Magnitude Comparator

A magnitude Comparator is a combinational circuit that compares two numbers, A and B, and determines their relative magnitudes. The out come of the comparison is specified by three binary variables that indicate whether $A>B$, $A=B$, or $A<B$.

The circuit for comparing two n-bit numbers has 2^{2n} entries in the truth table and becomes too cumbersome even with $n=3$.

Consider two numbers, A and B, with four digits each. write the coefficients of the numbers with descending significance

$$A = A_3\ A_2\ A_1\ A_0$$
$$B = B_3\ B_2\ B_1\ B_0$$

The two numbers are *equal* if all pairs of significant digits are *equal*: $A_3=B_3$ and $A_2=B_2$ and $A_1=B_1$ and $A_0=B_0$. The *equality* relation of each pair of bits can be expressed logically with an exclusive-NOR function as

$$x_i = A_i B_i + \bar{A}_i \bar{B}_i \qquad \text{for } i=0,1,2,3$$

where $x_i=1$ only if the pair of bits in position i are *equal* (i.e., if both are 1 or both are 0). For the *equality* condition to exist, all x_i variables must be *equal* to 1. This dictates an AND operation of all variables:

$$(A=B) = x_3 x_2 x_1 x_0$$

The binary variable $(A=B)$ is *equal* to 1 only if all pairs of digits of the two numbers are *equal*.

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 (MSB). 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 $A < B$.

$$(A>B) = A_3\bar{B}_3 + x_3A_2\bar{B}_2 + x_3x_2A_1\bar{B}_1 + x_3x_2x_1A_0\bar{B}_0$$

$$(A<B) = \bar{A}_3B_3 + x_3\bar{A}_2B_2 + x_3x_2\bar{A}_1B_1 + x_3x_2x_1\bar{A}_0B_0$$

The symbols $(A>B)$ and $(A<B)$ are binary output variables that are *equal* to 1 when $A>B$ or $A<B$, respectively.

The logic diagram of the 4-bit magnitude comparator is show in the next page (P55).

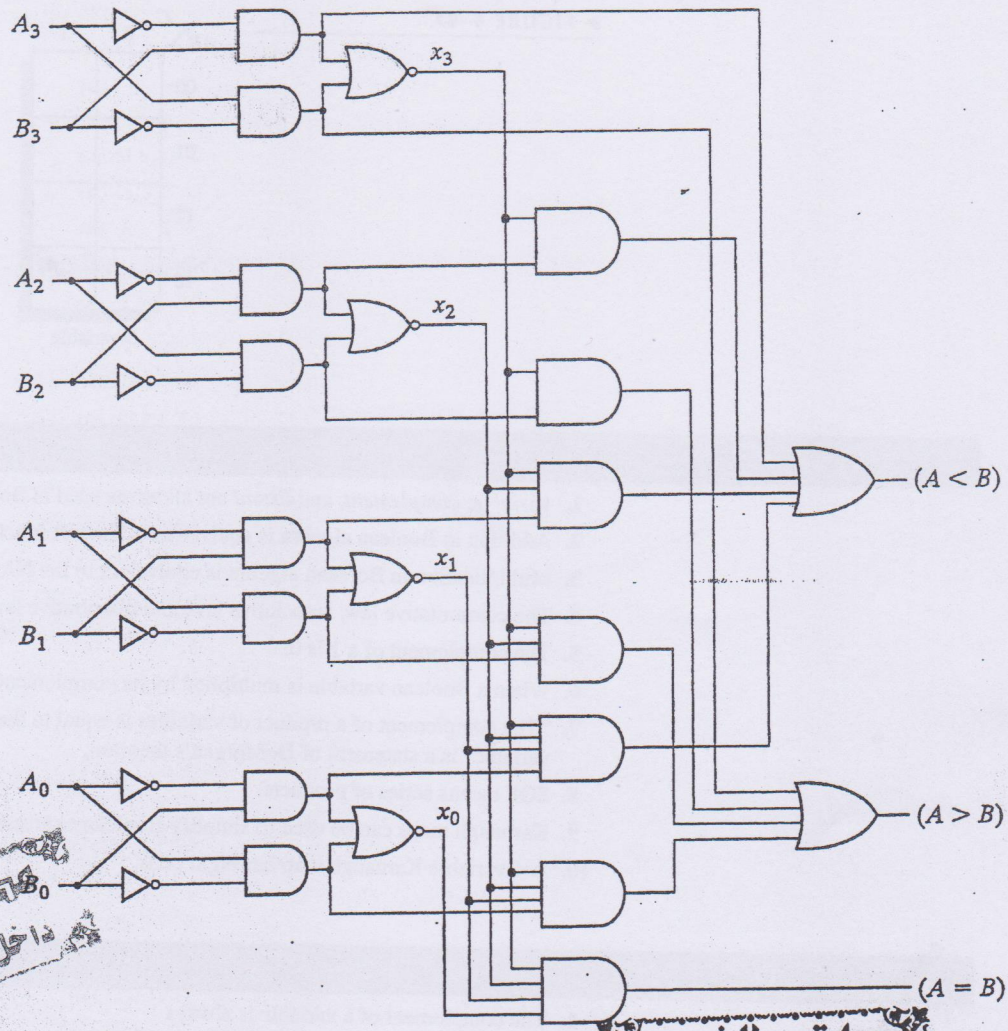


FIGURE 4-17
4-Bit Magnitude Comparator

that are needed to generate the equal output. The logic diagram of the 4-bit magnitude parator is shown in Fig. 4-17. The four x outputs are generated with exclusive-NOR circuit applied to an AND gate to give the output binary variable $(A = B)$. The other two output the x variables to generate the Boolean functions listed previously. This is a multilevel ir mentation and has a regular pattern. The procedure for obtaining magnitude comparator ci for binary numbers with more than four bits is obvious from this example.

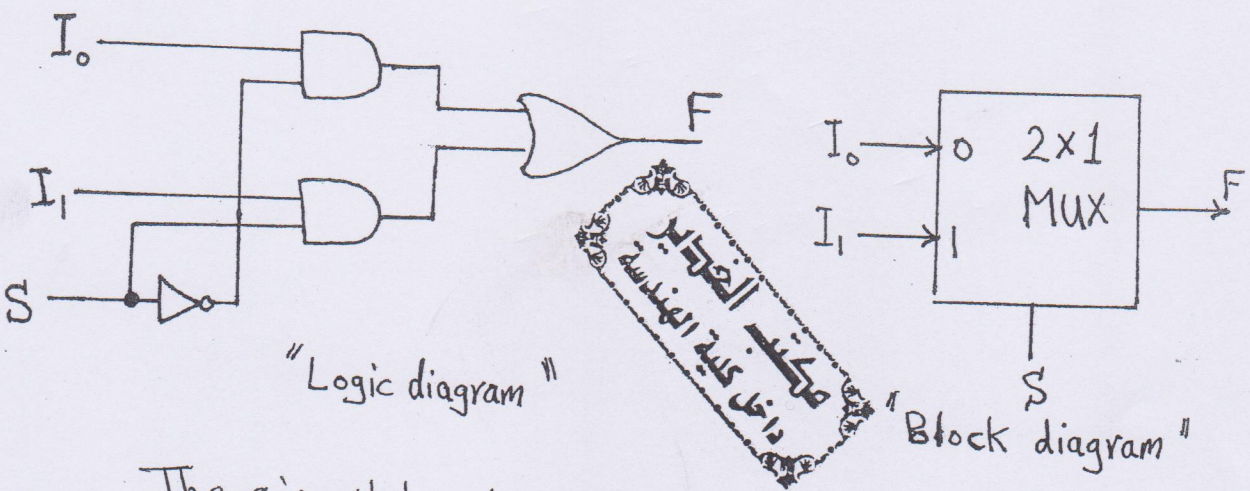
4-8 DECODERS

Discrete quantities of information are represented in digital systems by binary codes. A ry code of n bits is capable of representing up to 2^n distinct elements of coded informati *decoder* is a combinational circuit that converts binary information from n input lines to a imum of 2^n unique output lines. If the n-bit coded information has unused combi decoder may have fewer than 2^n outputs.

Multiplexers

It is a combinational circuit 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 a set of selection lines. Normally, there are 2^n input lines and n selection lines whose bit combinations determine which input is selected.

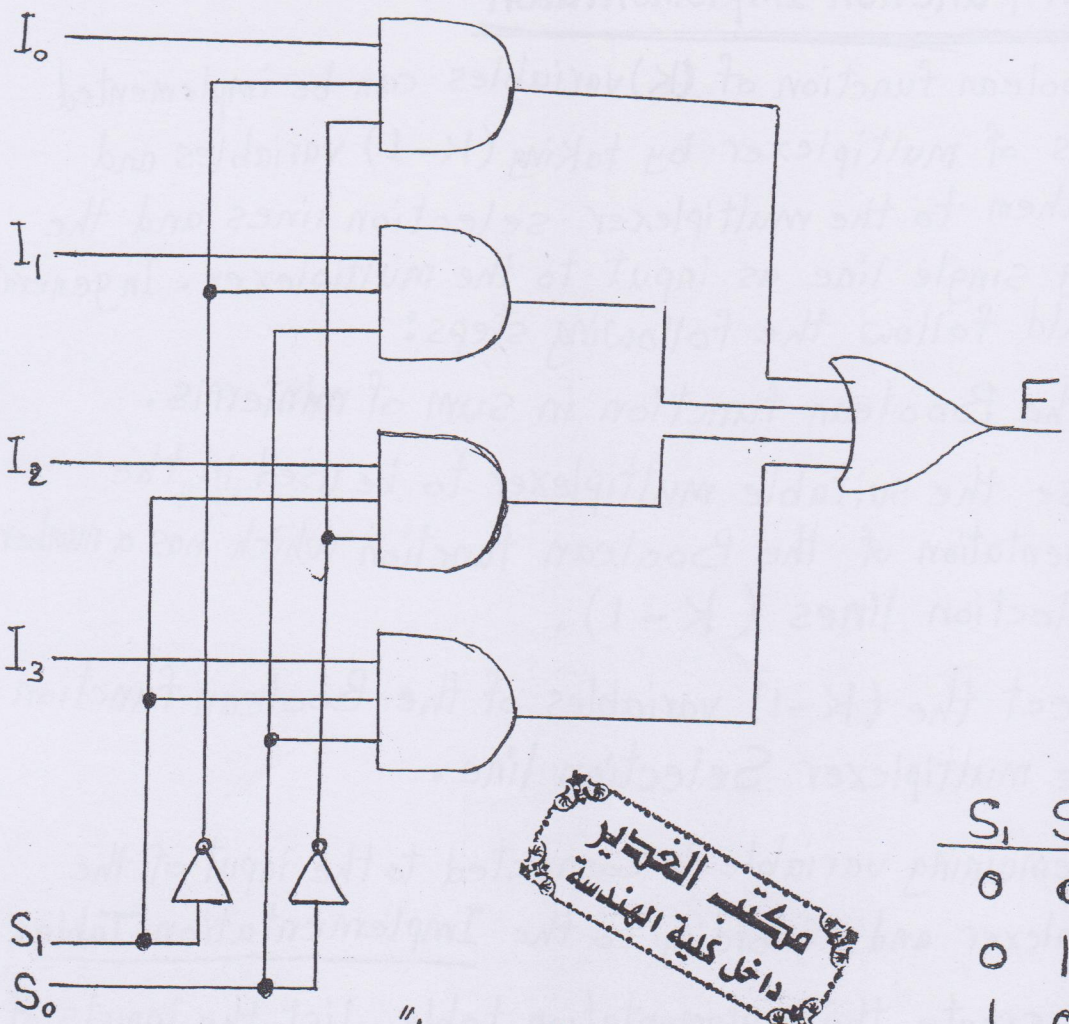
A 2-to-1-line multiplexer connects one of two 1-bit sources to a common destination as shown below.



The circuit has two data input lines, one output line, and one selection line S. When $S=0$, the upper AND gate is enabled and I_0 has a path to the output. When $S=1$, the lower AND gate is enabled and I_1 has a path the output. The multiplexer acts like an electronic switch that selects one of two sources.

A 4-to-1-line multiplexer is shown in the fdlowing figure. Each of the four inputs, I_0 through I_3, is applied to one input of an AND gate.

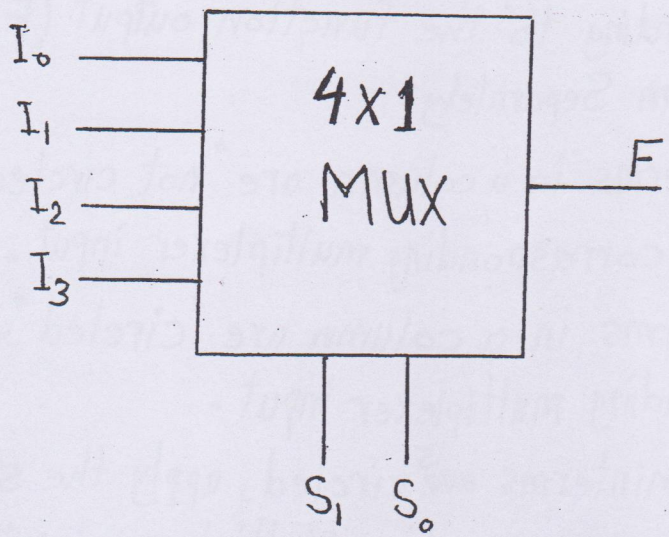
The size of a multiplexer is specified by the number 2^n of its data input lines and the single output line. The n selection lines are implied from the 2^n data lines.



"Logic diagram"

S_1	S_0	F
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

"Function Table"



"Block diagram"

Boolean Function Implementation

A Boolean function of (K) variables can be implemented by means of multiplexer by taking (K-1) variables and connect them to the multiplexer selection lines and the remaining single line as input to the multiplexer. In general you should follow the following steps:

1- Put the Boolean function in sum of minterms.

2- Choose the suitable multiplexer to be used in the implementation of the Boolean function which has a number of selection lines (K-1).

3- Connect the (K-1) variables of the Boolean function to the multiplexer Selection line.

4- The remaining variable is connected to the input of the multiplexer and according to the Implementation Table.

To create the implementation table, list the inputs of the multiplexer and under them list all the minterms in rows according to the Selected input variable. Circle the minterms according to the function output (F=1) and inspect each column Separately:

i- If all the minterms in a column are "not circled", apply 0 to the corresponding multiplexer input.

ii- If all the minterms in a column are "circled", apply 1 to the corresponding multiplexer input.

iii- If one of the minterms is circled, apply the status of the row to the corresponding Multiplexer input.

Example: Implement the following Boolean function using 4x1 Multiplexer, $F=\sum(1,3,5,6)$.

Soln: Number of variables (function) $K=3$ (A,B,C).

Number of Selection lines $n=2$.

	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

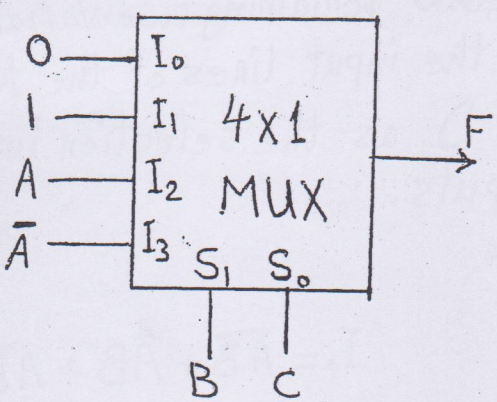
We will choose B and C as Selection lines.

$I_0=\bar{B}\bar{C}$, $I_1=\bar{B}C$

$I_2=B\bar{C}$, $I_3=BC$

Implementation Table

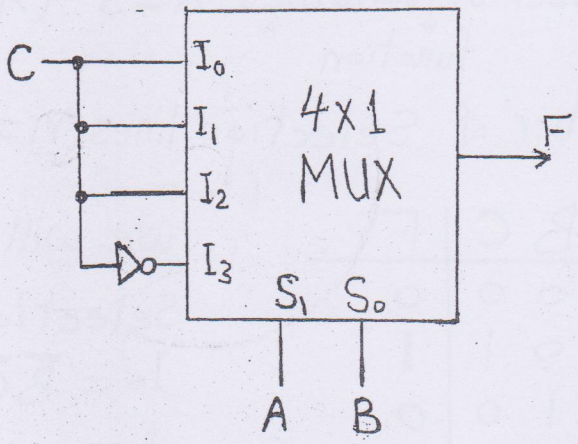
	I_0	I_1	I_2	I_3
$\bar{A}$	0	(1)	2	(3)
A	4	(5)	(6)	7
	0	1	A	$\bar{A}$



Now if A and B are choosen a selection lines, then the Implementation Table will be as follows:

	I_0	I_1	I_2	I_3
$\bar{C}$	0	2	4	(6)
C	(1)	(3)	(5)	7
	C	C	C	$\bar{C}$

$I_0 = \bar{A}\bar{B}$, $I_1 = \bar{A}B$, $I_2 = A\bar{B}$, $I_3 = AB$



Example: Implement the following function using 4x1 Multiplexer and any required logic gates:

$F(A,B,C,D) = \Sigma(0,1,3,4,8,9,15)$.

Soln:

Using 4x1 Mux, this means 2 variables will be as selection lines and the ~~inputs~~ remaining two variables will be connected to the input lines of the MUX.

We choose C and D as the selection lines, while A and B as inputs.

	I_0	I_1	I_2	I_3
$\bar{A}\bar{B}$	(0)	(1)	2	(3)
$\bar{A}B$	(4)	5	6	7
$A\bar{B}$	(8)	(9)	10	11
AB	12	13	14	(15)
	$\bar{A}+\bar{B}$	$\bar{B}$	0	$A \odot B$

$I_0 = \bar{A}\bar{B} + \bar{A}B + A\bar{B}$

$= \bar{A} + A\bar{B}$

$\boxed{I_0 = \bar{A} + \bar{B}}$

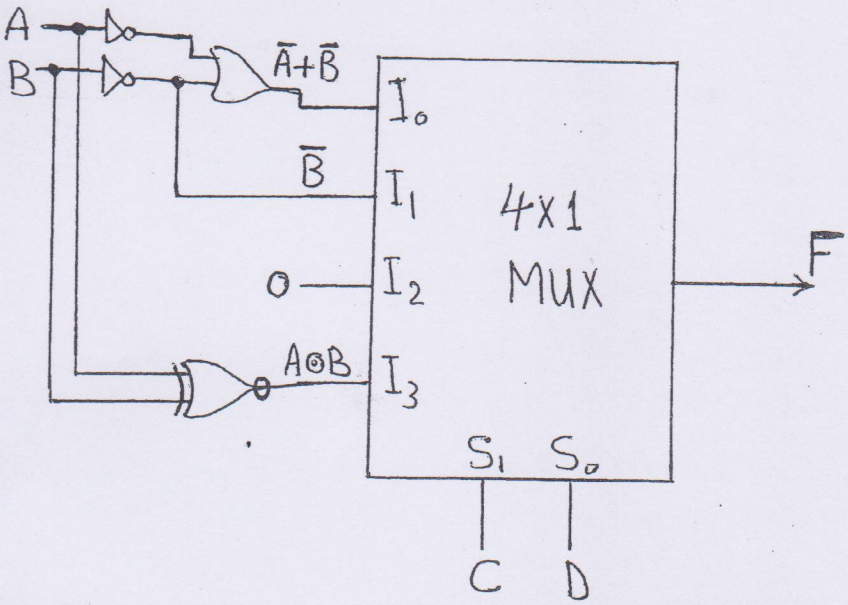
$I_1 = \bar{A}\bar{B} + A\bar{B}$

$\boxed{I_1 = \bar{B}}$

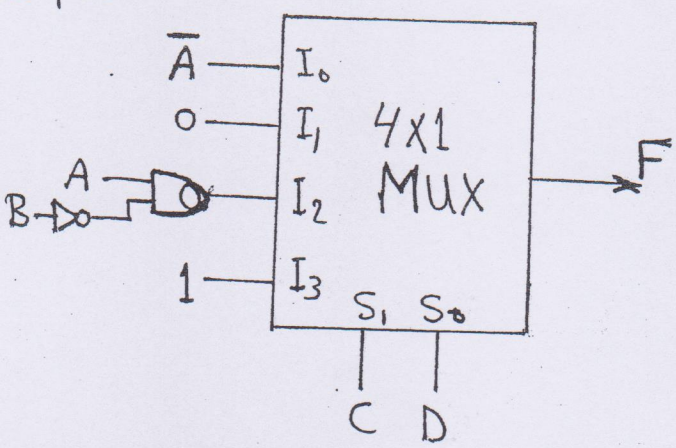
$\boxed{I_2 = 0}$

$I_3 = \bar{A}\bar{B} + AB$

$\boxed{I_3 = A \odot B}$



Ex: Find the minimized Boolean function Implemented by the following Multiplexer.



Sol^n

$$F = I_0 \cdot \bar{S}_1\bar{S}_0 + I_1 \cdot \bar{S}_1 S_0 + I_2 \cdot S_1\bar{S}_0 + I_3 \cdot S_1 S_0$$
$$= \bar{A}\bar{C}\bar{D} + 0 \cdot \bar{C}D + A\bar{B}C\bar{D} + 1 \cdot CD$$
$$= \bar{A}\bar{C}\bar{D} + A\bar{B}C\bar{D} + CD$$
$$F = \bar{A}\bar{C}\bar{D} + A\bar{B}C + CD$$