

Combinational Logic with MSI and LSI

Contents:

- ✓ Binary Parallel Adder.
- ✓ BCD Adder.
- ✓ Magnitude Comparator.
- ✓ Decoders.
- ✓ Encoders.
- ✓ Multiplexer
- ✓ Demultiplexer.
- ✓ ROM
- ✓ PLA

Prepared By

Mohammed Abdul kader
Assistant Professor, EEE, IIUC

Subscript i	4	3	2	1	
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}

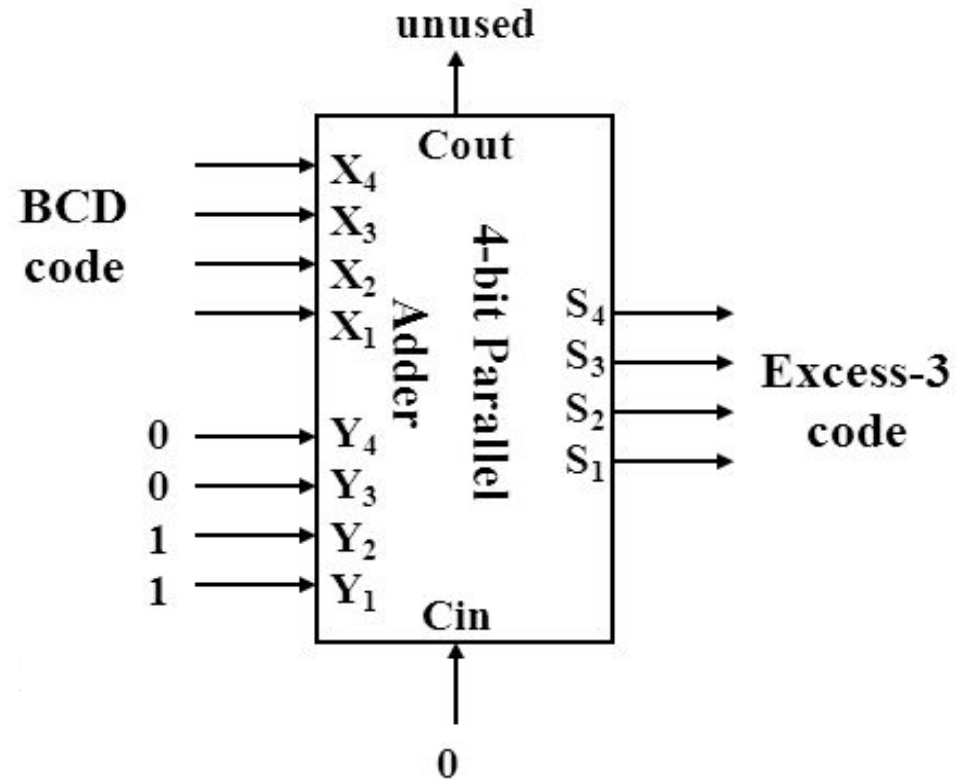
Fig. 2: Addition of A=1011 and B=0011

The diagram shows four Full Adder (FA) blocks arranged horizontally. Each block has two inputs at the top and one output at the bottom. The inputs are labeled A_1, B_1 for the first block, A_2, B_2 for the second, A_3, B_3 for the third, and A_4, B_4 for the fourth. The outputs at the bottom are labeled S_1, S_2, S_3, S_4 respectively. Carry signals are shown as arrows between blocks: C_1 enters the first block from the right, C_2 goes from the first to the second, C_3 from the second to the third, and C_4 from the third to the fourth. A final carry output C_5 exits the fourth block to the left.

3

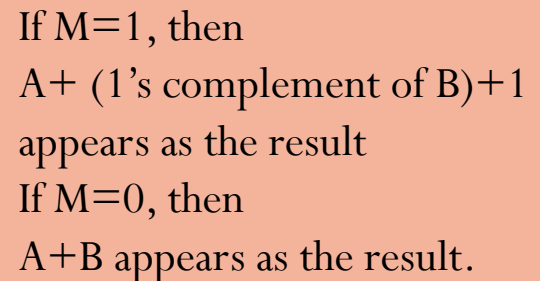
Example 5-1: Design a BCD-to-excess-3 code converter using a 4-bit full adders MSI circuit.

$$\text{Excess-3 code} = \text{BCD Code} + (0011)_2$$



Problem 5-1: Design an excess-3-to-BCD code converter using a 4-bit full adders MSI circuit

Hints: BCD code = Excess-3 Code $- (0011)_2 \Rightarrow$ BCD Code = Excess-3 Code $+ (1101)_2$

[illegible]

Carry Propagation:

- Since each bit of the sum output depends on the value of the input carry, the value of S_i in any given stage in the adder will be in its steady-state final value only after the input carry to that stage has been propagated.
- Consider output S_4 , inputs A_4 and B_4 reach a steady value as soon as input signals are applied to the adder. But input carry C_4 does not settle to its final steady-state value until C_3 is available in its steady-state value. Similarly, C_3 has to wait for C_2 , and soon down to C_1 . Thus only after carry propagates through all stages will be the last output S_4 and carry C_5 settle to their final steady-state value.

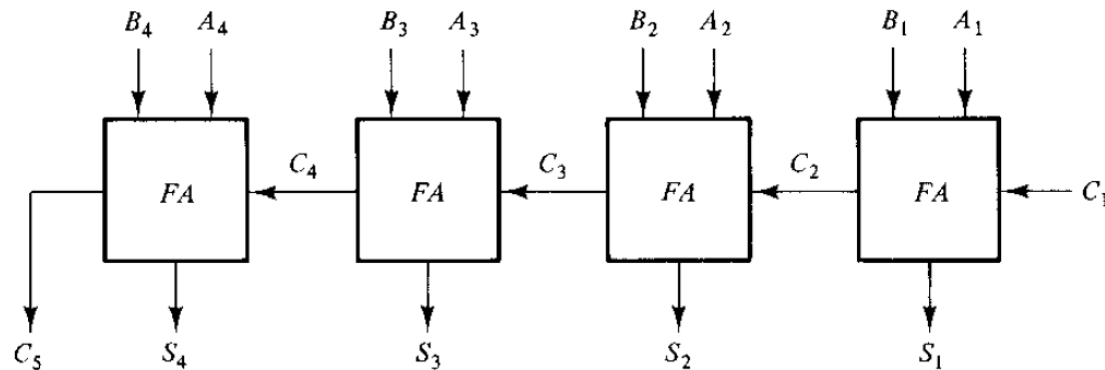



Fig. 5: 4-bit parallel adder

Carry Propagation (Cont...):

- The number of gate levels for the carry propagation can be found from the circuit of the full-adder.
- 
- The diagram shows the input lines for a full adder. On the left, an input line is labeled A_i . It connects to a logic gate (likely an AND gate) which has another input line labeled P_i . The output of this gate is a curved line that continues to the right.

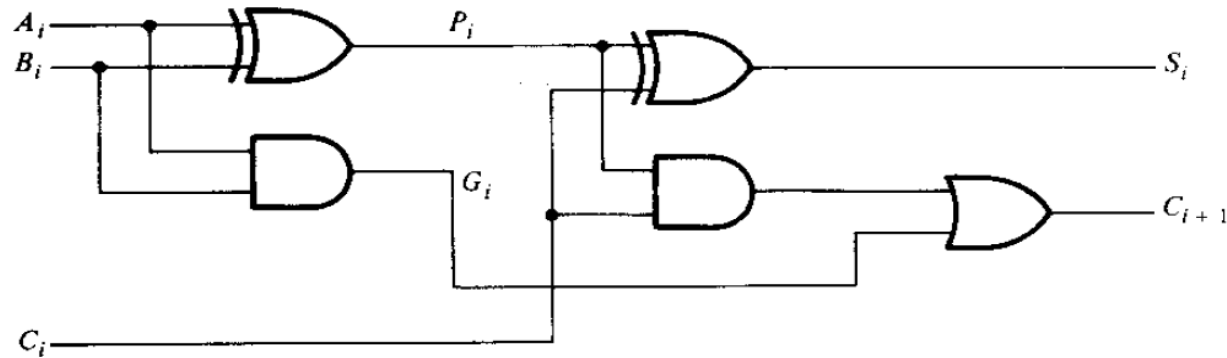


Fig. 6: Full adder

- The signals at **Pi** and **Gi** settle to their steady-state value after the propagation through the irrespective gates. These two signals are common to all FAs and depend only on the input augend and addend bits.
- The signals from the input carry C_i , to the output carry C_{i+1} propagates through an AND and an OR gate, which constitute two gate levels.
- For an n-bit parallel adder, there are 2n gate levels for the carry to propagate through.

- There are several techniques for reducing the carry propagation time in a parallel adder.

➤ From previous circuit (Fig. 6), we can define two new variables,

$$\begin{aligned} P_i &= A_i \oplus B_i \\ G_i &= A_i B_i \end{aligned}$$

- G_i is called a **carry generate** and it produces an output carry when both A_i and B_i are one, regardless of the input carry. P_i is called a **carry propagate** because it is the term associated with the propagation of the carry from C_i to C_{i+1}

$$C_2 = G_1 + P_1 C_1$$

$$C_4 = G_3 + P_3 C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_1$$



9

10

BCD Adder

- A BCD adder is a circuit that adds two BCD digits in parallel and produces a sum digital so in BCD.
- Consider the arithmetic addition of two decimal digits in BCD, together with a previous carry from a previous stage. The output sum cannot be greater than $9+9+1=19$, the 1 in the sum being an input carry.
- Suppose, we apply two BCD digits to a 4-bit binary adder. The adder will form the sum in binary and produce a result which may range from 0 to 19.

- $$C = K + Z_8 Z_4 + Z_8 Z_2$$

C=1, when binary sum is greater than 1001. binary 0110 is added to the binary sum through the bottom 4-bit binary adder to convert the binary sum into BCD sum.



- $$B = B_3 B_2 B_1 B_0$$

- $$x_i = A_i B_i + \dot{A}_i \dot{B}_i, \quad i=0,1,2,3$$

14

- For equity condition to exist, all x_i variables must be equal to 1.

$$(A=B) = x_3x_2x_1x_0$$

A and B will be equal if $x_3 = x_2 = x_1 = x_0 = 1$ or, 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 magnitude of pairs of significant digits starting from 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 unique digits is reached. If the digit of A is 1 and that of B is 0, we conclude $A > B$. If the corresponding digit of A is 0 and that of B is 1, we have that $A < B$.



$$(A > B) = A_3 B'_3 + x_3 A_2 B'_2 + x_3 x_2 A_1 B'_1 + x_3 x_2 x_1 A_0 B'_0$$

$$(A < B) = A'_3 B_3 + x_3 A'_2 B_2 + x_3 x_2 A'_1 B_1 + x_3 x_2 x_1 A'_0 B_0$$

If this term is 1, it means $A_3=1$ and $B_3=0$. As in most significant bit, A is greater than B, we conclude $A>B$.



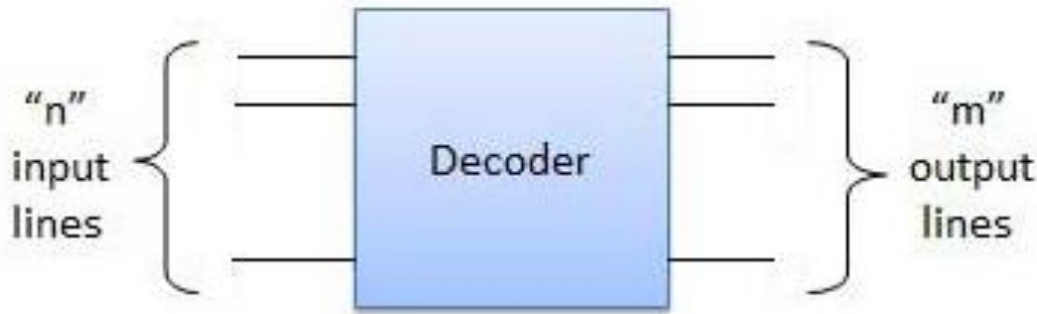




Decoders

Definition:

- A **decoder** is a combinational circuit that converts binary information from **n input lines** to a maximum of **2^n unique output lines**.
- If n-bit decoded information has unused or don't-care combinations, the decoder output will have less than 2^n outputs.
- The decoders presented here are called n-to-m line decoders where $m \leq 2^n$. Their purpose is to generate the 2^n (or less) minterms of n input variables.



Decoders

Application:

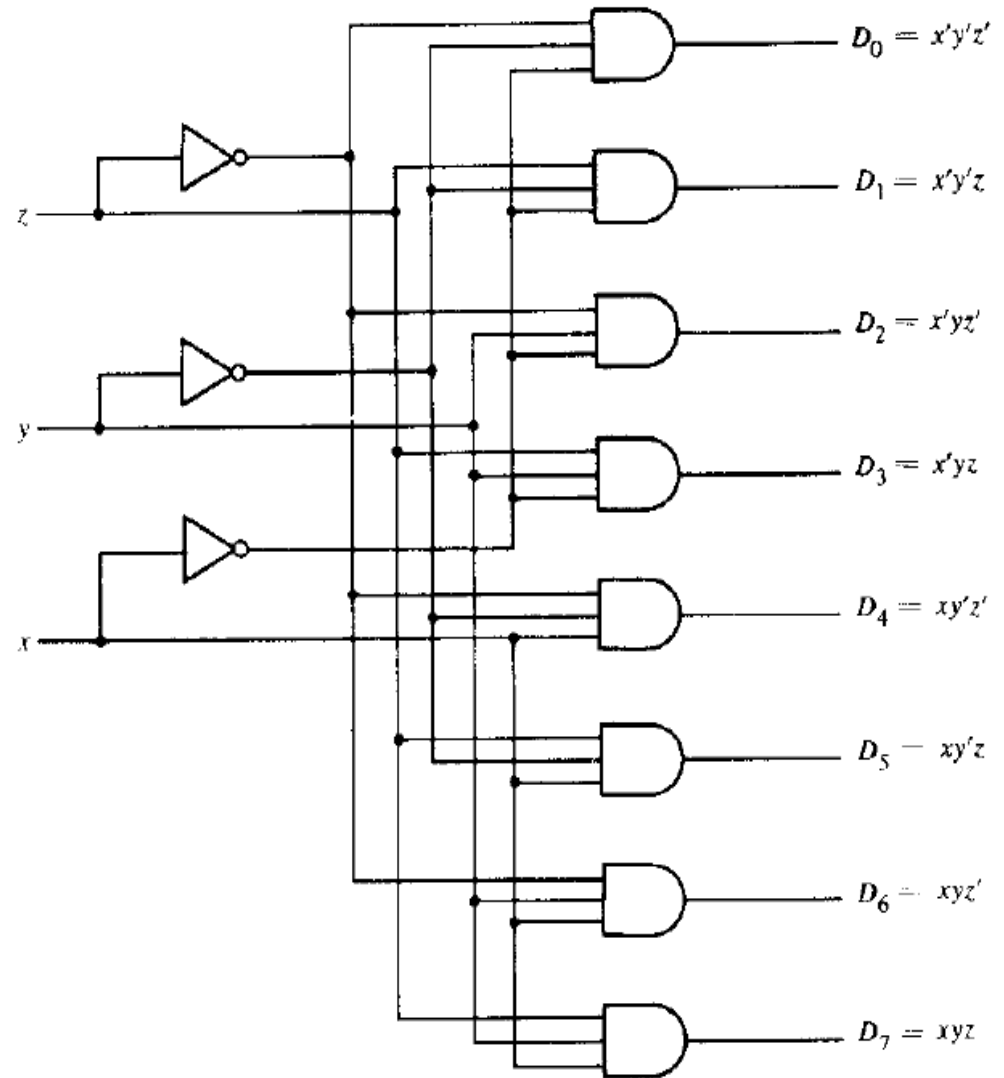
Decoders are greatly used in applications where the particular output or group of outputs to be activated only on the occurrence of a specific combination of input levels. Some important application of decoder circuit is given below-

- **Address Decoders:** Amongst its many uses, a decoder is widely used to decode the particular memory location in the computer memory system. Decoders accept the address code generated by the CPU which is a combination of address bits for a specific location in the memory.
- **Instruction Decoder:** Another application of the decoder can be found in the control unit of the central processing unit. This decoder is used to decode the program instructions in order to activate the specific control lines such that different operations in the ALU of the CPU are carried out.

Decoders

3 X 8 Decoder

- The three inputs are decoded into eight outputs, **each output representing one of the minterms** of the 3-input variables.
- A particular application of this decoder would be a **binary-to-octal conversion**. The input variable may represent a binary number and the outputs will then represent the eight digits in the octal number system



Decoders

Truth Table of 3 X 8 line decoder

From the truth table it is observed that the output variables are mutually exclusive because only one output can be equal to 1 at any one time. The output line whose value is equal to 1 represents the minterm equivalent of the binary number presently available in the input lines.

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

Example 5-2: Design a BCD-to-decimal decoder

[illegible]

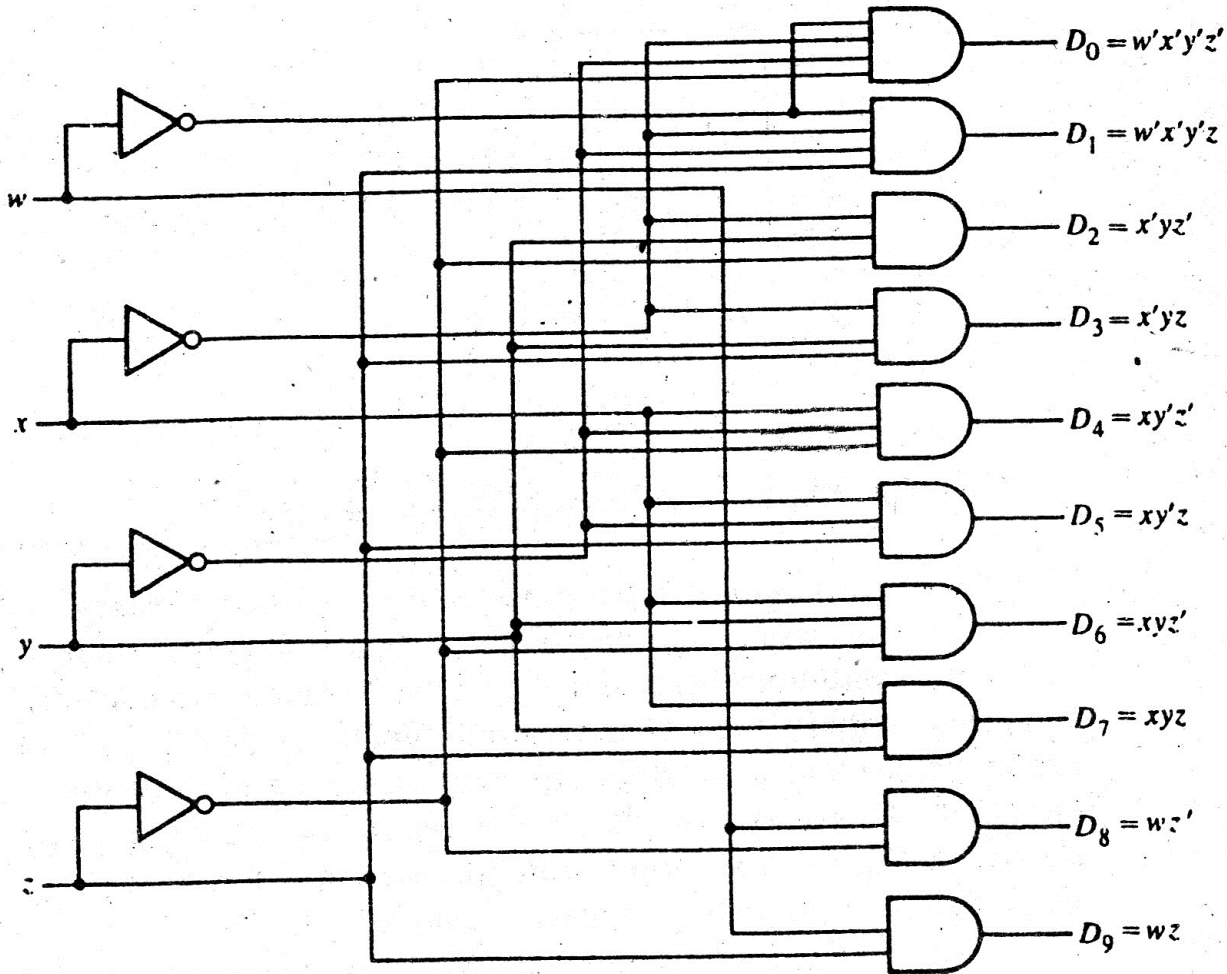
Decoders

- | | | | | | | |
|------|----|-------|-------|-------|-------|-------|
| | | yz | | y | | |
| | | 00 | 01 | 11 | 10 | |
| wx | 00 | D_0 | D_1 | D_3 | D_2 | |
| | 01 | D_4 | D_5 | D_7 | D_6 | |
| | 11 | X | X | X | X | } x |
| w | 10 | D_8 | D_9 | X | X | |
| | | z | | | | |

23

Decoders

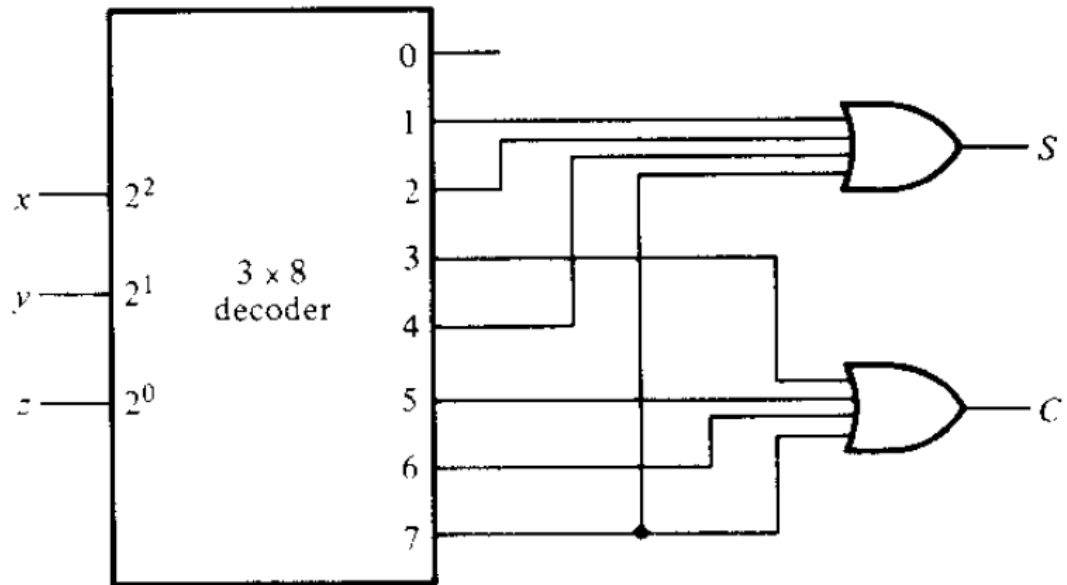
Logic diagram of BCD-to-decimal decoder



Example 5-3: Implement a full-adder circuit with a decoder and two OR gates

From the truth table of the full-adder circuit , we obtain the functions for this circuit in sum of minterms:

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



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

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

Encoders

An encoder is a digital function that produces a reverse operation from that of a decoder. An encoder has **2^n input lines and n output lines.**

The output lines generate the binary code corresponding to the input value.

Example: Octal to binary encoder which has 8 inputs and 3 outputs

Truth Table of binary to octal 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

From the truth table:

- Output bit z is 1 if octal digit is odd.
- Output y is 1 for octal digits 2,3,6 or 7.
- Output x is 1 for octal digits 4,5,6, or 7.

Boolean function of output variables

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

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

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

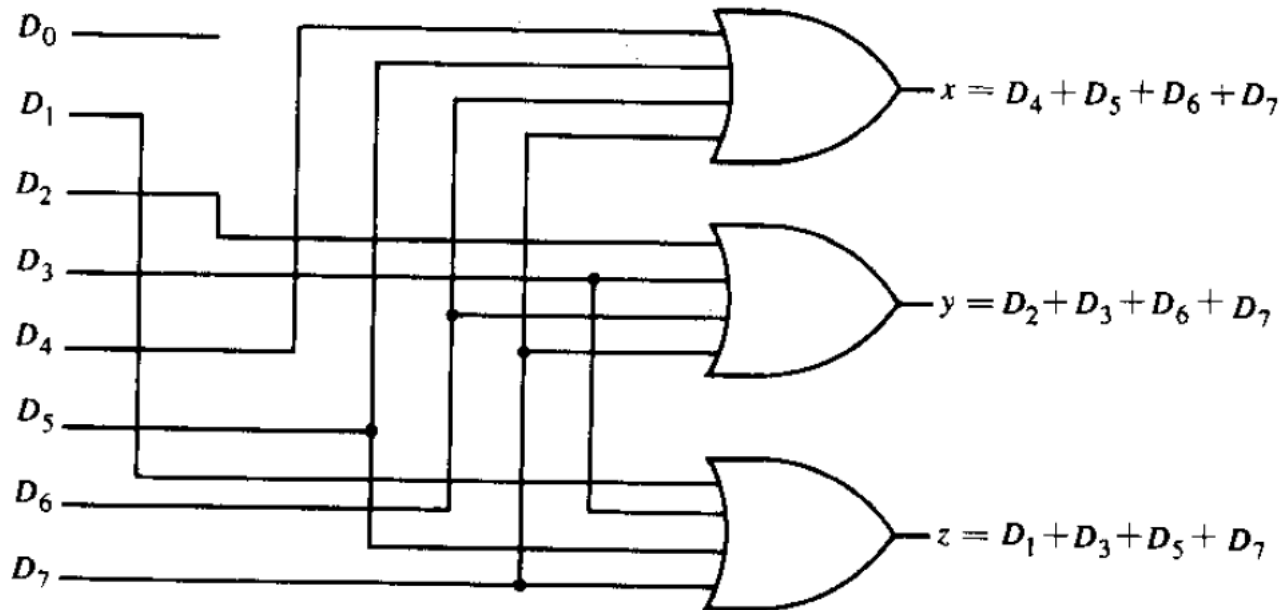
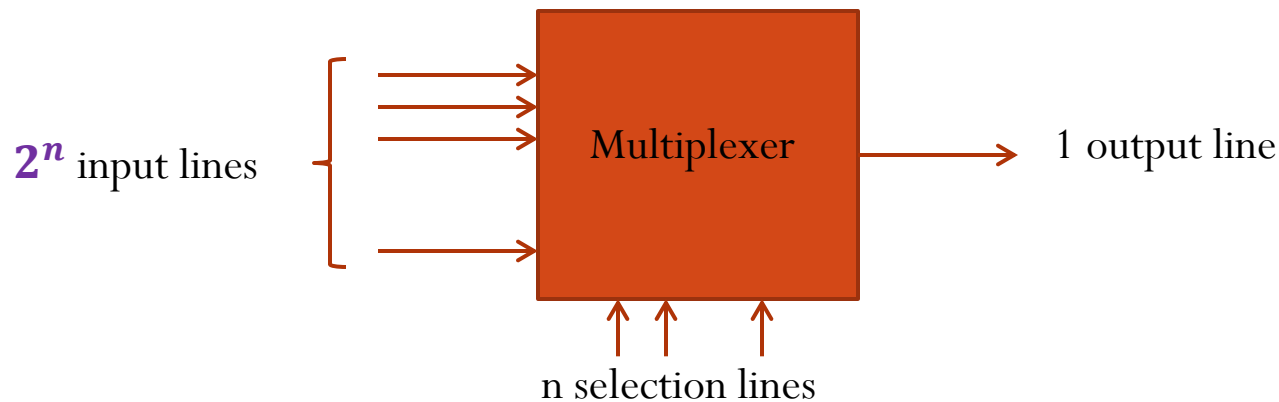


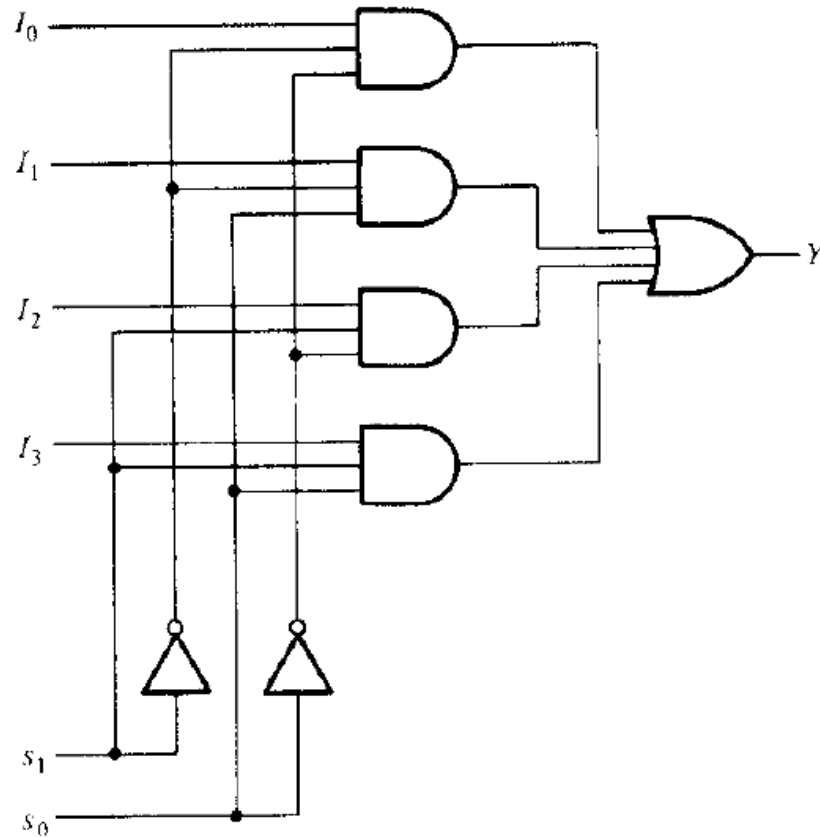
Fig. Octal to binary encoder

Multiplexer

- Multiplexer means transmitting a large number of information units over a smaller number of channels or lines.
- A digital multiplexer 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 lines 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 multiplexer is also called a data selector, since it selects one of many inputs and steers the binary information to the output line.



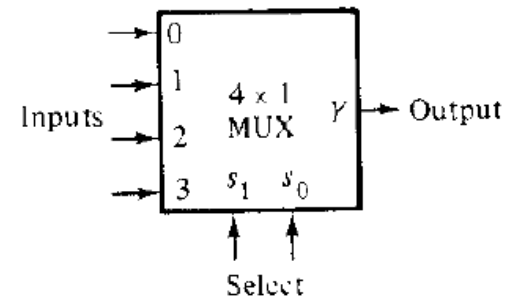
4-line to 1-line multiplexer



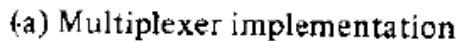
(a) Logic diagram

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

(b) Function table



(c) Block diagram

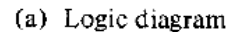


(b) Truth table

(c) Implementation table

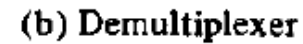
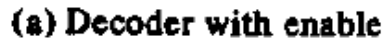
Implementation of Boolean function $F(A, B, C, D) = \sum(0, 1, 3, 4, 8, 9, 15)$ by multiplexer

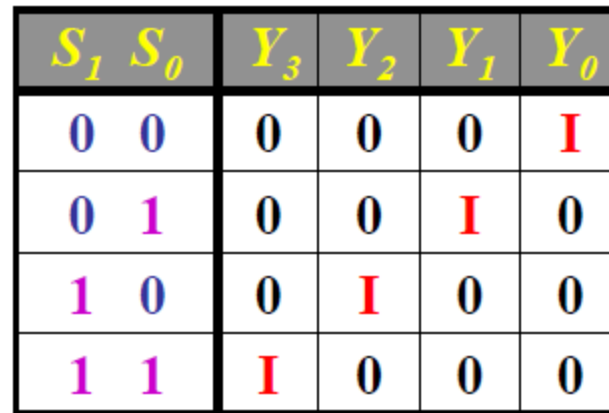
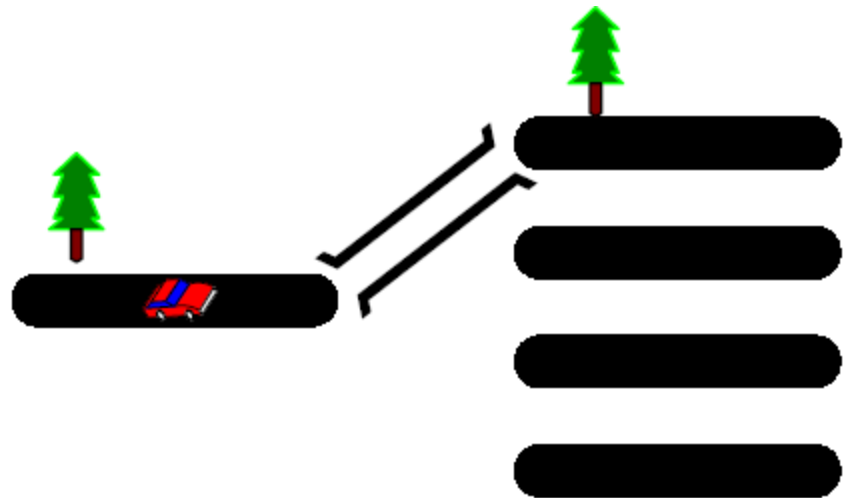
A de-multiplexer is a circuit that **receives information on a single line** and transmit this information on one of **2^n possible output lines**.



(b) Truth table

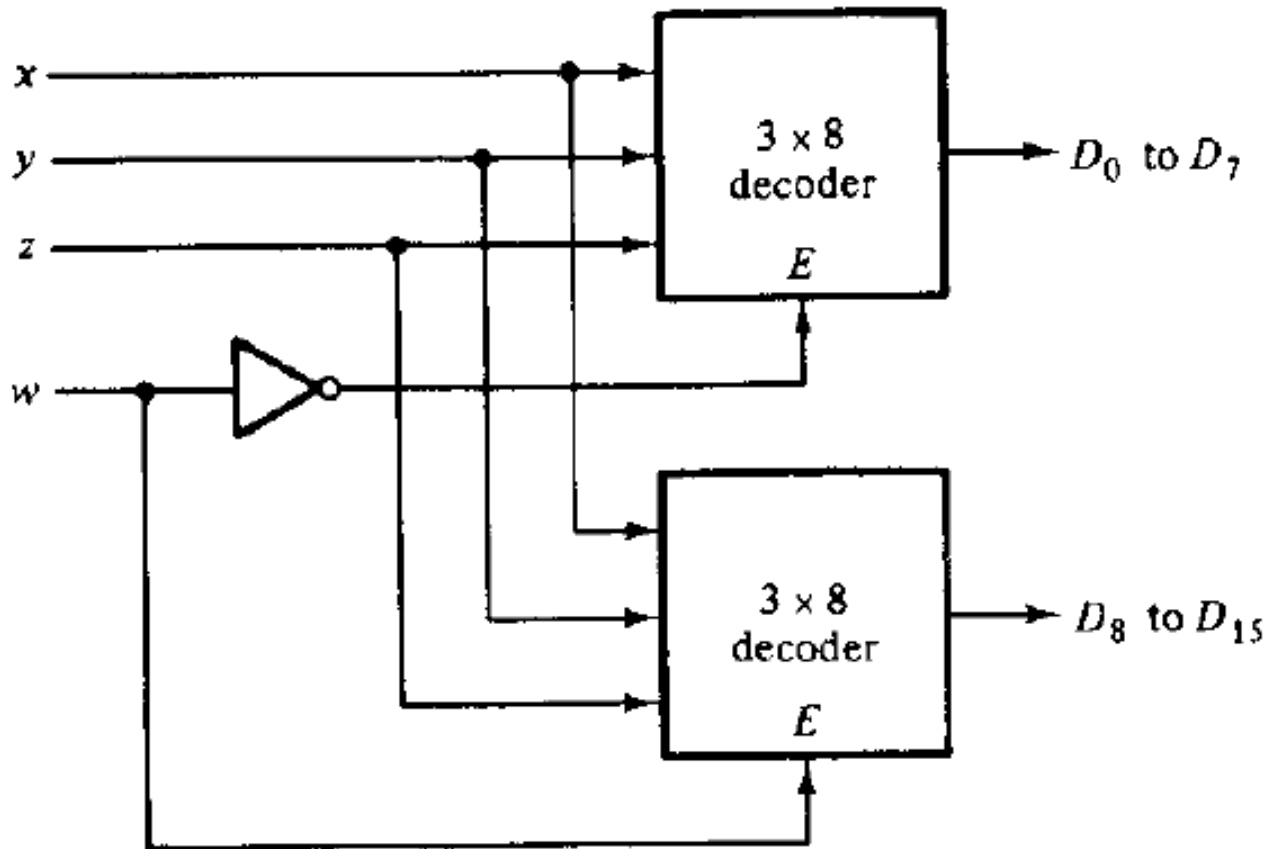
32





De-multiplexer

Decoder with enable/demultiplexer circuits can be connected together to form a larger decoder circuit. Fig. shows two 3X8 decoders with enable inputs connected to form a 4X16 decoder.

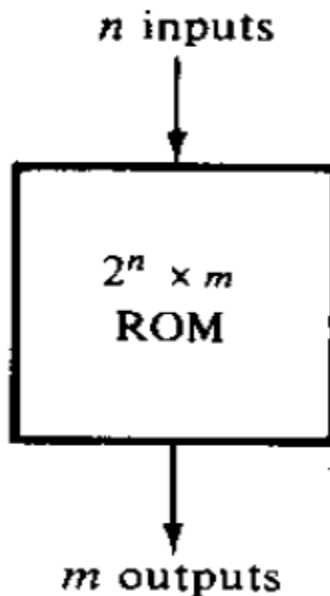


Read-Only Memory (ROM)

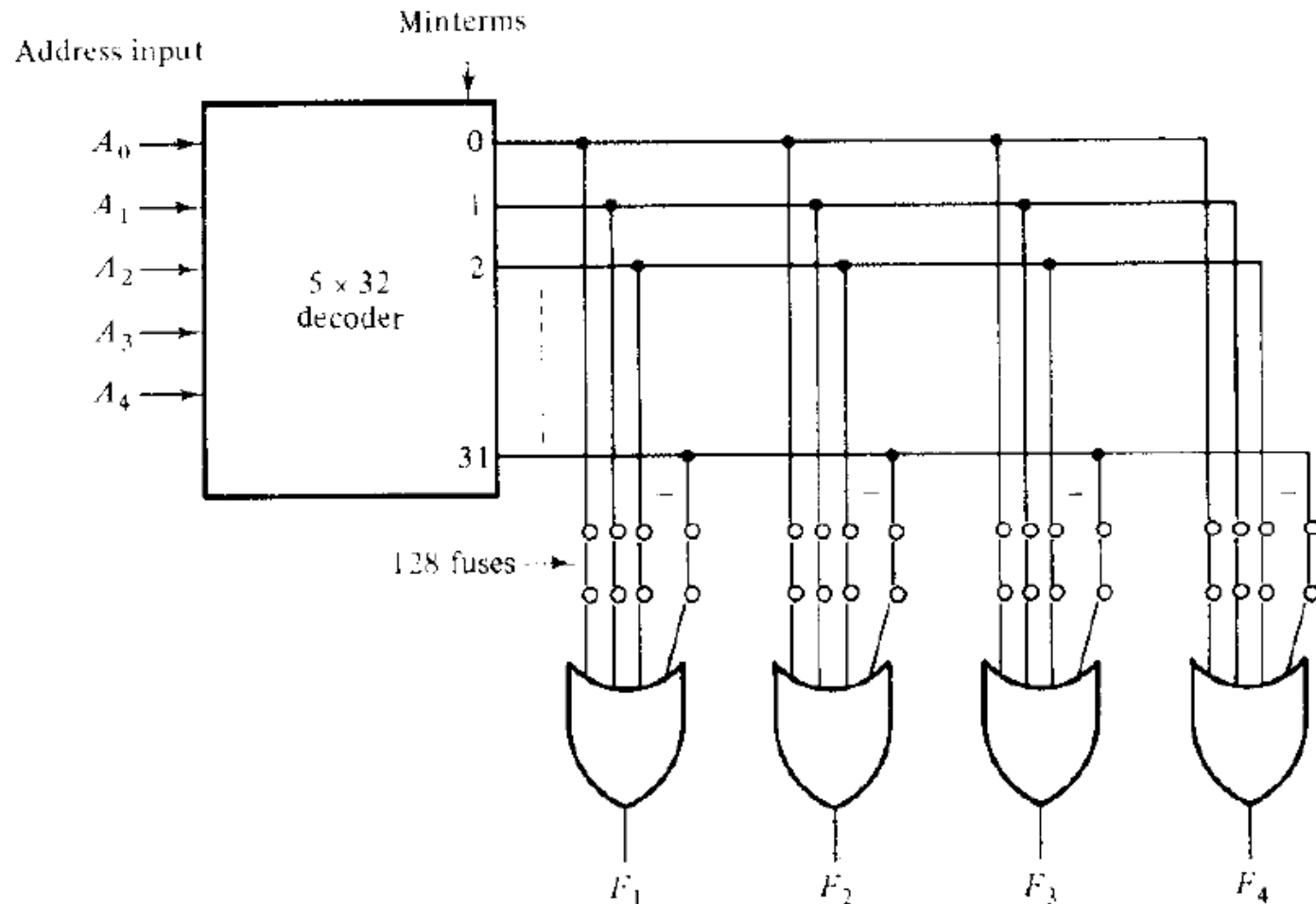
- A decoder generates the 2^n minterms of the n input variable. By inserting OR gates to sum the minterms of Boolean functions, we are able to generate any desired combinational circuit.
- A read-only memory (ROM) is a device that includes both the decoder and the OR gates within a single IC package. The connections between the outputs of the decoder and the inputs of the OR gates can be specified for each particular configuration by “programming” the ROM.
- A ROM is essentially a memory (or storage) device in which a fixed set of binary information is stored.
- The binary information must first be specified by the user and is then embedded in the unit to form the required interconnection pattern. ROM's come with special internal links that can be fused or broken. The desired interconnection for a particular application requires that certain links be fused to form the required circuit paths. Once a pattern is established for a ROM, it remain fixed even when power is turned off and then on again.

Read-Only Memory (ROM)

- A ROM consists of n input lines and m output lines.
- Each bit combination of input variables is called an address.
- Each bit combination that comes out of the output lines is called a word. The number of bits per word is equal to the number of output lines m .
- A ROM with n input lines has 2^n distinct addresses, so there are 2^n distinct words which are said to be stored in the unit.



Internally, the ROM is a combinational circuit with AND gates connected as a decoder and a number of OR gates equal to the number of outputs in the unit. Figure shows the logic construction of a 32X4 ROM.

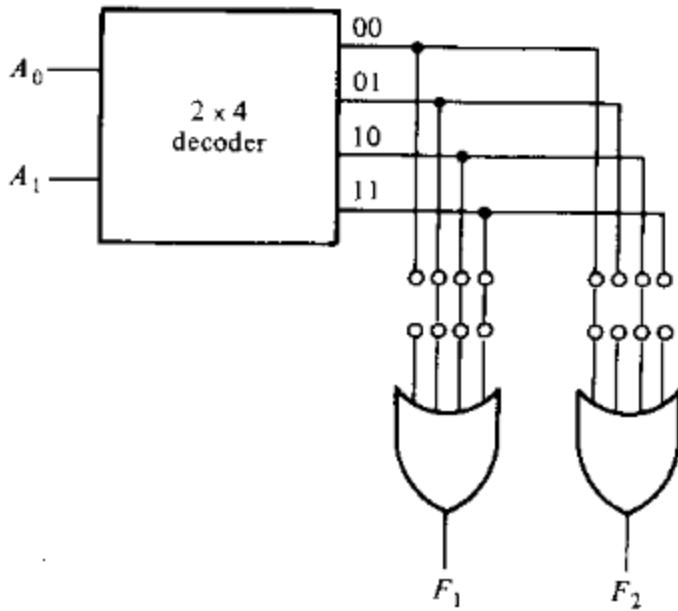


Combinational Logic Implementation

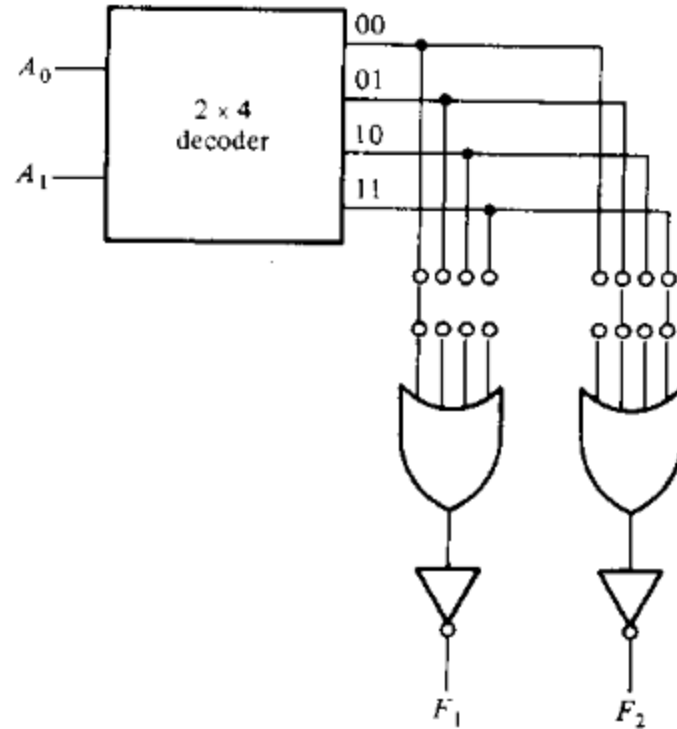
Implement the following combinational circuit with a 4X2 ROM

$$F_1(A_1, A_0) = \Sigma(1, 2, 3)$$

$$F_2(A_1, A_0) = \Sigma(0, 2)$$



ROM with AND-OR gates



ROM with AND-OR-INVERT gates

Read-Only Memory (ROM)

Combinational Logic Implementation

Design a combinational circuit using a ROM. The circuit accepts a 3-bit number and generates an output binary number equal to the square of the input number.

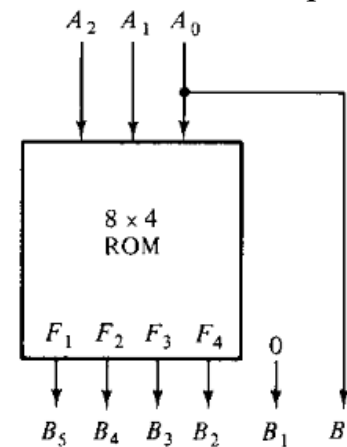
Solution:

Output B0 is always equal to input A0; so there is no need to generate B0 with a ROM since it is equal to an input variable. Moreover, output B1 is always 0, so this outputs is always known.

Truth Table

Inputs			Outputs						Decimal
A_2	A_1	A_0	B_5	B_4	B_3	B_2	B_1	B_0	
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1	1
0	1	0	0	0	0	1	0	0	4
0	1	1	0	0	1	0	0	1	9
1	0	0	0	1	0	0	0	0	16
1	0	1	0	1	1	0	0	1	25
1	1	0	1	0	0	1	0	0	36
1	1	1	1	1	0	0	0	1	49

Implementation by ROM



(a) Block diagram

A_2	A_1	A_0	F_1	F_2	F_3	F_4
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	1
0	1	1	0	0	1	0
1	0	0	0	1	0	0
1	0	1	0	1	1	0
1	1	0	1	0	0	1
1	1	1	1	1	0	0

(b) ROM truth table

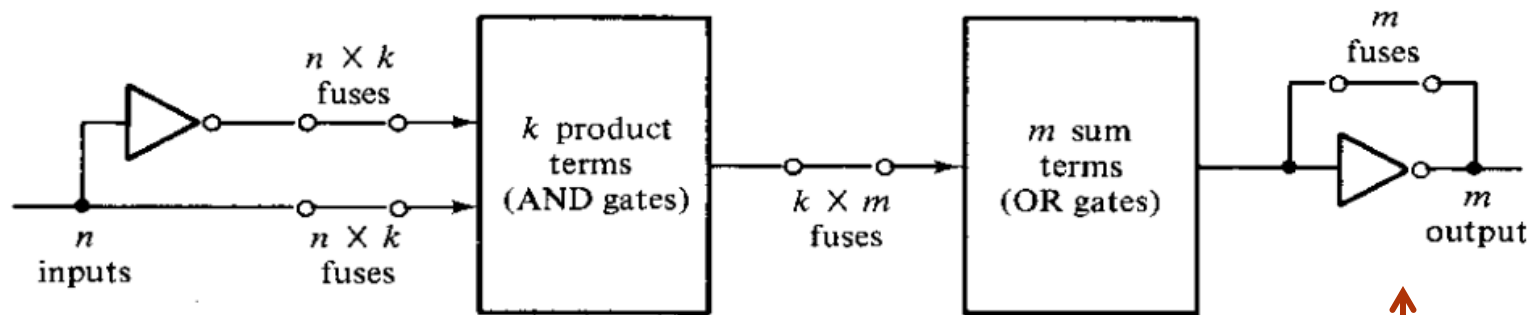
Programmable Logic Array (PLA)

Why PLA?

- A combinational circuit may occasionally have don't care conditions. When implemented with a ROM, a don't care condition becomes an address input that will never occur. The words at the don't care addresses need not be programmed and may be left in their original state (all 0's or all 1's). The result is that not all the bit patterns available in the ROM are used, which may be considered as waste of available equipment.
- For example, a combinational circuit that converts a 12-bit card code to a 6-bit internal alphanumeric code.
 - * It consists 12 inputs and 6 outputs. The size of the ROM must be 4096×6 ($2^{12} \times 6$).
 - * There are only 47 valid entries for the card code, all other input combinations are don't care. The remaining 4049 words of ROM are not used and are thus wasted.
- So, **for cases where the number of don't care conditions is excessive**, it is more economical to use a second type of LSI component called Programmable Logic Array (PLA).

Programmable Logic Array (PLA)

- PLA does not provide full decoding of the variables and does not generate all the minterms as in the ROM.
- A block diagram is shown in fig. It consists n inputs, m -outputs, k product terms and m sum terms. The product terms constitute a group of k AND gates and the sum terms constitute a group of m OR gates.



Links between all n inputs and their complement values to each of the AND gates.

$2n \times k$ links

Links between outputs of the AND gates and the inputs of the OR gates.

$k \times m$ links

Links to generate AND-OR form or, AND-OR-INVERT form

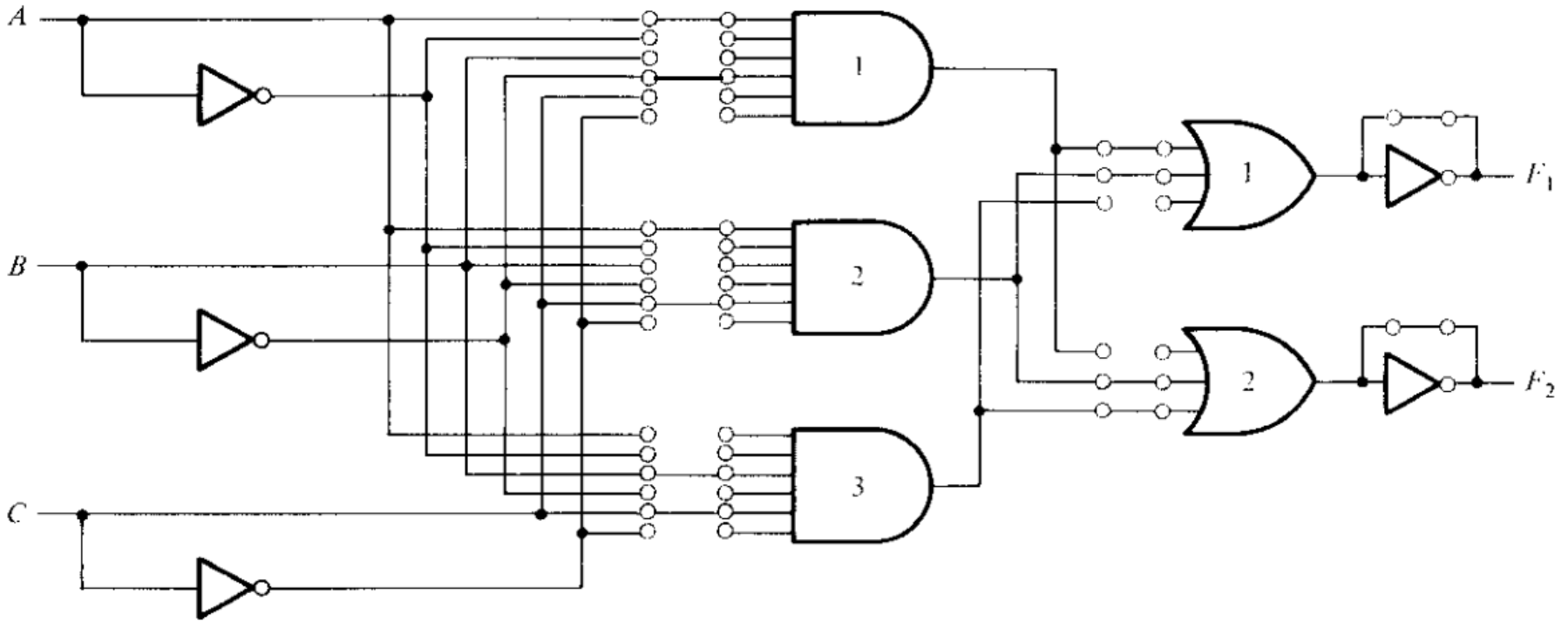
m links

- The number of programmed links is $2n \times k + k \times m + m$, whereas that of a ROM is $2^n \times m$

Implementation of combinational circuit by PLA.

$$F_1 = AB' + AC$$

$$F_2 = AC + BC$$



PLA Types.

- PLA may be mask-programmable or field programmable.
- With a **mask programmable PLA**, the customer must submit a PLA program table to the manufacturer. This table is used by the vendor to produce a custom-made PLA that has the required internal paths between inputs and outputs.
- A second type of PLA available is called **field programmable logic array or FPLA**. The FPLA can be programmed by the user by means of certain recommended procedure. Commercial hardware programmable units are available for use in conjunction with certain FPLAs