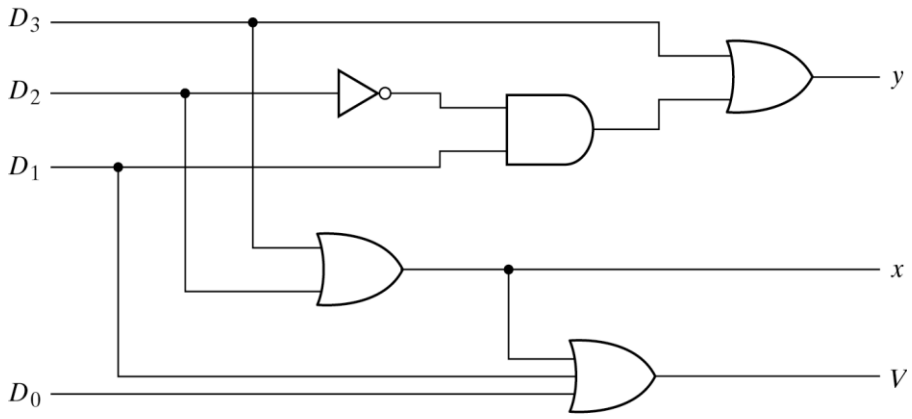
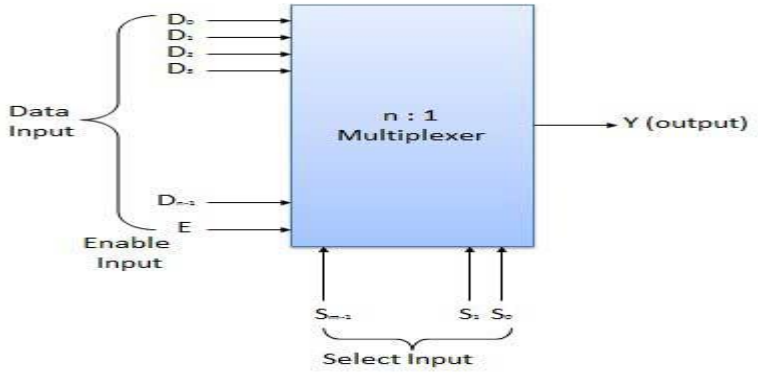


Fig. 4-23 4-Input Priority Encoder

Multiplexers

Multiplexer is a special type of combinational circuit. There are n-data inputs, one output and m select inputs with 2m = 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. 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 required operation when it is low.

Block diagram



2 to 1 Multiplexer:

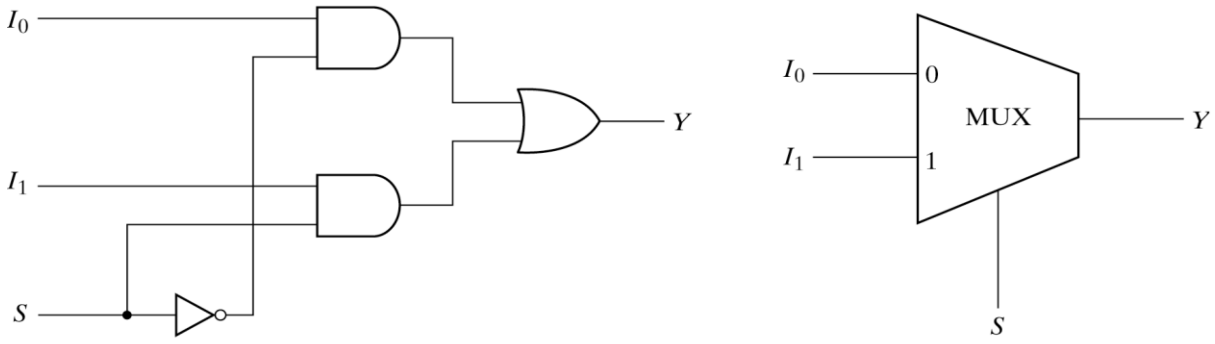
S = 0, Y = I_0 Truth Table→

S	Y
0	I_0

$Y = S'I_0 + SI_1$

S = 1, Y = I_1

1 I_1



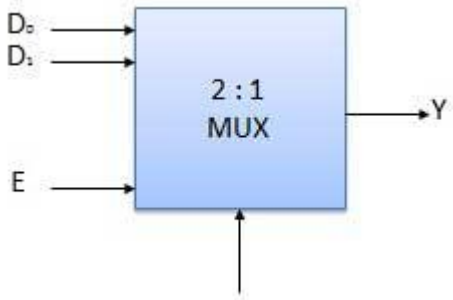
(a) Logic diagram (b) Block diagram

Fig. 4-24 2-to-1-Line Multiplexer

Multiplexers come in multiple variations

- 2 : 1 multiplexer
- 4 : 1 multiplexer
- 16 : 1 multiplexer
- 32 : 1 multiplexer

Block Diagram

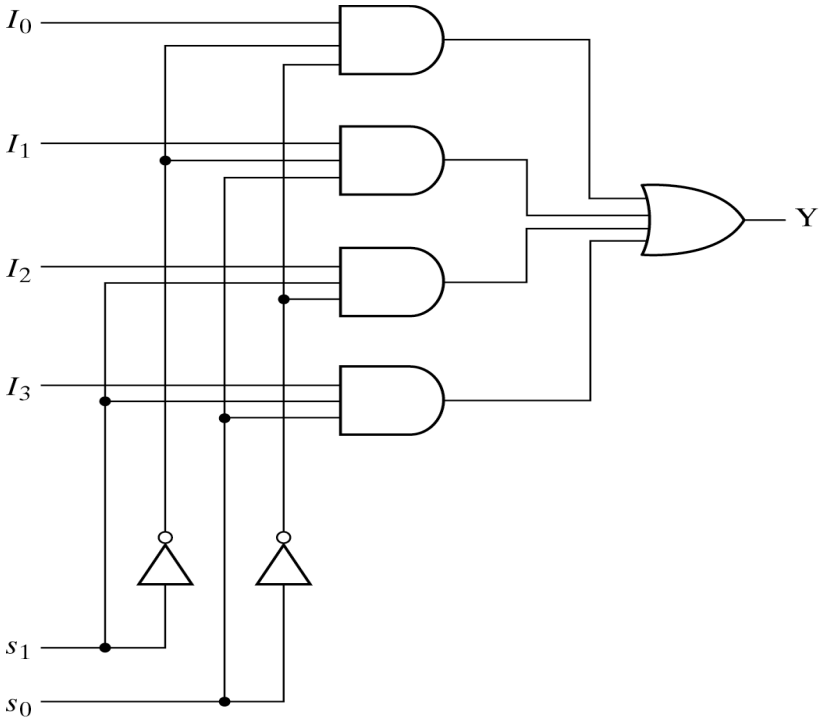


Truth Table

Enable	Select	Output
E	S	Y
0	x	0
1	0	D_0
1	1	D_1

x = Don't care

4-to-1 Line Multiplexer:



(a) Logic diagram

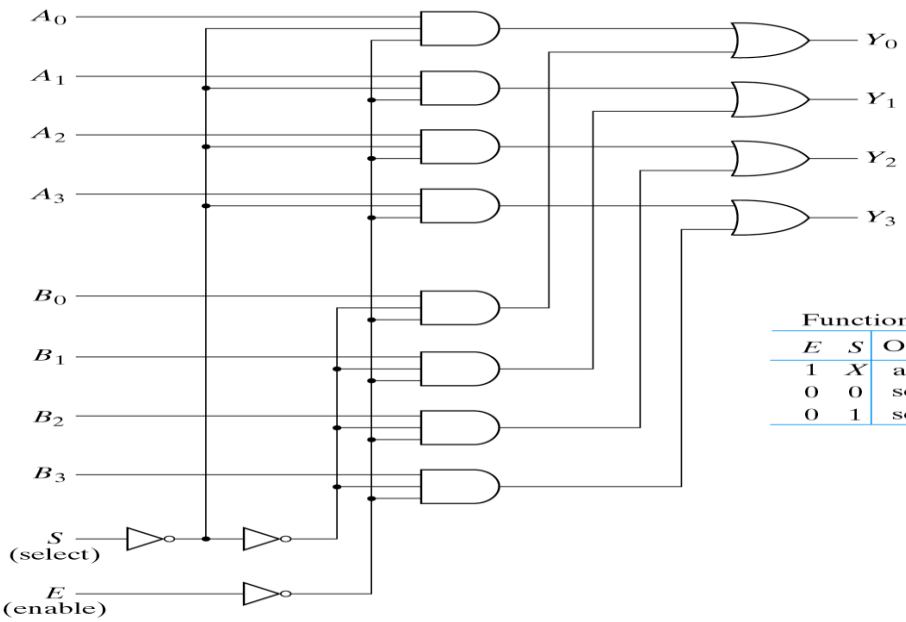
s_1	s_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

(b) Function table

Fig. 4-25 4-to-1-Line Multiplexer

Quadruple 2-to-1 Line Multiplexer:

- Multiplexer circuits can be combined with common selection inputs to provide multiple-bit selection logic. Compare with Fig4-24.



Function table

E	S	Output Y
1	X	all 0's
0	0	select A
0	1	select B

Fig. 4-26 Quadruple 2-to-1-Line Multiplexer

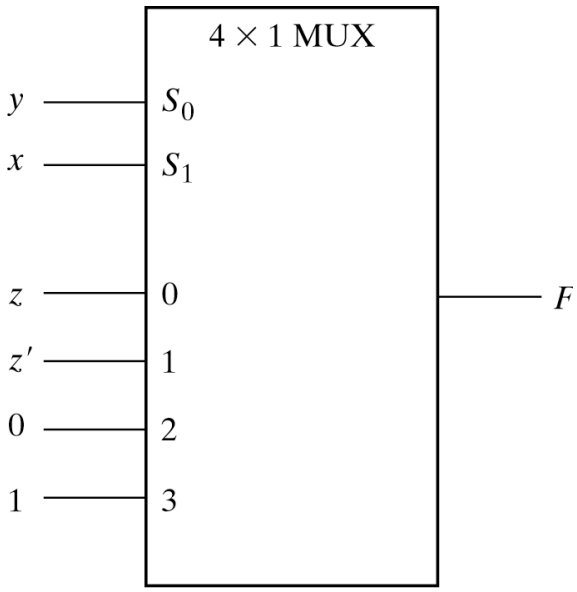
Boolean function implementation :

- A more efficient method for implementing a Boolean function of n variables with a multiplexer that has n-1 selection inputs.

F(x, y, z) = Σ(1,2,6,7)

x	y	z	F	
0	0	0	0	$F = z$
0	0	1	1	
0	1	0	1	$F = z'$
0	1	1	0	
1	0	0	0	$F = 0$
1	0	1	0	
1	1	0	1	$F = 1$
1	1	1	1	

(a) Truth table



(b) Multiplexer implementation

Fig. 4-27 Implementing a Boolean Function with a Multiplexer

4-input function with a multiplexer :

F(A, B, C, D) = Σ(1, 3, 4, 11, 12, 13, 14, 15)

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

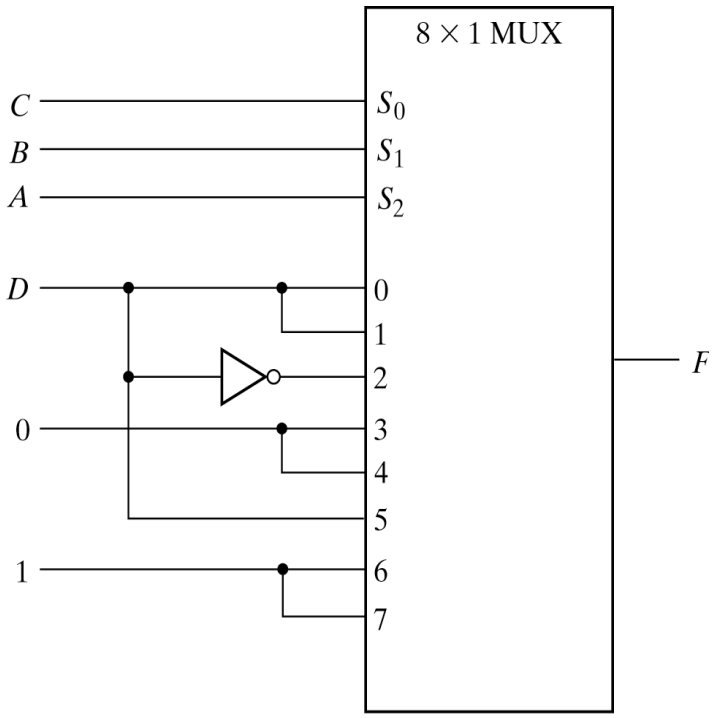


Fig. 4-28 Implementing a 4-Input Function with a Multiplexer

Three-State Gates :

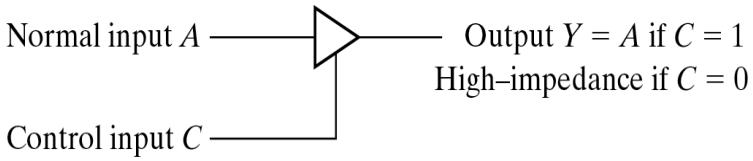
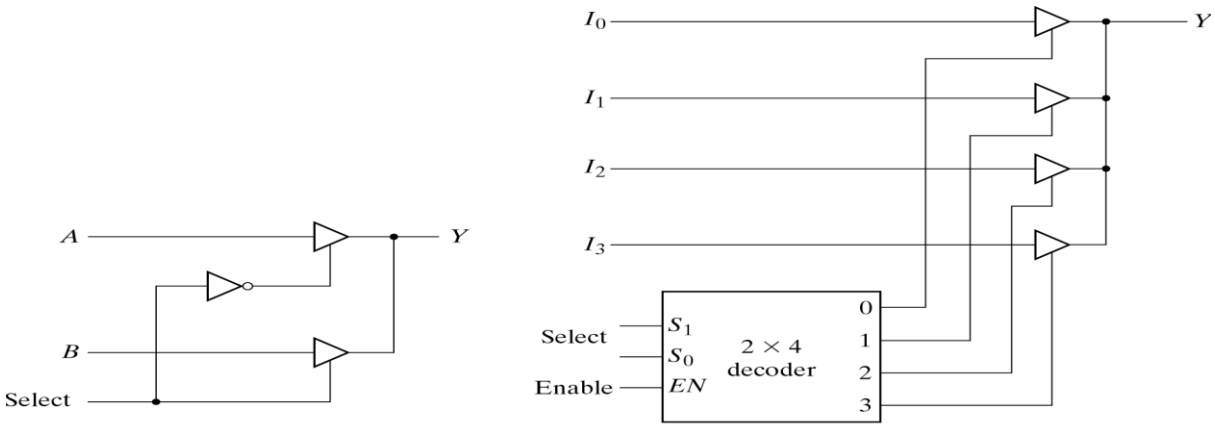


Fig. 4-29 Graphic Symbol for a Three-State Buffer



(a) 2-to-1- line mux

(b) 4 - to - 1 line mux

Fig. 4-30 Multiplexers with Three-State Gates

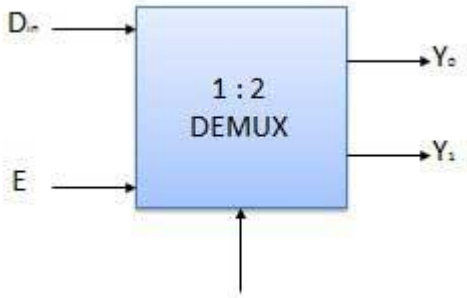
Demultiplexers

A demultiplexer performs the reverse operation of a multiplexer i.e. it receives one input and distributes it over several outputs. It has only one input, n outputs, m select input. At a time only one output line is selected by the select lines and the input is transmitted to the selected output line. A de-multiplexer is equivalent to a single pole multiple way switch as shown in fig.

Demultiplexers comes in multiple variations.

- 1 : 2 demultiplexer
- 1 : 4 demultiplexer
- 1 : 16 demultiplexer
- 1 : 32 demultiplexer

Block diagram

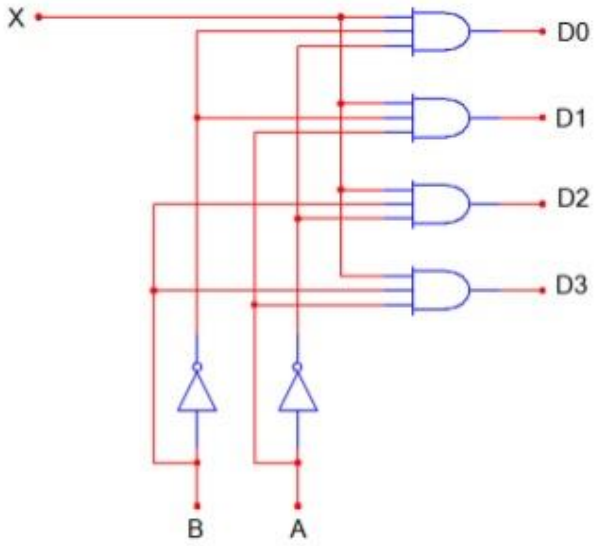


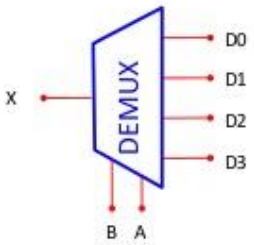
Truth Table

Enable	Select	Output
E	S	Y0 Y1
0	x	0 0
1	0	0 D_{in}
1	1	D_{in} 0

x = Don't care

1-to-4 De-Multiplexer (DEMUX)





B	A	D0	D1	D2	D3
0	0	X	0	0	0
0	1	0	X	0	0
1	0	0	0	X	0
1	1	0	0	0	X