SWITCHING THEORY & LOGIC DESIGN
BIT-11

Contact Hours Lecture : 3, Tutorial : 1 , Practical: 2 (online 4 lectures/week)
Number of Credits :5

B.Tech (IT) **Semester: III**

By
JAY PRAKASH
(Asst. Prof)

Department of information technology & Computer Application
Madan Mohan Malaviya University of Technology
Gorakhpur, India

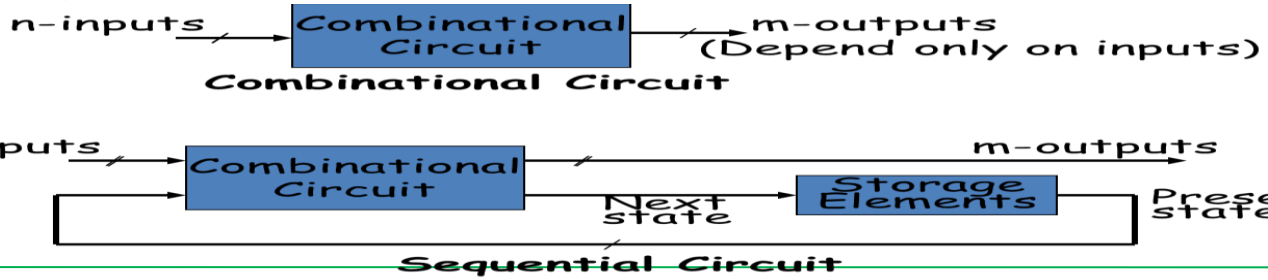
Unit-II

Digital Circuits

- *Digital circuits are two types*

1. *Combinational circuit* consists of logic gates whose outputs at any time are determined directly from the present combination of inputs without regard to previous inputs.

2. *Sequential Circuit* employ memory elements in addition to logic gates. Their outputs are a function of the inputs and the state of the memory elements.



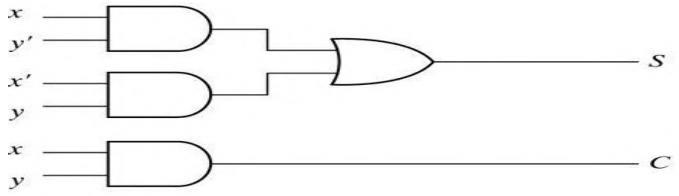
- Derive the truth table
- Simplify the Boolean expression for each output
- Produce the required circuit

Half Adder : A combinational circuit that performs the addition of two bits is called a half adder, We need two input and outputs

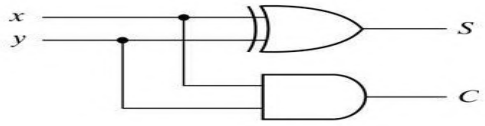
■The truth table for the half adder is listed below:

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

S(x,y) = x'y + xy' **C(x,y) = xy**



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

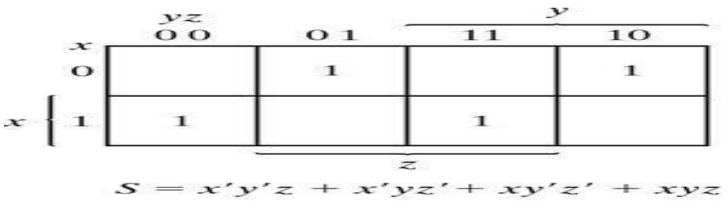


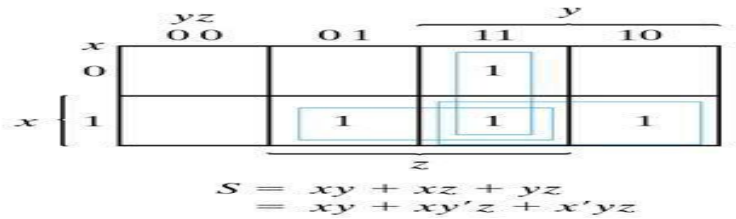
(b) $S = x \oplus y$
$C = xy$

Full Adder :

One that performs the addition of three bits (two significant bits and a previous carry) is a 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





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

- Alternative formulae using algebraic manipulation:

C = X.Y + X.Z + Y.Z

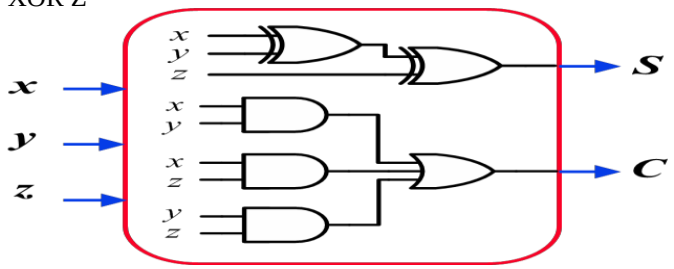
= X.Y + (X + Y).Z

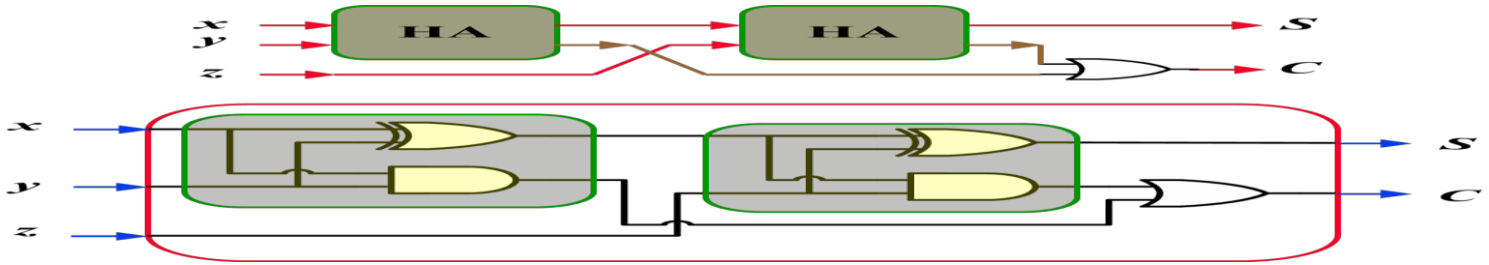
= X.Y + ((X XOR Y) + X.Y).Z

= X.Y + (X XOR Y).Z + X.Y.Z

= X.Y + (X XOR Y).Z

S = X'.Y'.Z + X'.Y.Z' + X.Y'.Z' + X.Y.Z
= X‘. (Y'.Z + Y.Z') + X. (Y'.Z' + Y.Z)
= X'. (Y XOR Z) + X. (Y XOR Z)'
= X XOR (Y XOR Z) or (X XOR Y) XOR Z





Full adder using two half adder

Parallel Binary Adders

•Parallel Adder is a digital circuit that produces the arithmetic sum of 2 binary numbers.
•Constructed with full adders connected in cascade, with output carry from each full adder connected to the input carry of next full adder in the chain.
•The carries are connected in a chain through the full adders.
•Two methods to handle carries in parallel adder

1. Ripple carry: carry out of each FA is connected to the carry input of next FA
2. Carry Look ahead: it anticipates the output carry of each stage, and based on the input bits of each stage, produces the output carry by either carry generation or carry propagation.

Carry generation: output carry is produced internally by the FA.carry is generated only when both input bits are 1s.the generated carry C is expressed as the AND function of two input bits A and B so C=AB.

Carry propagation: occurs when the i/p carry is rippled to become the o/p carry.an i/p carrymay be propagated by the full adder when either or both of the i/p bits are 1s.the propagated carry Cp is expressed as the OR function of the i/p bits ie Cp=A+B

hree sum bits are □ , □

2- Bit Parallel Adder

- LSB of two binary numbers are represented by A_1 and B_1.The next higher bit are A_2 and B_2. The resulting $_{1\ 2}$ and C_O, in which the C_O becomes MSB.
- The carry output C_O of each adder is connected as the carry input of the next higher order.

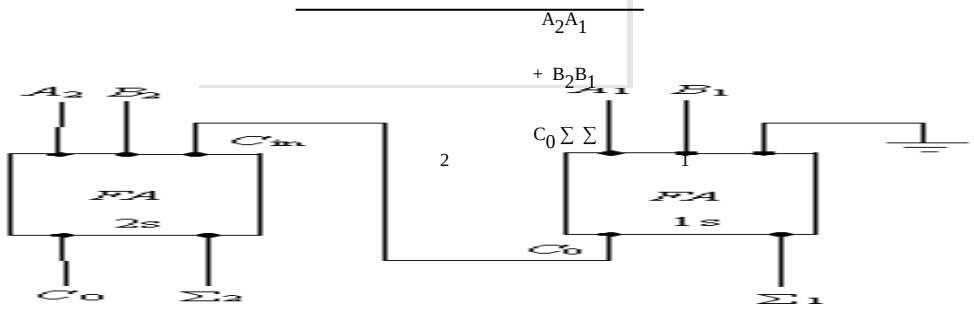


Fig : bit adder using two full adder

Four Bit Parallel Adders

- An n-bit adder requires n full adders with each output connected to the input carry of the next higher-order full adder.
- The carry output of each adder is connected to the carry input of next adder called as internal carries.

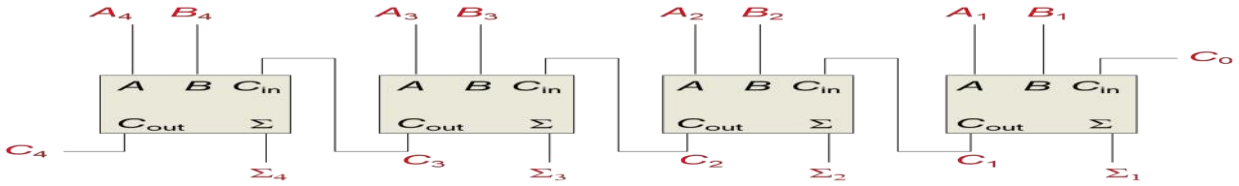


Fig : bit adder

The logic symbol for a 4-bit parallel adder is shown. This 4-bit adder includes a Carry In (labeled C_0) and a Carry Out (labeled C_4).

Input bit for number A	Input bit for number B	Carry bit input C_{IN}	Sum bit output S	Carry bit output 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

Fig : Truth table for 4 bit parallel adder

- 74283 Four-bit binary adder
- 7483 is an older chip that is functionally identical to the 74283, but the pins are laid out differently

Cascading Parallel Adders

When we connect the outputs from one circuit to the inputs of another identical circuit to expand the number of bits being operated on, we say that the circuits are *cascaded* together.

For example, you can cascade two 4-bit parallel adders to add two 8-bit numbers. To do this, connect the lower-order adder's Carry Out to the higher-order adder's Carry In.

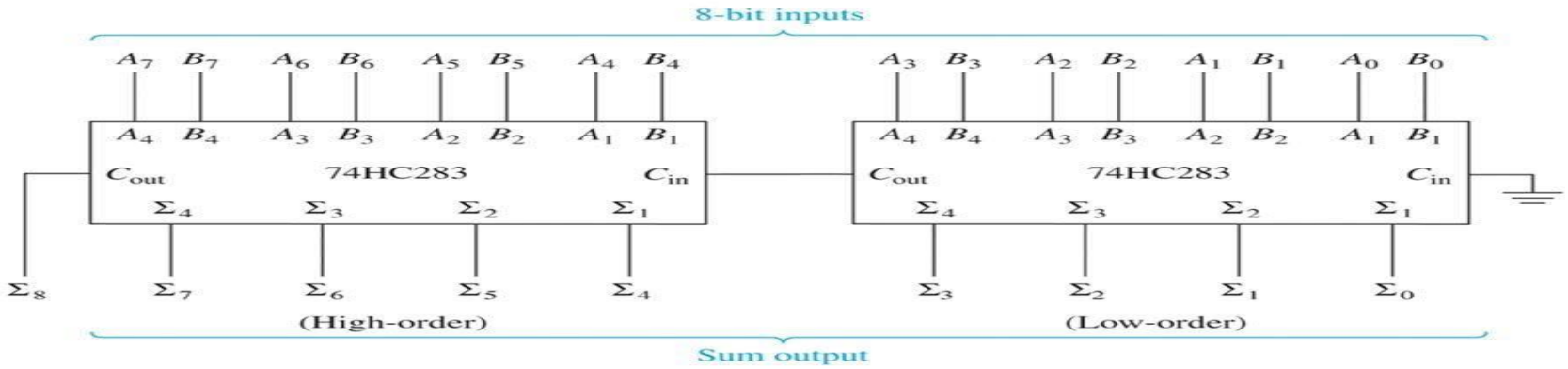


Fig : cascading of parallel adders

Binary Subtraction

- The subtraction *A*-*B* can be performed by taking the 2's complement of *B* and adding to *A*.
  - The 2's complement of *B* can be obtained by complementing B and adding one to the result.

$$A-B = A + 2C(B) \quad = \quad A + 1C(B) + 1 \quad = A + B' + 1$$

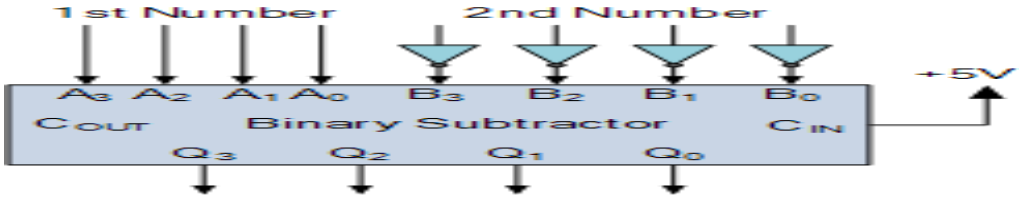


Fig : binary substractor circuit

Adder/Substractor

- **Design requires:**

(i) XOR gates:

(ii)S connected to carry-in.

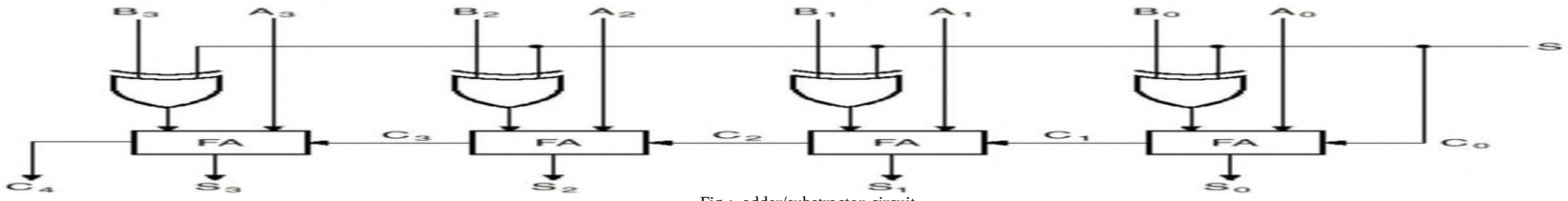


Fig : adder/substractor circuit

- When *S*=0, the circuit performs *A* + *B*. The carry in is 0, and the XOR gates simply pass *B* untouched.
- When *S*=1, the carry into the least significant bit (LSB) is 1, and *B* is complemented (1's complement) prior to the addition; hence, the circuit adds to A the 1's complement of *B* plus 1 (from the carry into the LSB).

Magnitude Comparator :

A magnitude comparator is a combinational circuit that compares two numbers, A and B, and then determines their relative magnitudes.

A > B A = B A < B

Problem: Design a magnitude comparator that compares 2 4-bit numbers A and B and determines whether:

A > B, or A = B, or A < B

Inputs

First n-bit number A and Second n-bit number B

Outputs: 3 output signals (GT, EQ, LT), where: GT = 1 IFF A > B EQ = 1 IFF A = B LT = 1 IFF A < B

Exactly One of these 3 outputs equals 1, while the other 2 outputs are 0`s.

Solution:

Inputs: 8-bits (A ⇒ 4-bits, B ⇒ 4-bits).A and B are two 4-bit numbers. Let A = $A_3A_2A_1A_0$, and Let B = $B_3B_2B_1B_0$.

Design of the EQ

•Define $X_i = A_i$ xnor $B_i = A_i B_i + A_i' B_i'$

x_i = 1 IFF $A_i = B_i$ ∀ i =0, 1, 2 and 3

x_i = 0 IFF $A_i \neq B_i$

•Therefore the condition for A = B or EQ=1 IFF

$A_3 = B_3 \rightarrow (X_3 = 1)$, and $A_2 = B_2 \rightarrow (X_2 = 1)$, and $A_1 = B_1 \rightarrow (X_1 = 1)$, and $A_0 = B_0 \rightarrow (X_0 = 1)$.

•Thus, EQ=1 IFF $X_3 X_2 X_1 X_0 = 1$. In other words, EQ = $X_3 X_2 X_1 X_0$

Designing GT and LT:

•GT = 1 if A > B:

✓ If $A_3 > B_3$ $_3 = 1$ and $B_3 = 0$ If $A_3 = B_3$ and $A_2 > B_2$

✓

If $A_3 = B_3$ and $A_2 = B_2$ and $A_1 >$ A_1

✓

If $A_3 = B_3$ and $A_2 = B_2$ and $A_1 = B_1$ and $A_0 > B_0$

✓

•Therefore,

GT = A_3B_3'+ $X_3 A_2 B_2$'+ $X_3 X_2 A_1 B_1$'+ $X_3 X_2$ X_1A_0 B_0'

Similarly, LT = $A_3'B_3 + X_3 A_2'B_2 + X_3 X_2 A_1'B_1 + X_3 X_2 X_1A_0'$ B_0

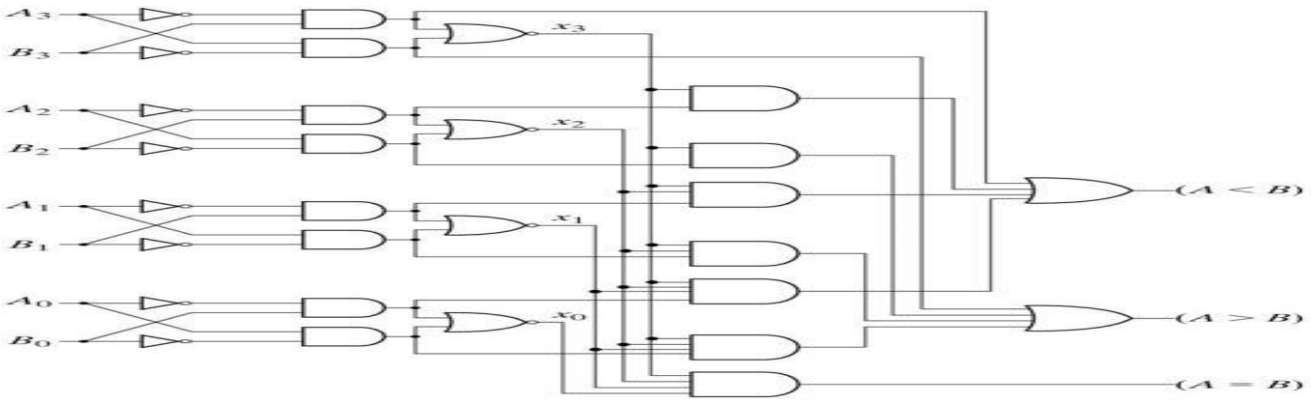


Fig :4 bit magnitude comparator

Decoder : A decoder is a logic circuit that accepts a set of inputs that represents a binary number and activates only the output that corresponds to the input number.

- A decoder has ------ N inputs and 2^N outputs
  - Exactly one output will be active for each combination of the inputs.

- Each of these input combinations only one of the *M* outputs will be active *HIGH* (1), all the other outputs are *LOW* (0).
- An AND gate can be used as the basic decoding element because it produces a HIGH output only when all inputs are HIGH.

- If an active-LOW output (74138, one of the output will low and the rest will be high) is required for each decoded number, the entire decoder can be implemented with NAND gates Inverters
- If an active-HIGH output (74139, one of the output will high and the rest will be low) is required for each decoded number, the entire decoder can be implemented with AND gates Inverters

Decoder example : **2-to-4-Line Decoder**

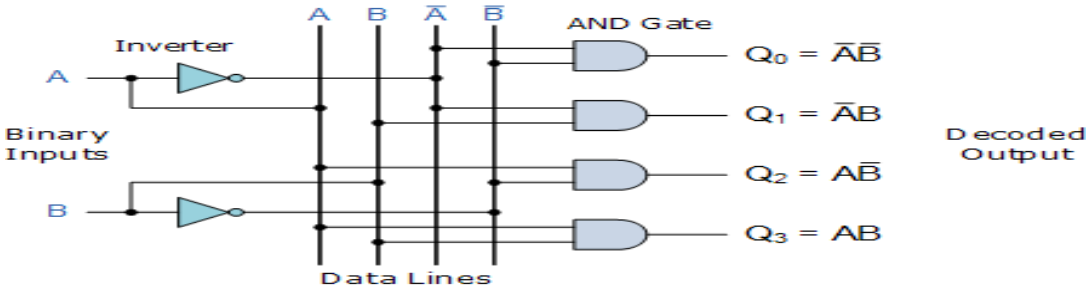
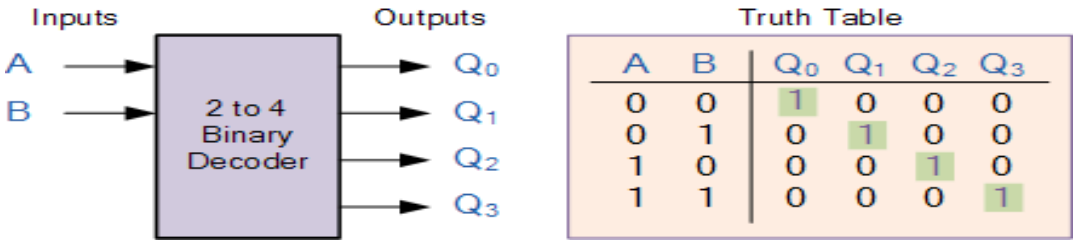


fig : 2-4 decoder with active high output

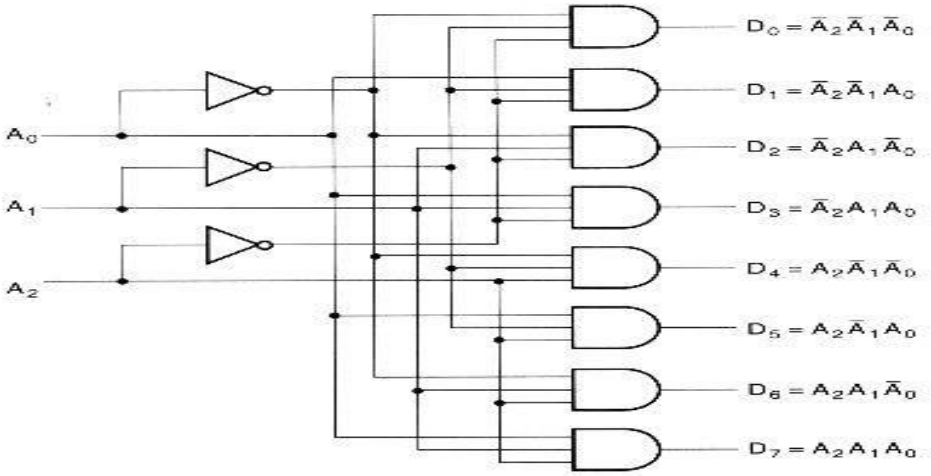


A	B	Q_0	Q_1	Q_2	Q_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Fig : logical symbol and truth table of 2-4 decoder

3 line to 8 line decoder (active-HIGH)

- It can be called a 3-line-to- 8-line decoder, because it has three input lines and eight output lines.
- It could also be called a binary-octal decoder or converters because it takes a three bit binary input code and activates the one of the eight outputs corresponding to that code. It is also referred to as a 1-of-8 decoder, because only 1 of the 8 outputs is activated at one time.



A2	A1	A0	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

Application example

- **A simplified computer I/O port system with a port address decoder with only four address lines shown.**

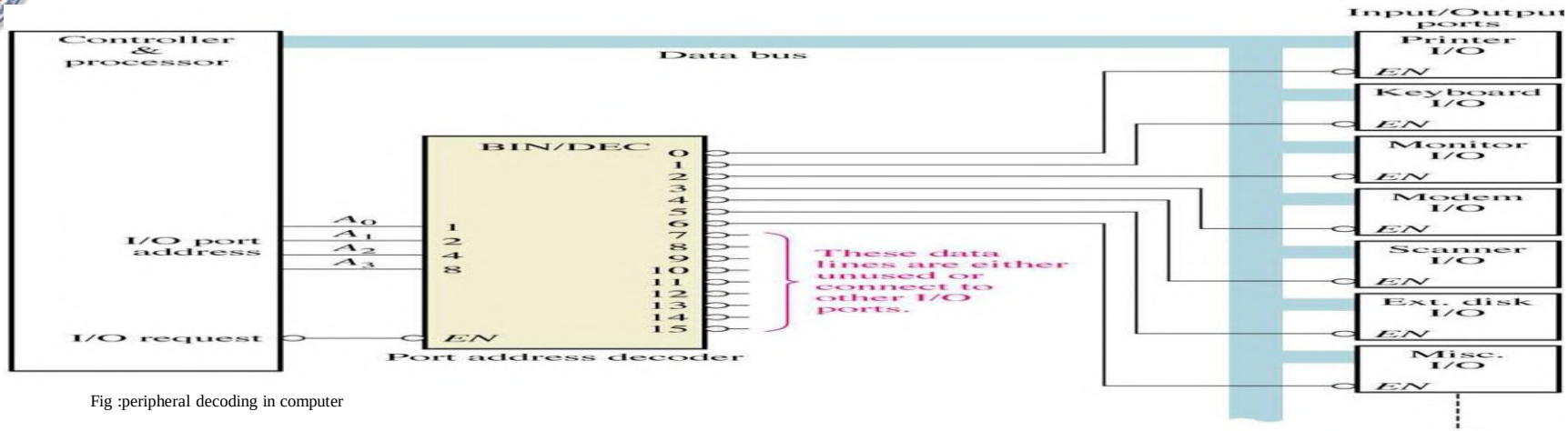


Fig :peripheral decoding in computer

•Computer must communicate with a variety of external devices called peripherals by sending and/or receiving data through what is known as input/output (I/O) ports

•Each I/O port has a number, called an address, which uniquely identifies it. When the computer wants to communicate with a particular device, it issues the appropriate address code for the I/O port to which that particular device is connected. The binary port address is decoded and appropriate decoder output is activated to enable the I/O port

•Binary data are transferred within the computer on a data bus, which is a set of parallel lines

BCD -to- Decimal decoders

- The BCD- to-decimal decoder converts each BCD code into one of Ten Positional decimal digit indications. It is frequently referred as a 4-line -to- 10 line decoder
- The method of implementation is that only ten decoding gates are required because the BCD code represents only the ten decimal digits 0 through 9.
- Each of these decoding functions is implemented with NAND gates to provide active -LOW outputs. If an active HIGH output is required, AND gates are used for decoding

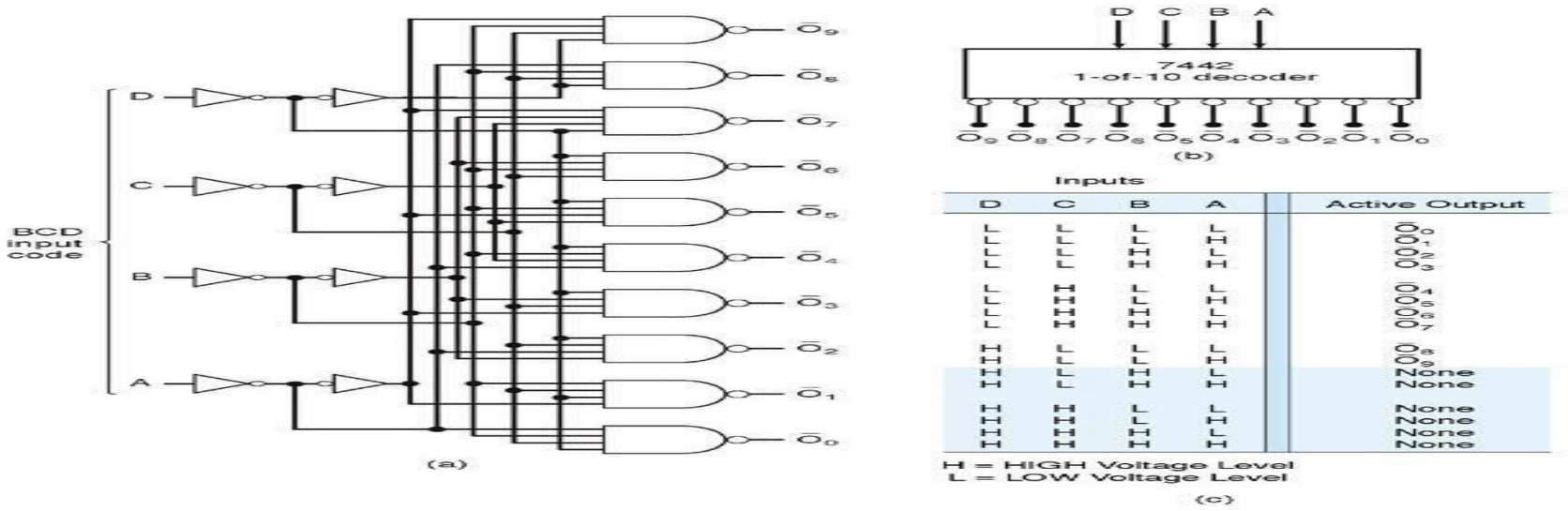


Fig : a. logic circuit for BCD to Decimal decoder b. logical symbol of BCD to Decimal decoder IC c. truth table

BCD to 7-Segment Display Decoder

1. Used to convert a BCD or a binary code into a 7 segment code used to operate a 7 segment LED display.
2. It generally has 4 input lines and 7 output lines. Here we design a simple display decoder circuit using logic gates.

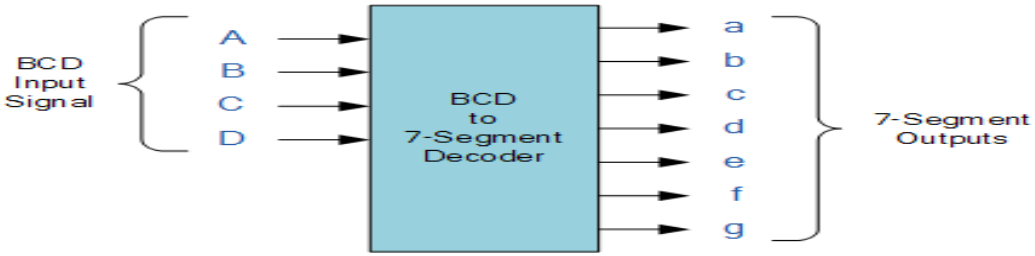


Fig : logical symbol of BCD-7 segment decoder

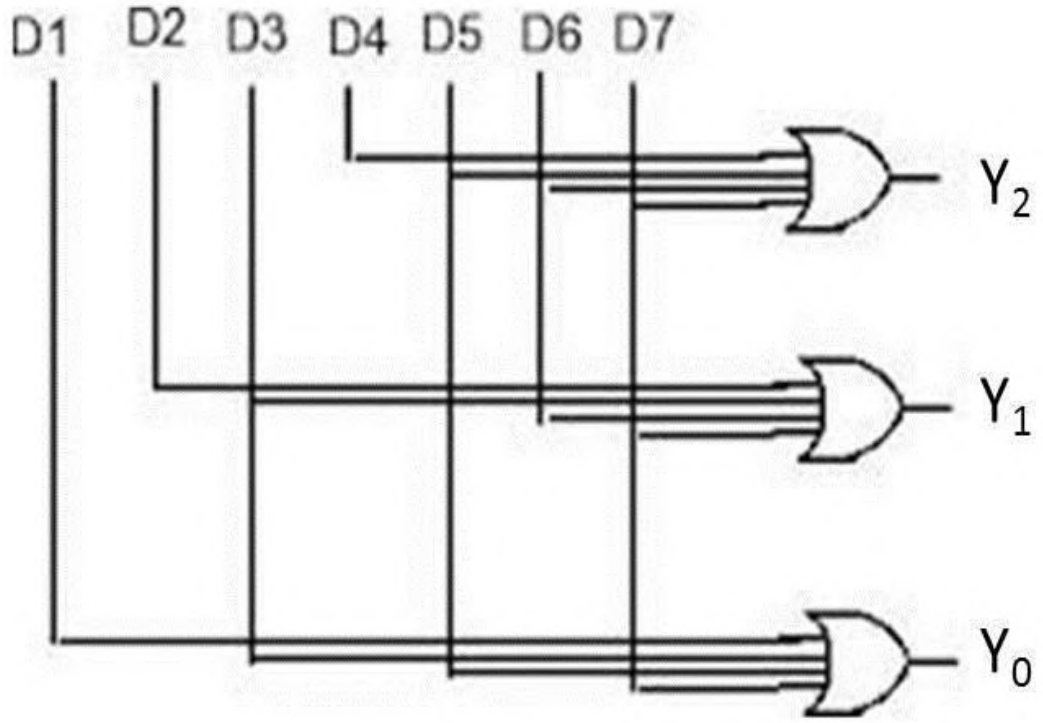
Encoders

- An encoder is a digital circuit that performs the inverse operation of a decoder. An encoder has 2^n input lines and n output lines.
- The output lines generate the binary equivalent to the input line whose value is 1.
- Example: 8-to-3 binary encoder (octal-to-binary)

Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

$$A_0 = Y_0 = D_1 + D_3 + D_5 + D_7$$
$$A_1 = Y_1 = D_2 + D_3 + D_6 + D_7$$
$$A_2 = Y_2 = D_4 + D_5 + D_6 + D_7$$

$$A_0 = Y_0 = D_1 + D_3 + D_5 + D_7$$
$$A_1 = Y_1 = D_2 + D_3 + D_6 + D_7$$
$$A_2 = Y_2 = D_4 + D_5 + D_6 + D_7$$



Decimal – BCD encoder

•Encoder will produce a BCD output corresponding to the highest-order decimal digit input that is active and will ignore any other lower order active inputs.

DECIMAL DIGIT	BCD CODE A_3	A_2	A_1	A_0
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

Fig :truth table for 10-4 encoder

-- From the truth table, the outputs can be expressed by following Boolean Function.

Note: Below Boolean functions are formed by OR ing all the input lines for which output is 1. For instance A_0 is 1 for 1,3,5,7 or 9 input lines.

$A_0 = 1+3+5+7+9$

A_1 = 2+3+6+7
A_2 = 4+5+6+7
A_3 = 8+9

The decimal to bcd encoder can therefore be implemented with OR gates whose inputs are determined directly from truth table as shown in the image below.

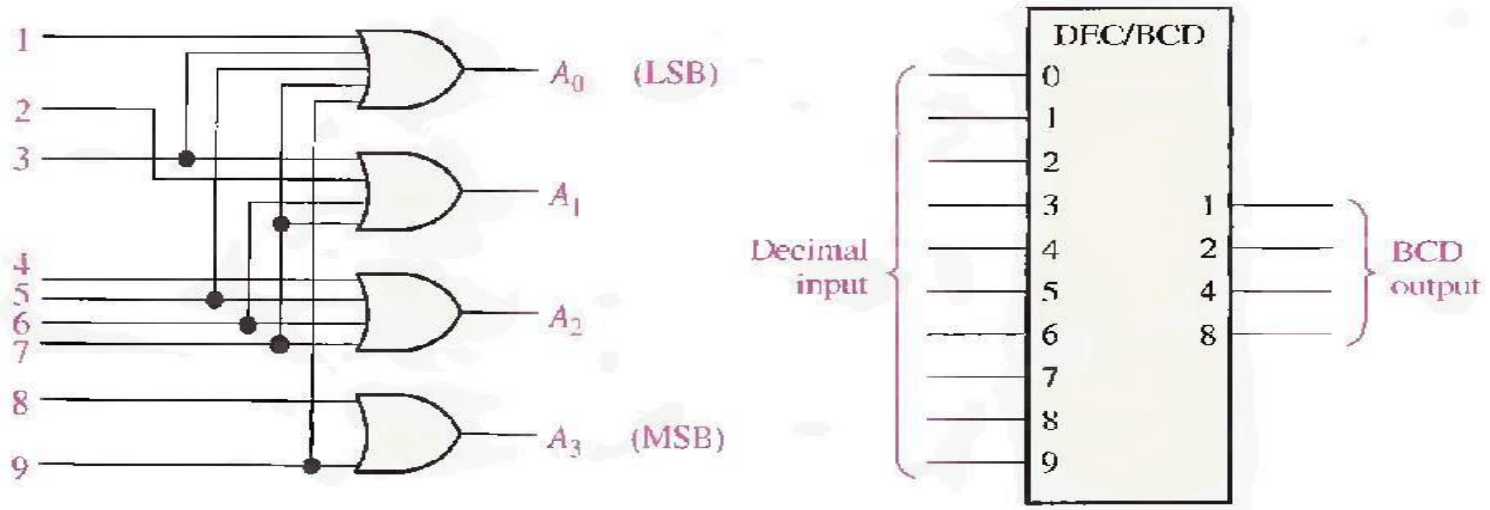


Fig : Logic circuit and symbol for 10-4 encoder

Encoder Design Issues

- There are two ambiguities associated with the design of a simple encoder:
  1. Only one input can be active at any given time. If two inputs are active simultaneously, the output produces an undefined combination (for example, if D_3 and D_6 are 1 simultaneously, the output of the encoder will be 111.
  2. An output with all 0's can be generated when all the inputs are 0's,or when D_0 is equal to 1.

Priority Encoders

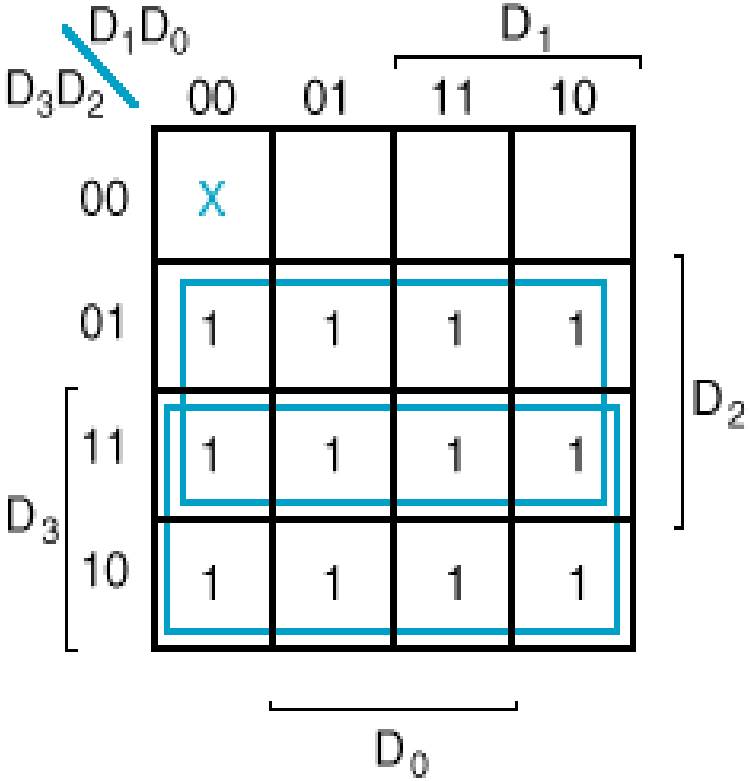
- Solves the ambiguities mentioned above.
- Multiple asserted inputs are allowed; one has priority over all others.
- Separate indication of no asserted inputs.

Example: 4-to-2 Priority Encoder Truth Table

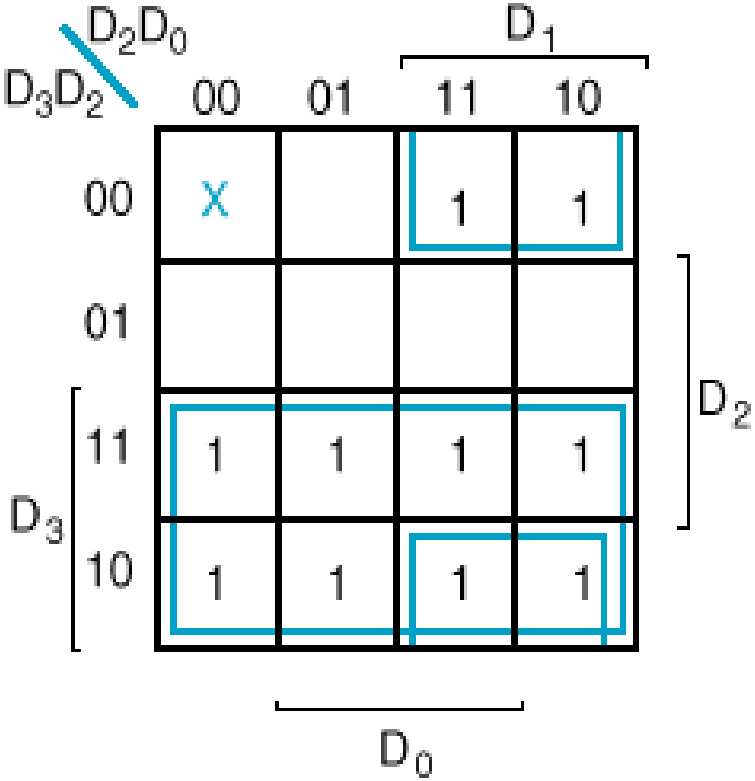
Inputs				Outputs		
D_3	D_2	D_1	D_0	A_1	A_0	V
0	0	0	0	X	X	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1

- The operation of the priority encoder is such that:
- If two or more inputs are equal to 1 at the same time, the input in the highest-numbered position will take precedence.
- A ***valid output indicator***, designated by V, is set to 1 only when one or more inputs are equal to 1. $V = D_3 + D_2 + D_1 + D_0$ by inspection.

Example: 4-to-2 Priority Encoder
K-Maps

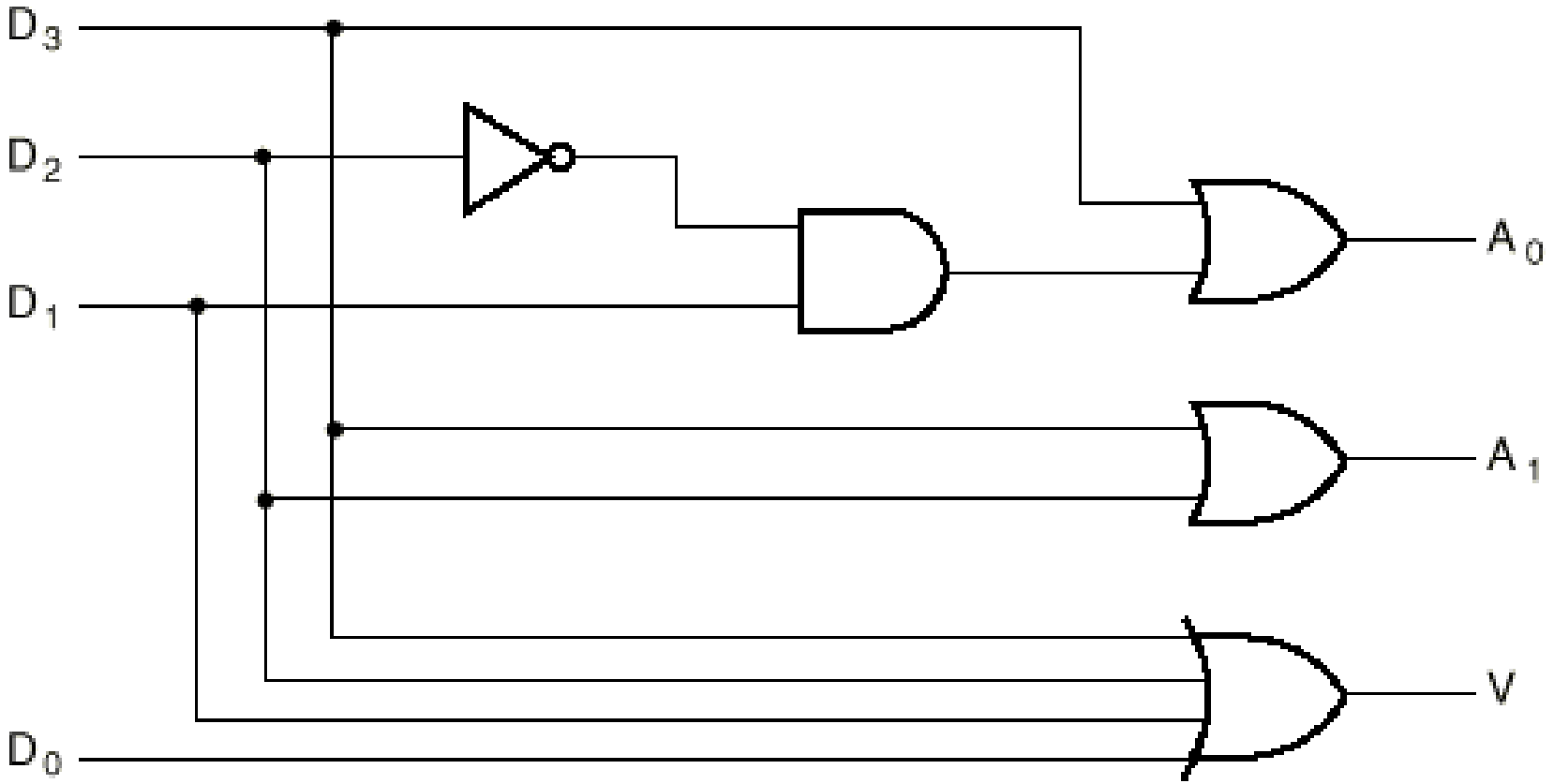


$A_1 = D_2 + D_3$



$A_0 = D_3 + D_1\overline{D}_2$

Example: 4-to-2 Priority Encoder
Logic Diagram



8-to-3 Priority Encoder

Table 4.6 A priority encoder.

A_0	A_1	A_2	A_3	A_4	A_5	A_6	A_7	Z_0	Z_1	Z_2	NR
0	0	0	0	0	0	0	0	X	X	X	1
X	X	X	X	X	X	X	1	1	1	1	0
X	X	X	X	X	X	1	0	1	1	0	0
X	X	X	X	X	1	0	0	1	0	1	0
X	X	X	X	1	0	0	0	1	0	0	0
X	X	X	1	0	0	0	0	0	1	1	0
X	X	1	0	0	0	0	0	0	1	0	0
X	1	0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	0	0

Uses of priority encoders (cont.)

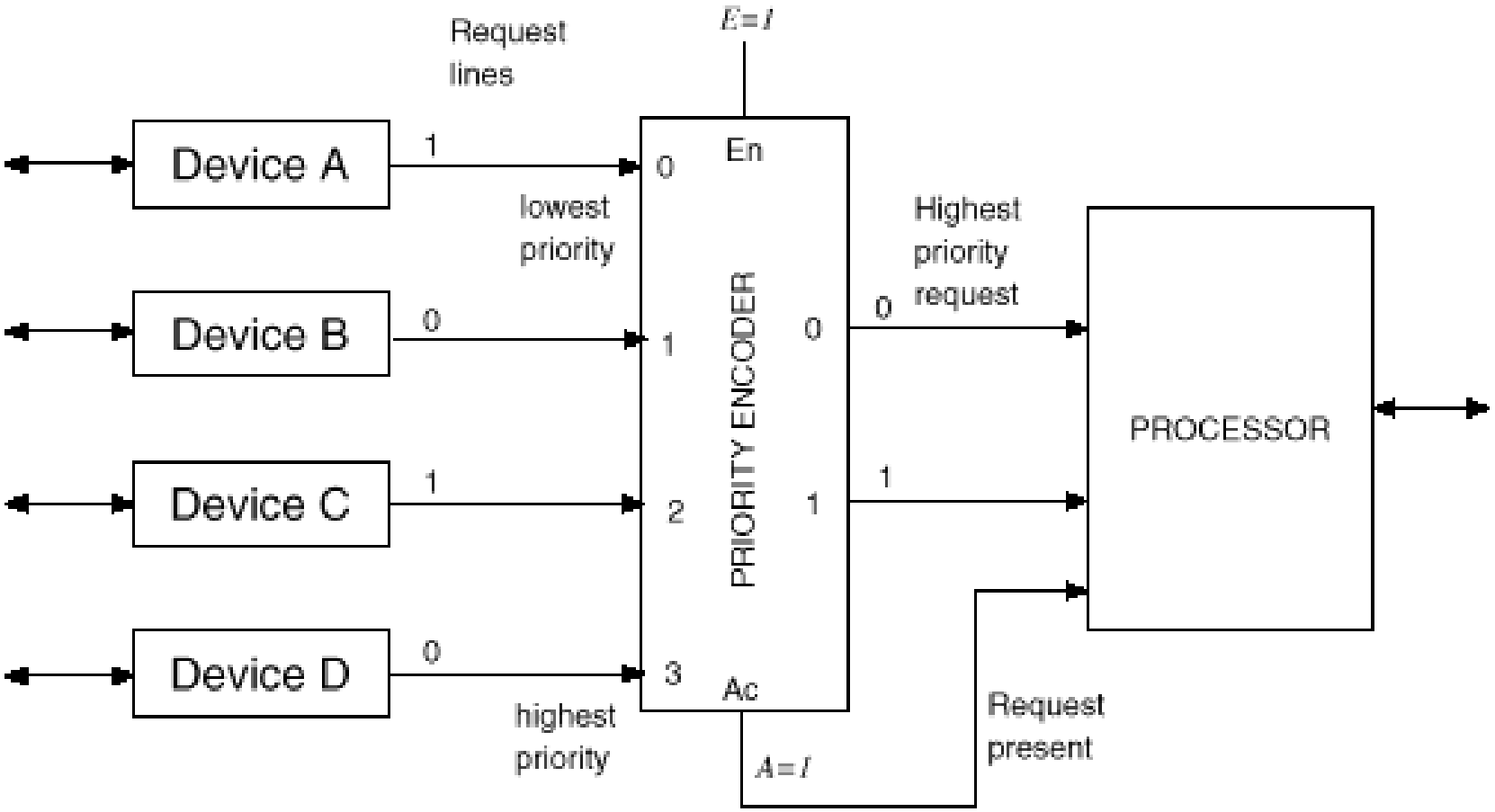


Figure 9.17: Resolving interrupt requests using a priority encoder.

Multiplexer

- A **multiplexer** or **mux** is a device that selects one of several analog or digital input signals and forwards the selected input into a single line. A multiplexer of 2^n inputs has n select lines, which are used to select which input line to send to the output."
- The select lines determine which input is connected to the output.
- MUX Types

  2-to-1 (1 select line)

  4-to-1 (2 select lines)

  8-to-1 (3 select lines)

  16-to-1 (4 select lines)

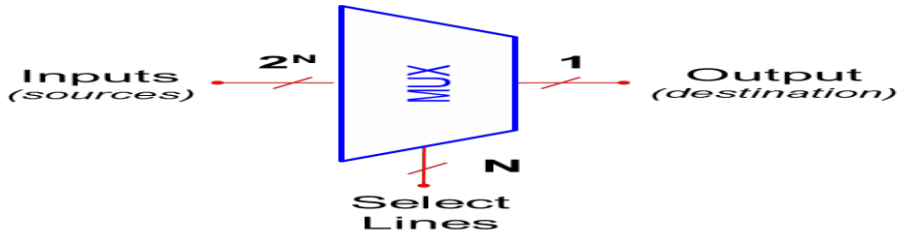
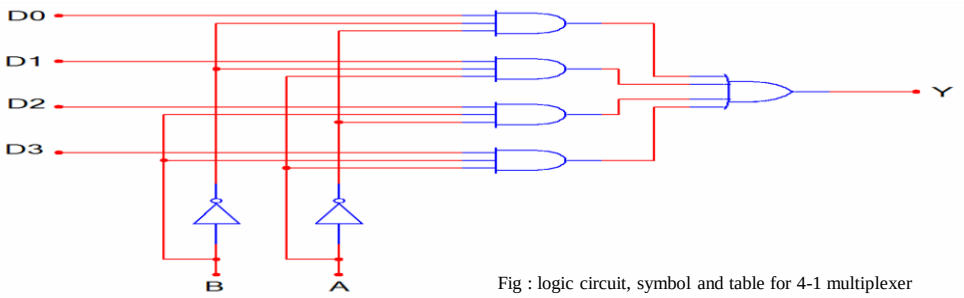
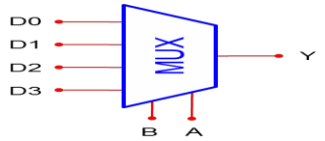


Fig : multiplexer block diagram

4to-1 Multiplexer (MUX)





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

Fig : logic circuit, symbol and table for 4-1 multiplexer

The Boolean expression for this 4-to-1 **Multiplexer** above with inputs D_0 to D_3 and data select lines A, B is given as:
$Y = A'B'D_0 + A'B\,D_1 + AB'D_2 + ABD_3$

Demultiplexer

- A DEMUX is a digital switch with a single input (source) and a multiple outputs (destinations).
- The select lines determine which output the input is connected to.
- DEMUX Types

Implementing Boolean functions with Multiplexers

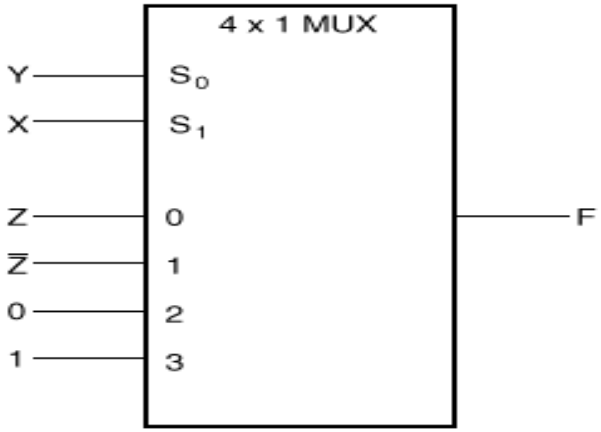
- Any Boolean function of *n* variables can be implemented using a 2^{n-1}-to-1 multiplexer. A MUX is basically a decoder with outputs ORed together, hence this isn't surprising.
- The SELECT signals generate the minterms of the function.
- The data inputs identify which minterms are to be combined with an OR.

Example

- ✓ $F(X,Y,Z) = X'Y'Z + X'YZ' + XYZ' + XYZ = \Sigma m(1,2,6,7)$
- ✓ There are n=3 inputs, thus we need a **2^2-to-1 MUX**
- ✓ **The first n-1 (=2) inputs serve as the selection lines**

X	Y	Z	F	
0	0	0	0	F = Z
0	0	1	1	
0	1	0	1	F = $\overline{Z}$
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

The Other Example

- Consider F(A,B,C) = ∑m(1,3,5,6). We can implement this function using a 4-to-1 MUX as follows.
- The index is ABC. Apply A and B to the S_1 and S_0 selection inputs of the MUX (A is most sig, S_1 is most sig.)
- Enumerate function in a truth table.

When A=B=0, F=C

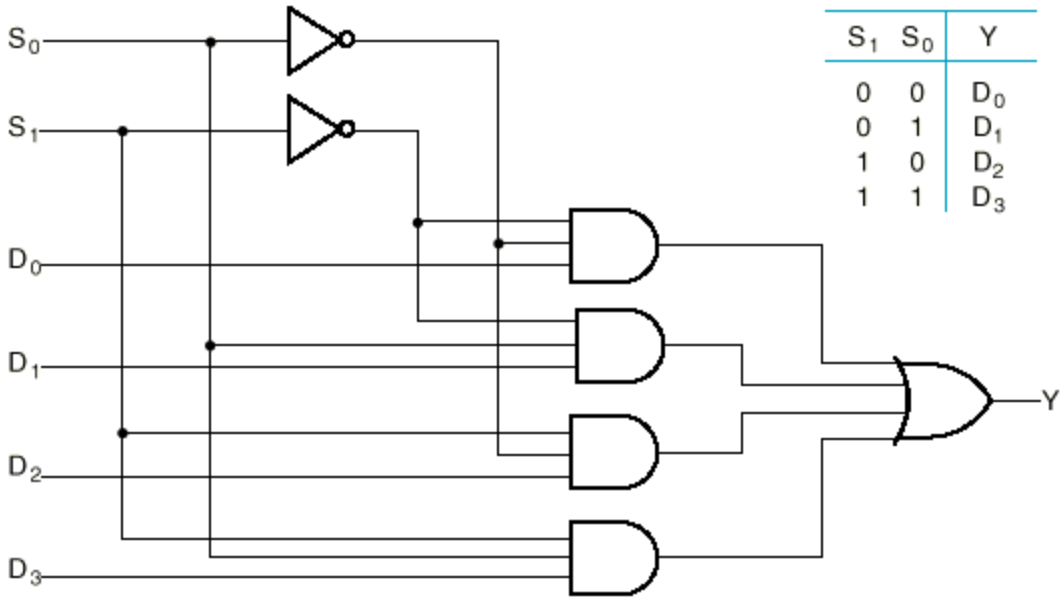
When A=0, B=1, F=C

When A=1, B=0, F=C

When A=B=1, F=C'

A	B	C		F
0	0	0		0
0	0	1		1
0	1	0		0
0	1	1		1
1	0	0		0
1	0	1		1
1	1	0		1
1	1	1		0

MUX implementation of F(A,B,C) = ∑m(1,3,5,6)



Function table

S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

1-to-4 De-Multiplexer (DEMUX)

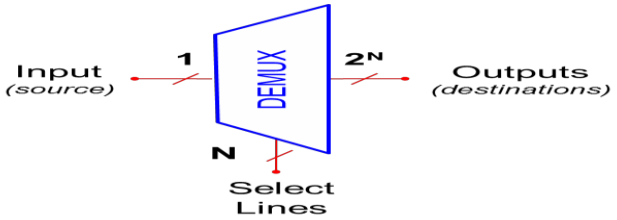
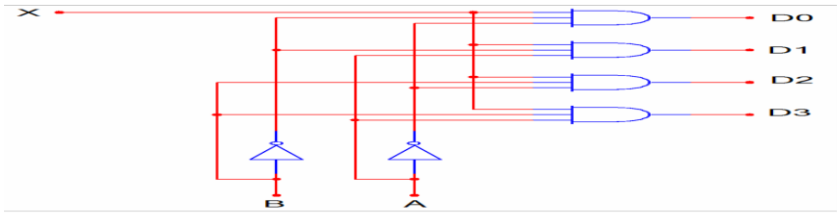
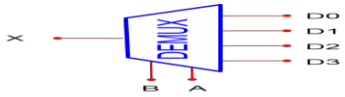


Fig : Demultiplexer circuit symbol





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

Fig: logic circuit for 1-4 demultiplexer, symbol and table

- **Parity Bit Generator**
  - The most common error detection code used is the parity bit.
  - A parity bit is an extra bit included with a binary message to make the total number of 1's either odd or even.
  - A parity bit added to *n*-bit code produces (*n*+1)-bit code with an odd (or even) count of 1s
  - Odd Parity bit: count of 1s in (*n*+1)-bit code is odd
    - So use an even function to generate the odd parity bit
  - Even Parity bit: count of 1s in (*n*+1)-bit code is even
    - So use an odd function to generate the even parity bit