DIGITAL SYSTEM DESIGN
III-Sem ECE

Ms. P.Binduswetha
Assistant Professor –ECE department
G Pulla Reddy Engineering College (Autonomous)
Mail: binduswetha.ece@gmail.com

Module-3 COMBINATIONAL LOGIC DESIGN **contents**

- Combinational circuits
- Half & Full adder
- Binary adder/ Subtractor
- CLA & BCD Adder
- Half & Full subtractor
- Binary multipliers
- Magnitude comparator
- Decoder and its applications for logic implementation
- Encoder & Priority encoder
- Multiplexor & De-multiplexer
- Logic diagrams using MUX
- Hazards
- Unit-2 PLD’s

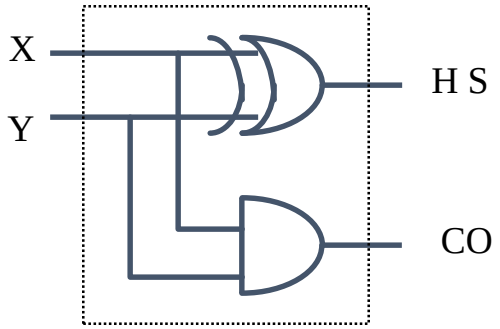
Combinational Logic Devices

- Combinational Logic Circuits are memoryless digital logic circuits whose output at any instant in time depends only on the combination of its inputs. i.e., the output is dependant at all times on the combination of its inputs. It is *memoryless*.
- Combinational circuit is a circuit in which it combine the different gates in the circuit, for example encoder, decoder, mux and demux. Some of the characteristics of combinational circuits are following −
- The output of combinational circuit at any instant of time, depends only on the levels present at input terminals.
- The combinational circuit do not use any memory. The previous state of input does not have any effect on the present state of the circuit.
- A combinational circuit can have an n number of inputs and m number of outputs.

Half Adder: adds two 1-bit operands

- Truth table :

X	Y	HS=(X+Y)	CO
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



- $$HS = X \oplus Y$$

$$CO = X \bullet Y$$

Full Adders: provide for carries between bit positions

- Basic building block is “full adder”
  - 1-bit-wide adder, produces sum and carry outputs
- Truth table:

Full Adders: provide for carries between bit positions

- Basic building block is “full adder”
  - 1-bit-wide adder, produces sum and carry outputs
- Truth table:

X	Y	Cin	S	Cout
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

Full Adders: provide for carries between bit positions

- Basic building block is “full adder”
  - 1-bit-wide adder, produces sum and carry outputs
- Truth table:

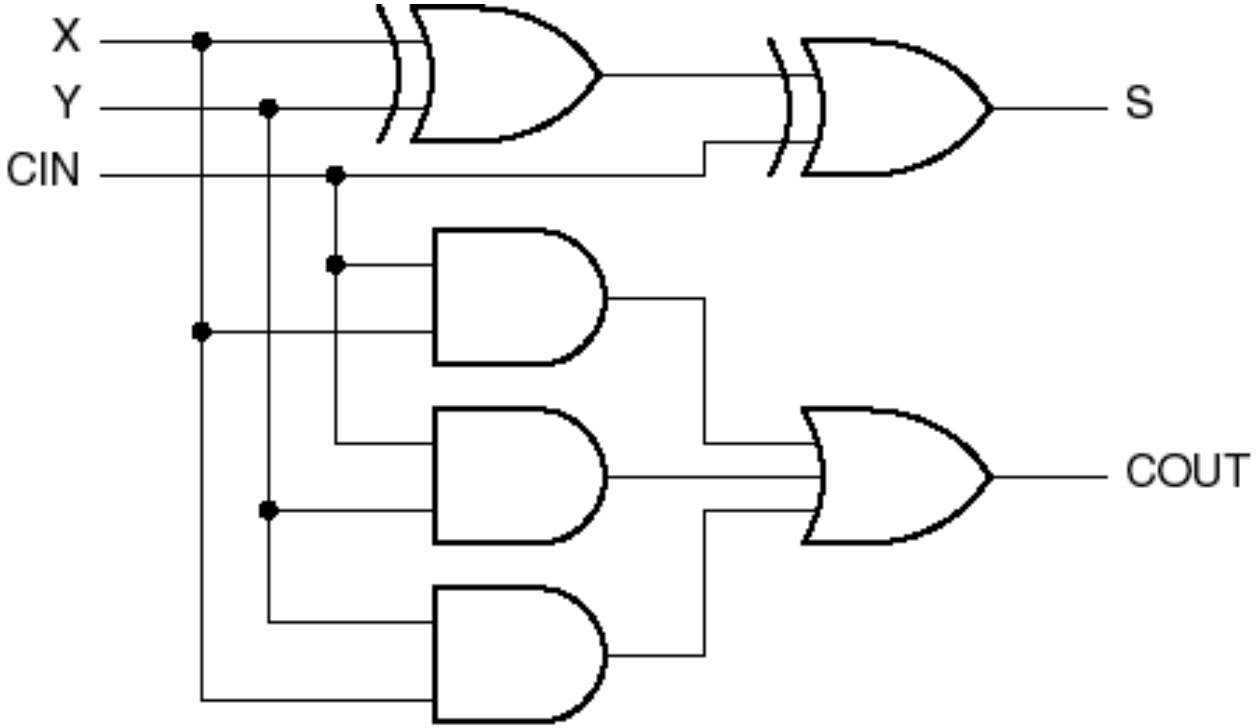
X	Y	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

S is 1 if an odd number of inputs are 1.

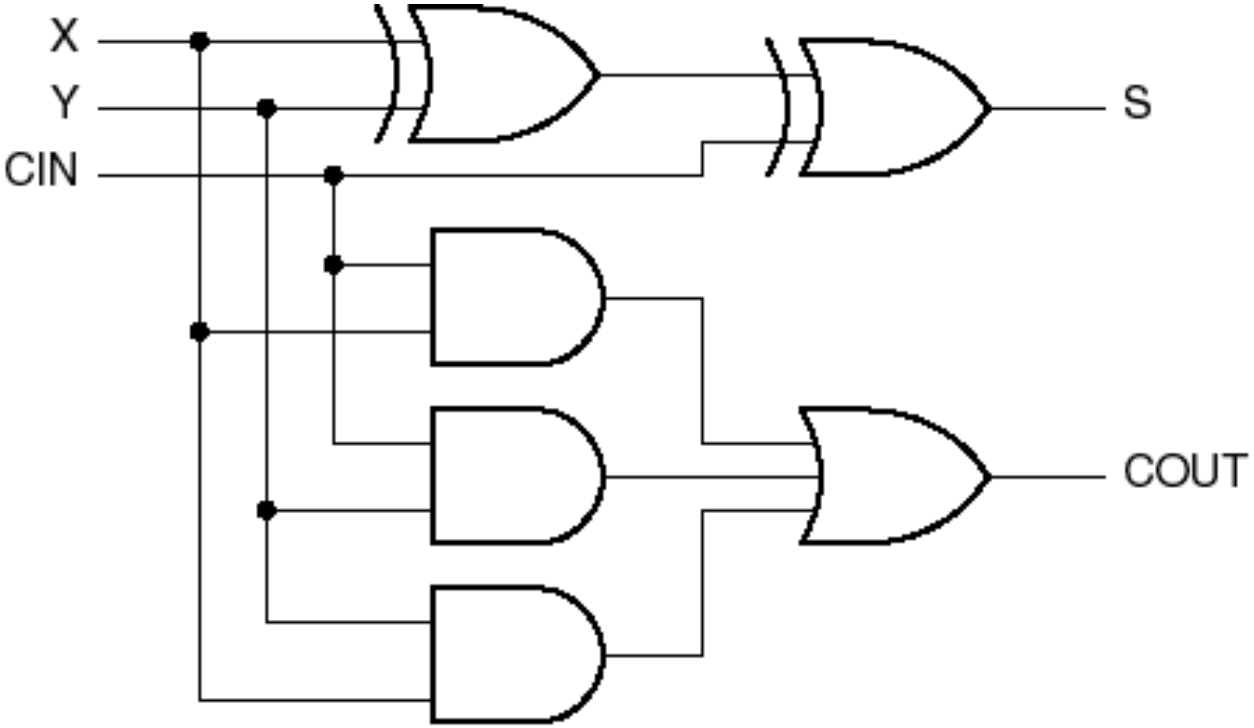
COUT is 1 if two or more of the inputs are 1.

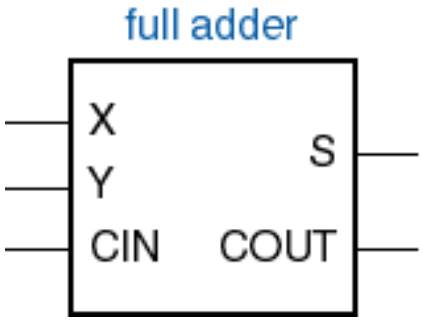
Recall: Table 2-3, pp32

Full-adder circuit

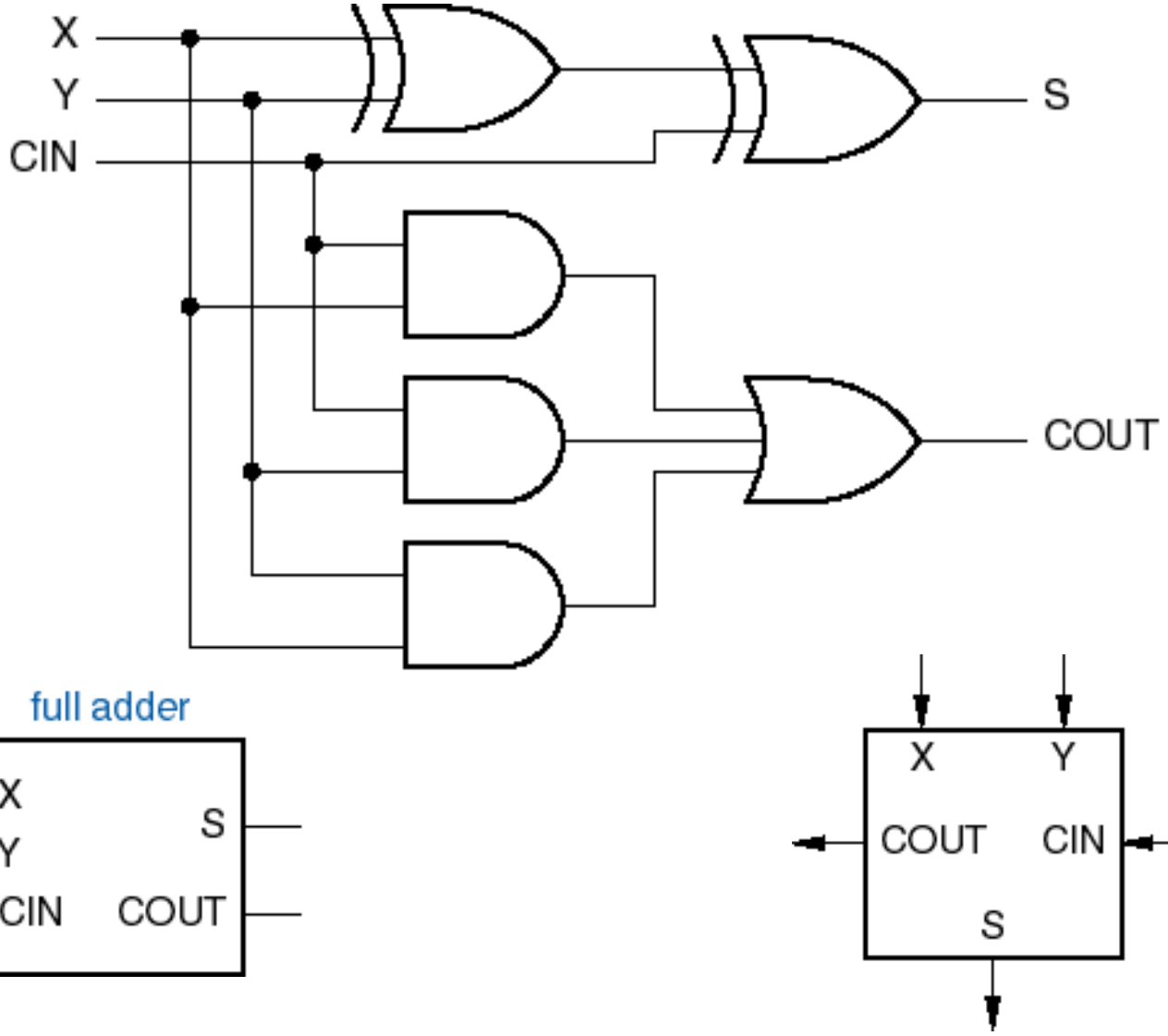


Full-adder circuit





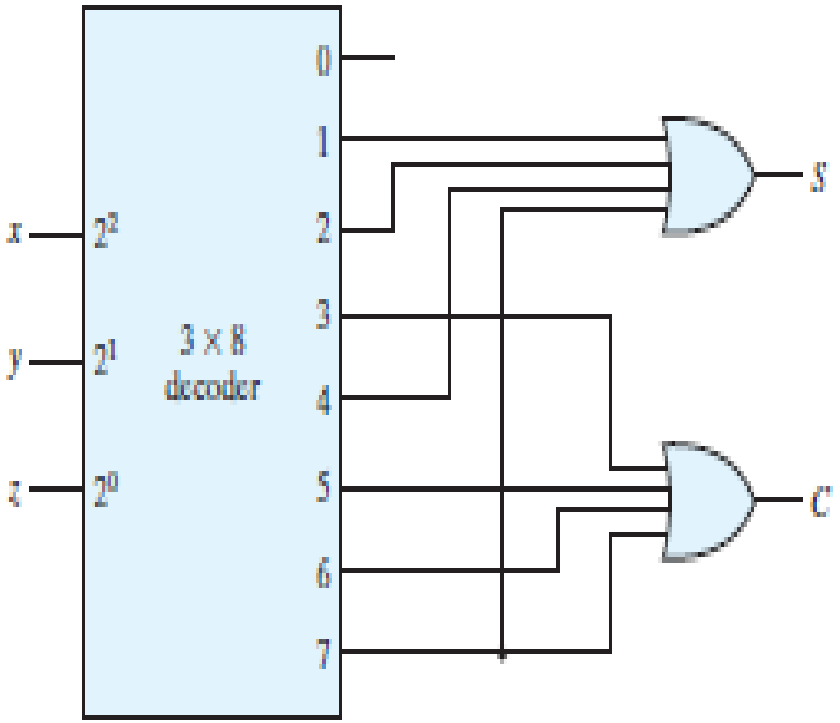
Full-adder circuit



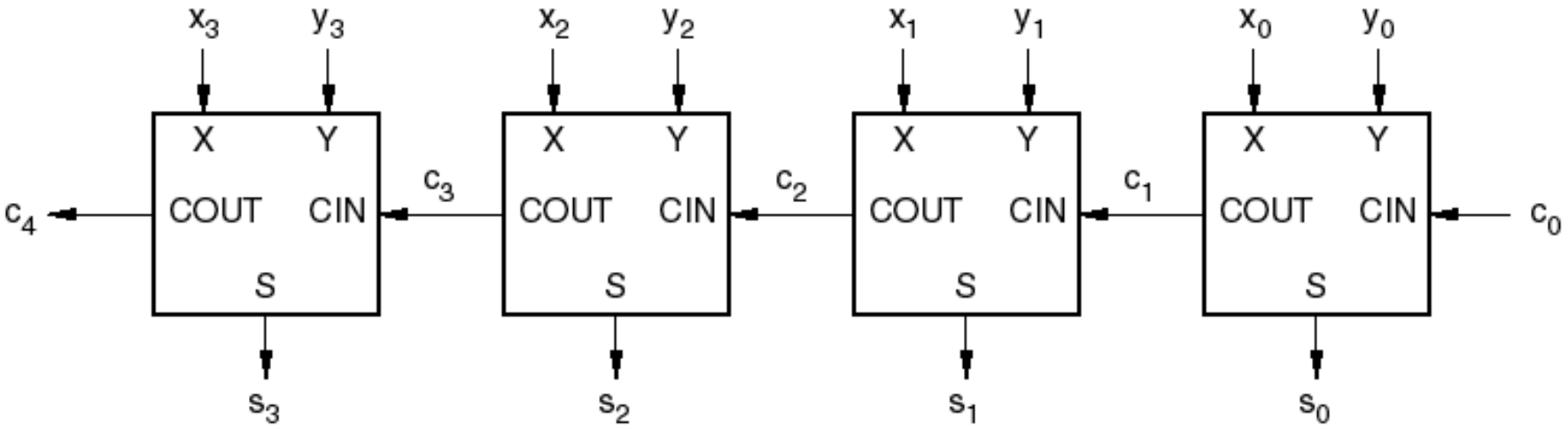
Combinational logic implementation
Realize Full Adder using Decoder concept.

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

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



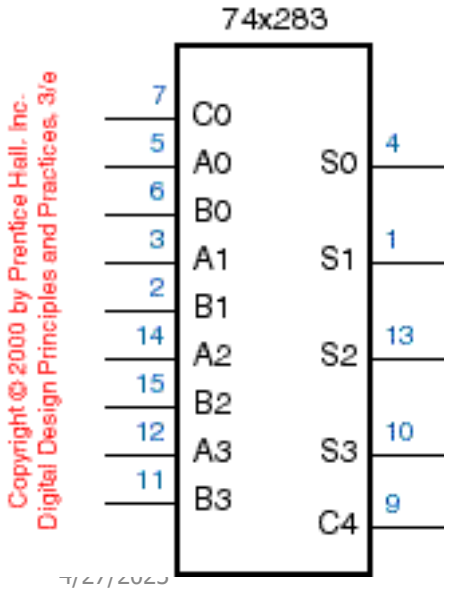
Ripple adder

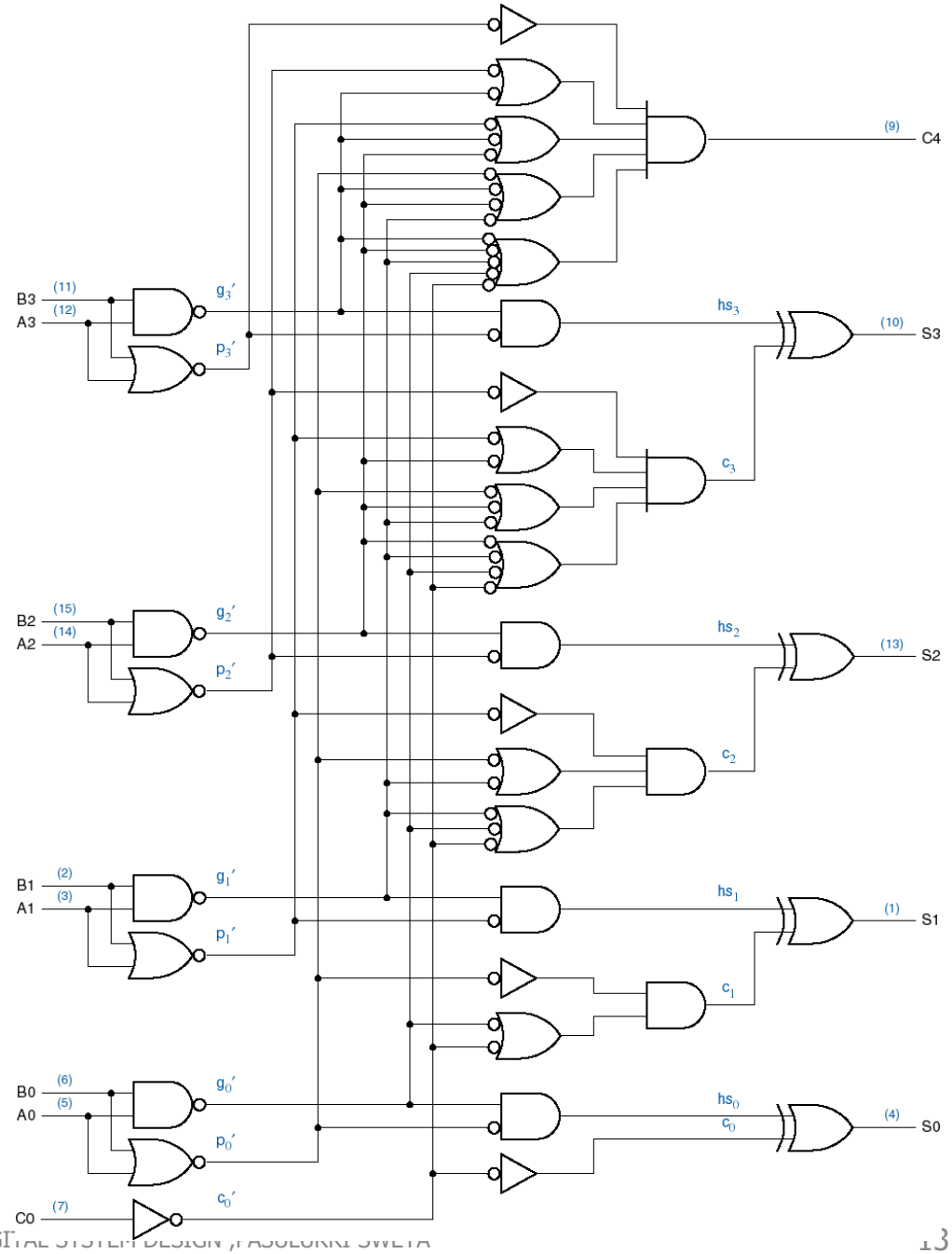


- Speed limited by carry chain
- Faster adders eliminate or limit carry chain
  - 2-level AND-OR logic ==> 2^n product terms
  - 3 or 4 levels of logic, carry look-ahead

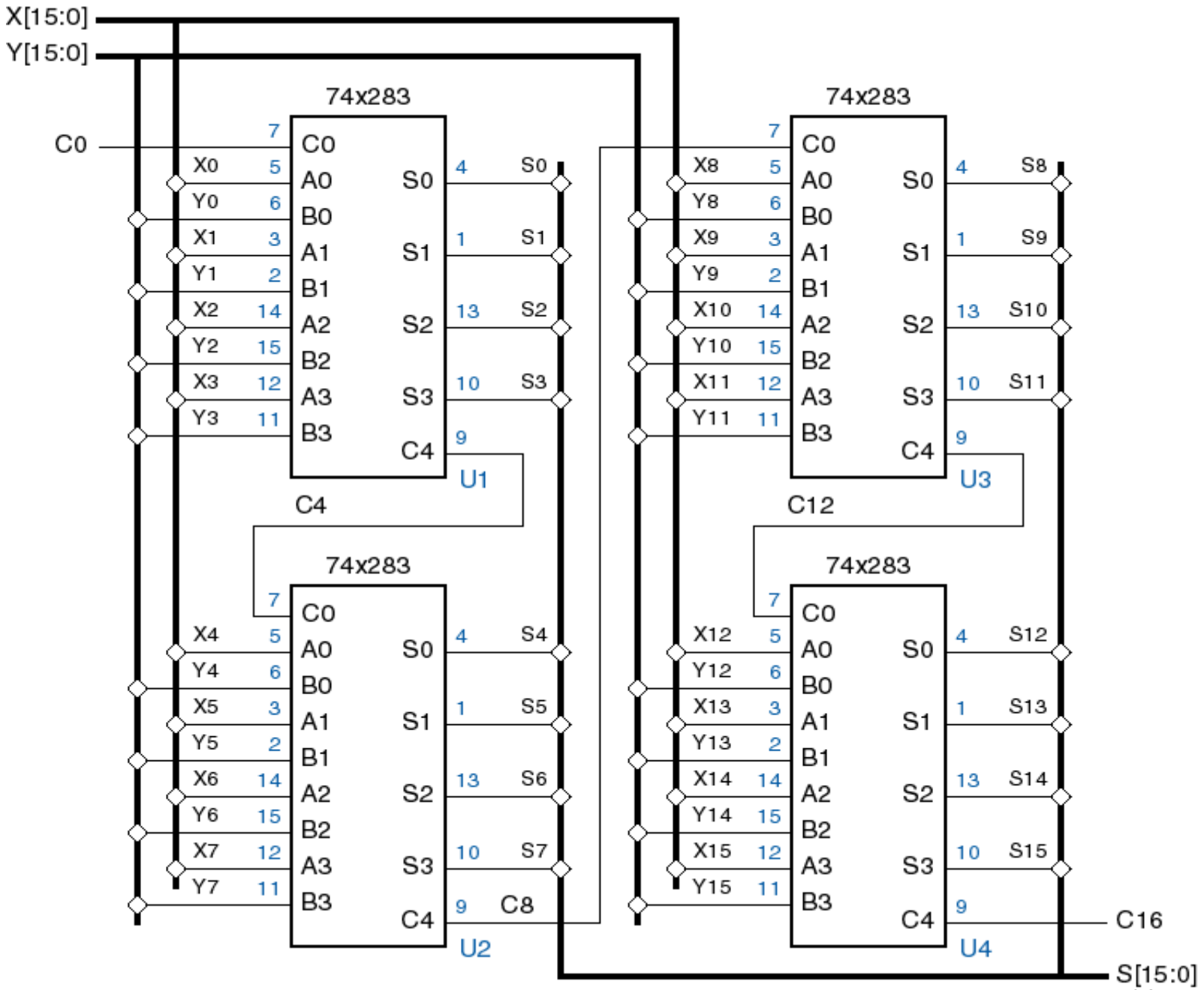
74x283
4-bit adder

- Uses carry look-ahead internally





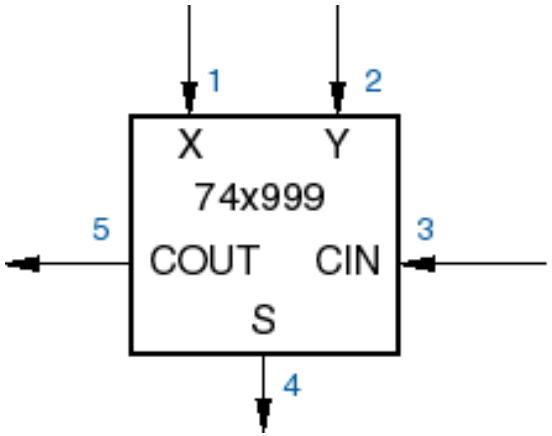
16-bit group ripple adder



Subtraction

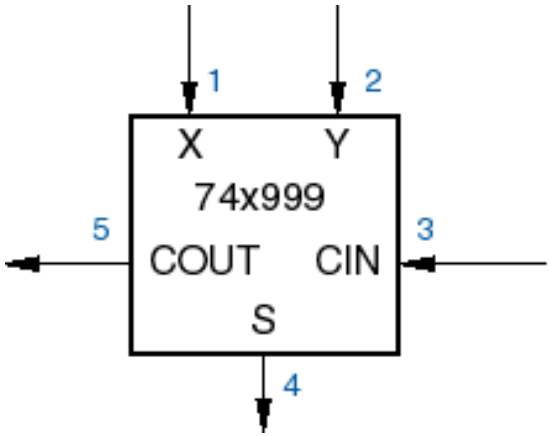
- Subtraction is the same as addition of the two's complement.
- The two's complement is the bit-by-bit complement plus 1.
- Therefore, $X - Y = X + \overline{Y} + 1$

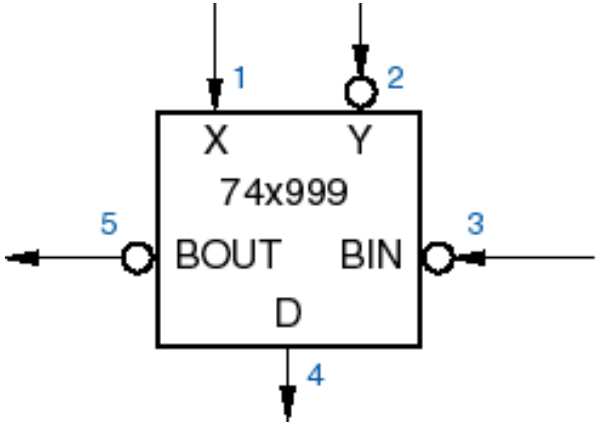
Full Subtractor = full adder, almost



- X,Y are n-bit unsigned binary numbers
- Addition : S = X + Y
- Subtraction : D = X - Y = X + (-Y) =
  = X+ (Two's Complement of Y)
  = X+ (One's Complement of Y) + 1
  = X+ Y'+ 1

Full Subtractor = full adder, almost





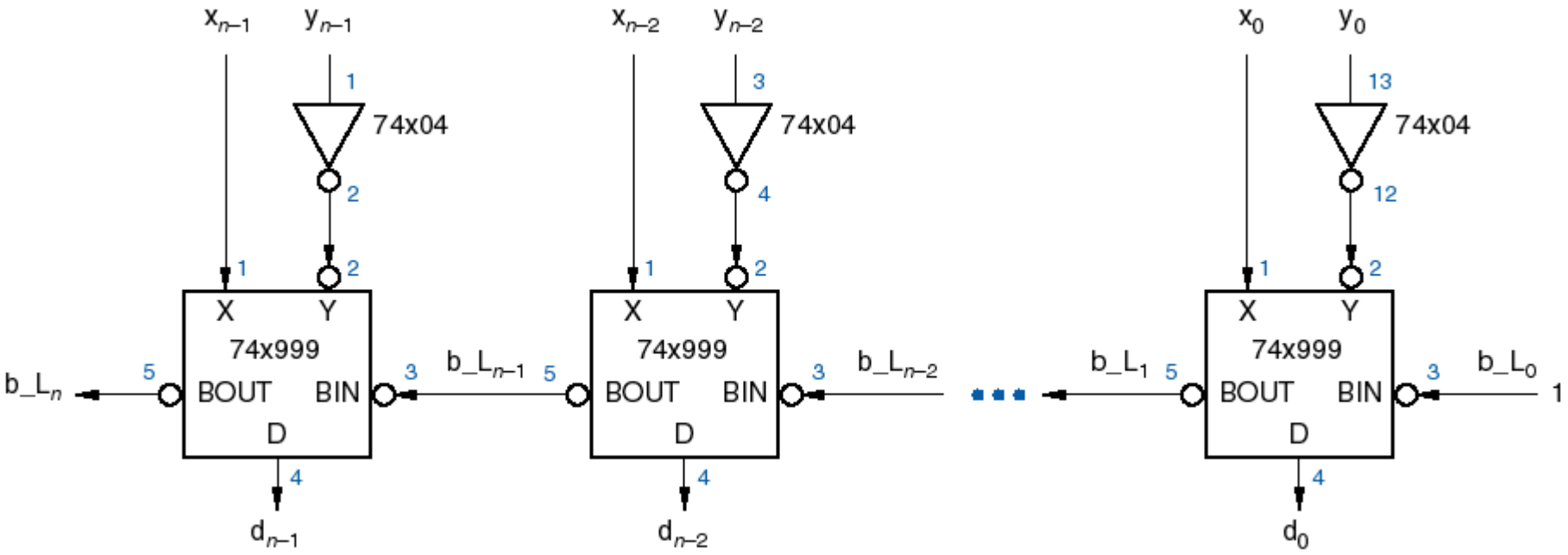
- X,Y are n-bit unsigned binary numbers
- Addition : S = X + Y
- Subtraction : D = X - Y = X + (-Y) =
  = X+ (Two's Complement of Y)
  = X+ (One's Complement of Y) + 1
  = X+ Y'+ 1

Using Adder as a Subtractor

- Ripple Adder can be used as a Subtractor by inverting Y and setting the initial carry (CIN) to 1

Using Adder as a Subtractor

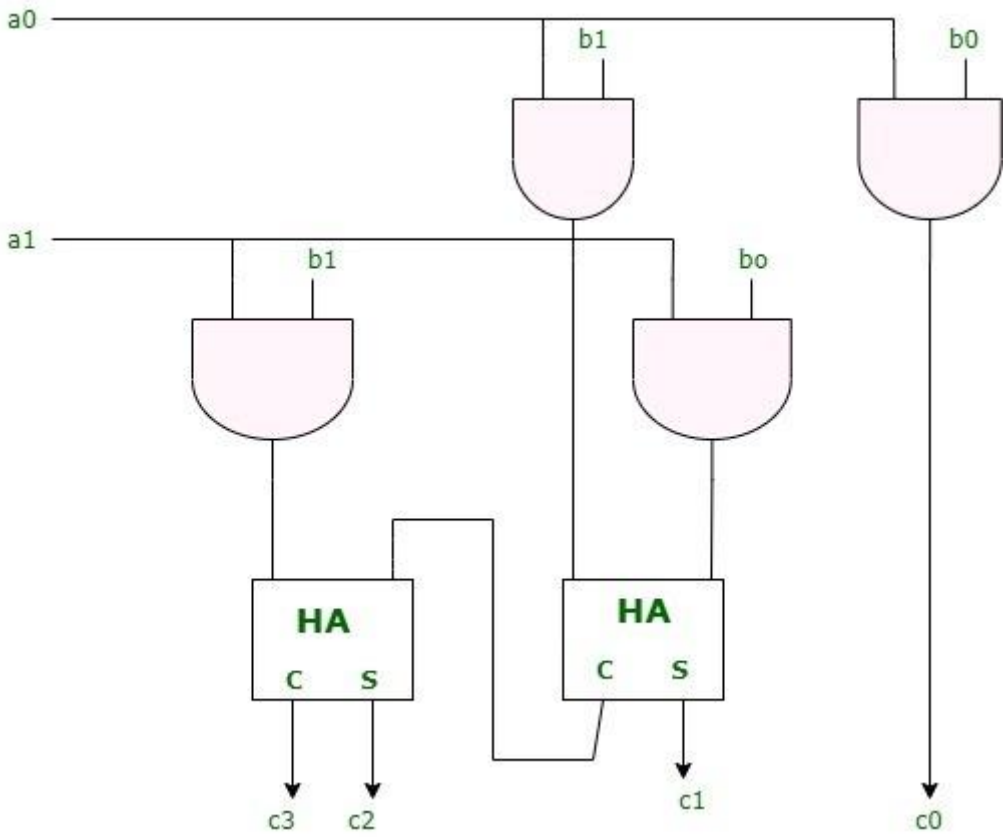
- Ripple Adder can be used as a Subtractor by inverting Y and setting the initial carry (CIN) to 1

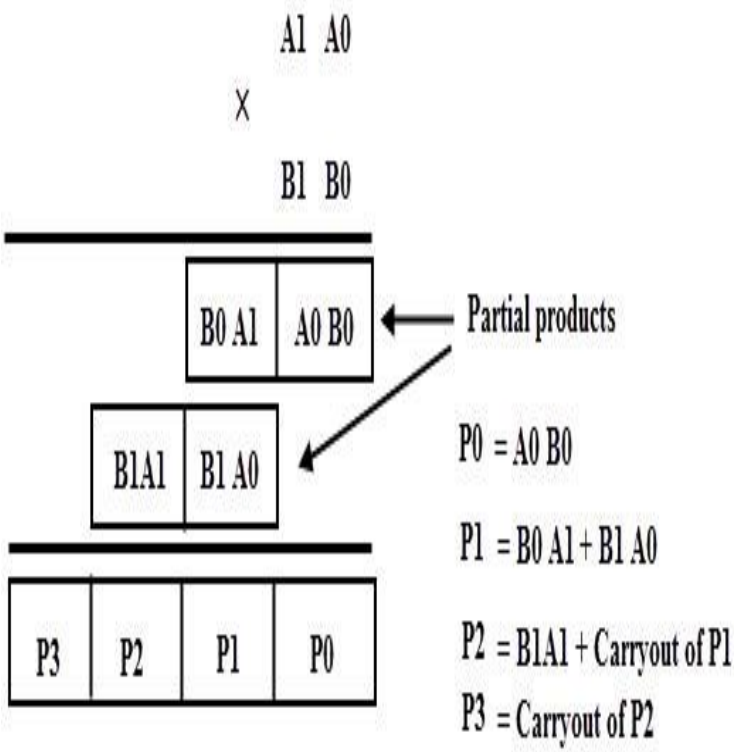


Binary Multipliers

- A Binary Multiplier is **a digital circuit used in digital electronics to multiply two binary numbers and provide the result as output**.

2:2 Multipliers

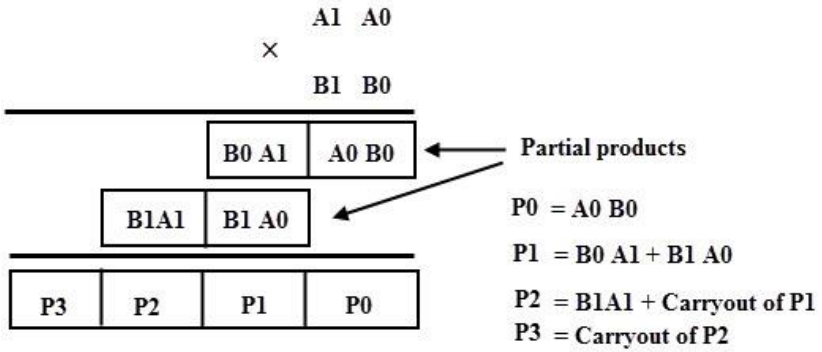




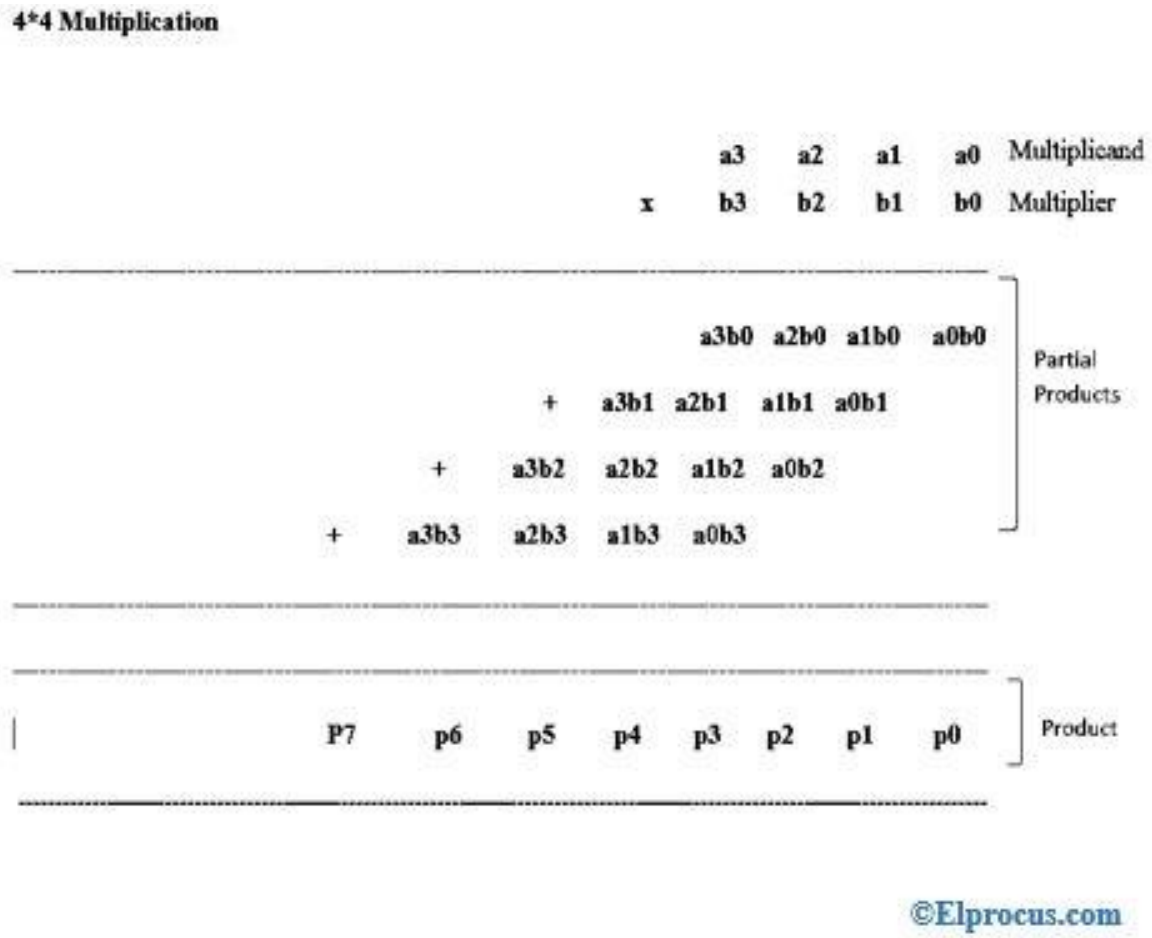
2: 2 multipliers

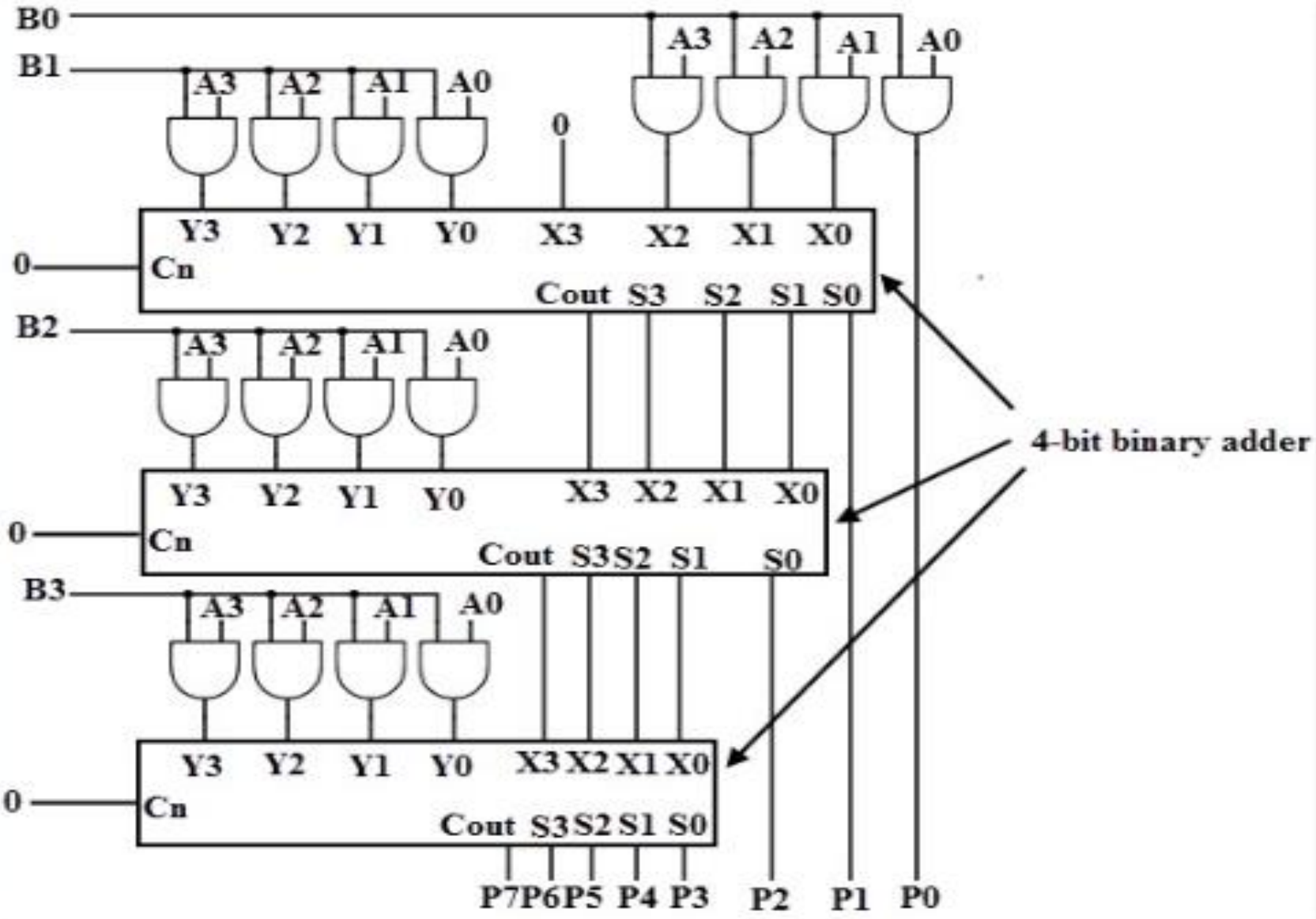
Table 1: Truth table for binary multiplier (inputs A_1A_0, B_1B_0, and outputs $P_3P_2P_1P_0$)

A_1	A_0	B_1	B_0	P_3	P_2	P_1	P_0	Output value in decimal
0	1	1	1	0	0	1	1	3
1	0	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0	2
1	0	1	0	0	1	0	0	6
1	0	1	1	0	1	1	0	6
1	1	0	0	0	0	0	0	0
1	1	0	1	0	0	1	1	3
1	1	1	0	0	1	1	0	6
1	1	1	1	1	0	0	1	9

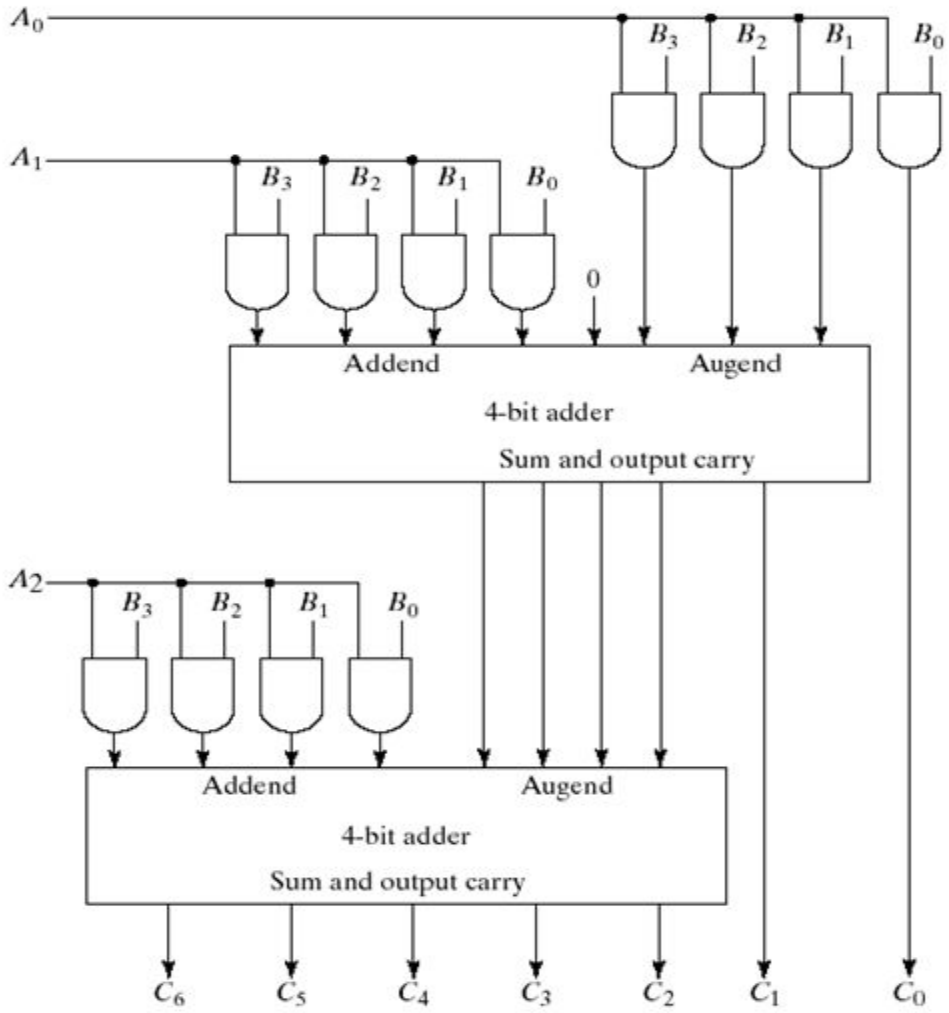


4*4 Multipliers




B0
B1
B2
B3
A3 A2 A1 A0
0
Y3 Y2 Y1 Y0
X3 X2 X1 X0
Cn
Cout S3 S2 S1 S0
4-bit binary adder
P7 P6 P5 P4 P3 P2 P1 P0

Example: 4-bit by 3-bit Multiplier



$n = 4$

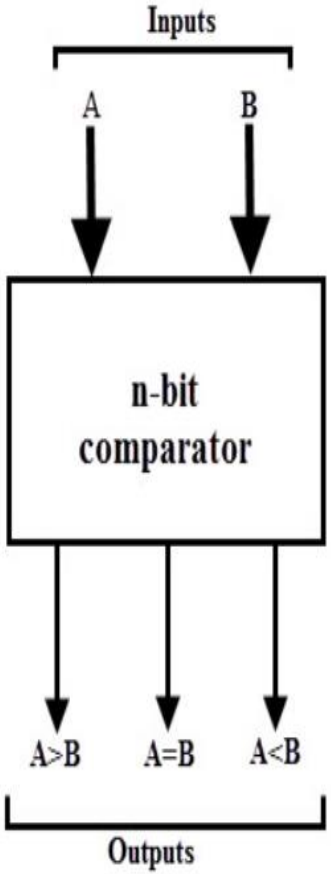
$m = 3$

4 x 3 AND gates

m-1 = 3 – 1 adders

COMPARATORS

Three binary variables are used to indicate the outcome of the comparison as A>B, A<B, or A=B. The below figure shows the block diagram of a n-bit comparator which compares the two numbers of n-bit length and generates their relation between themselves.

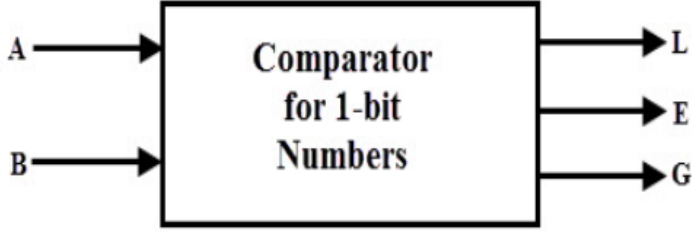


Magnitude comparators

- Magnitude Comparator are digital circuits which have two ports which accept and have three single bit outputs. It is used **to comparing individual bits**, multi-bit comparators can be constructed to compare whole BCD words to produce an output if one word is larger, equal to or less than the other.

Comparing two binary words for equality is a commonly used operation in computer systems and device interfaces. A circuit that compares two binary words and indicates whether they are equal is called a *comparator*.

Some comparators interpret their input words as signed or unsigned numbers and also indicate an arithmetic relationship (greater or less than) between the words. These devices are often called *magnitude comparators*.



The truth table for the single bit comparator is given below. When A0 B0 = 00 & 11, both inputs are equal, therefore A=B output will be high. When A0 B0 = 01, B is more than A and hence AB is active.

A_0	B_0	L	E	G
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

From the truth table logical expressions for each output can be expressed as

$$A0 < B0: L = \overline{A0}\, B0$$

$$A0 = B0: E = \overline{A0}\,\overline{B0} + A0\, B0$$

$$A0 > B0: G = A0\,\overline{B0}$$

1-bit comparator

A	B	A<B	A=B	A>B
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

1- Bit Comparator Logic diagram

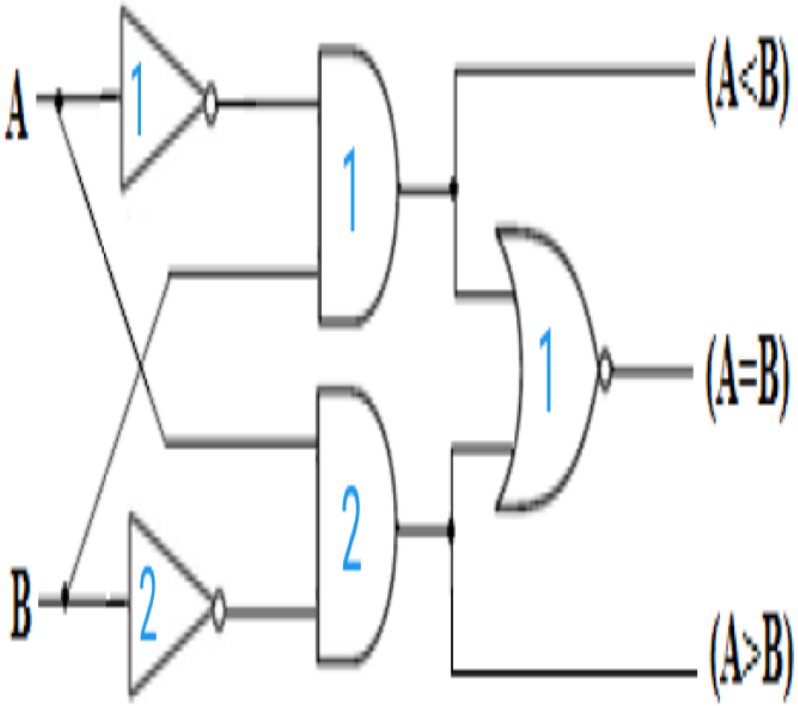
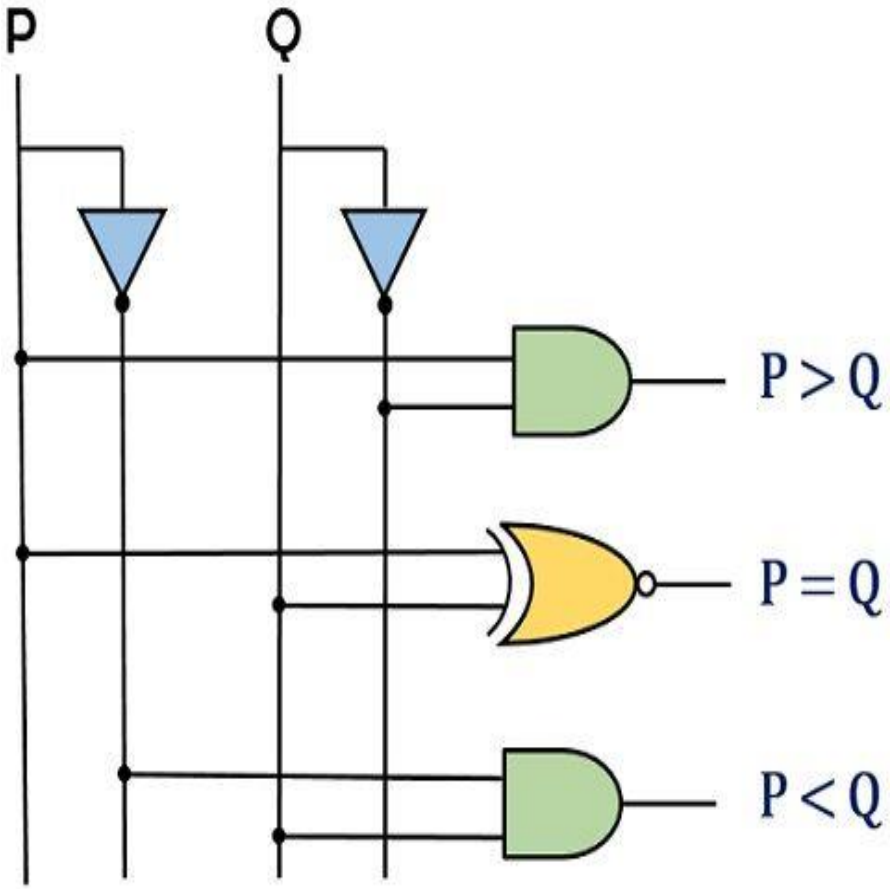


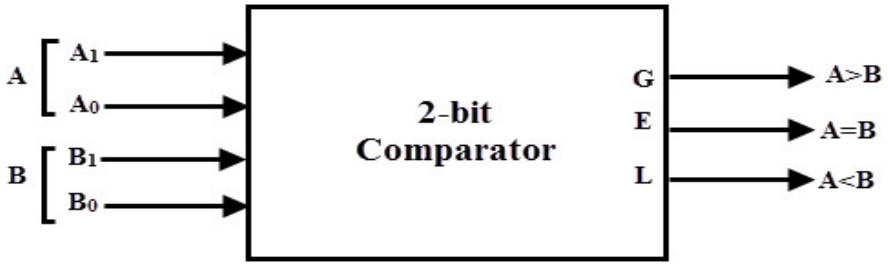
Fig. 1: One-bit Comparator



Logic Circuit of 1-bit Magnitude Comparator

Electronics Coach

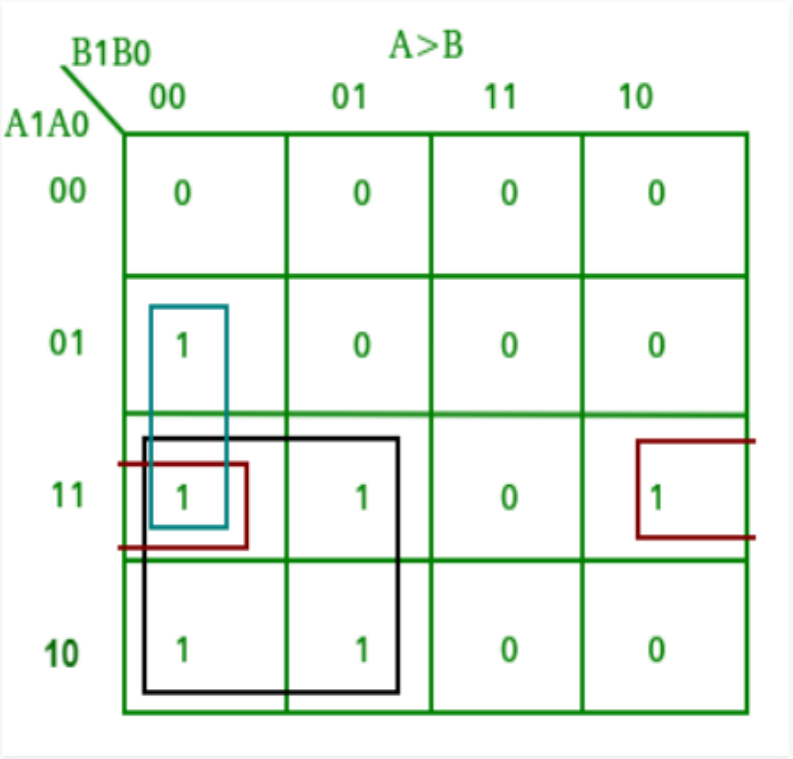
2-BIT COMPARATORS

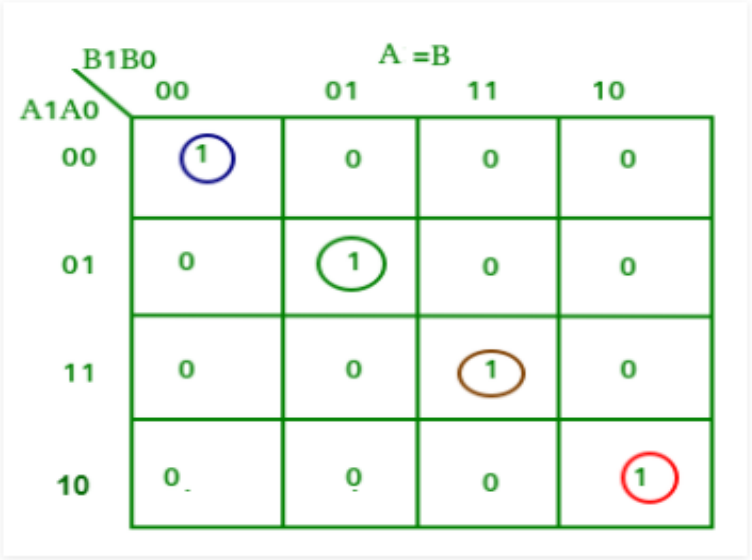


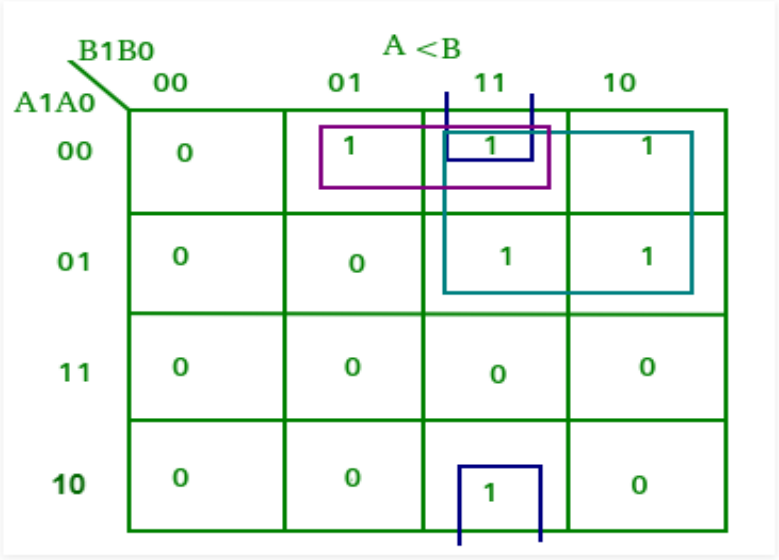
The truth table of this comparator is shown below which depicting various input and output states.

Inputs				Outputs		
A_1	A_0	B_1	B_0	A>B	A=B	A<B
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

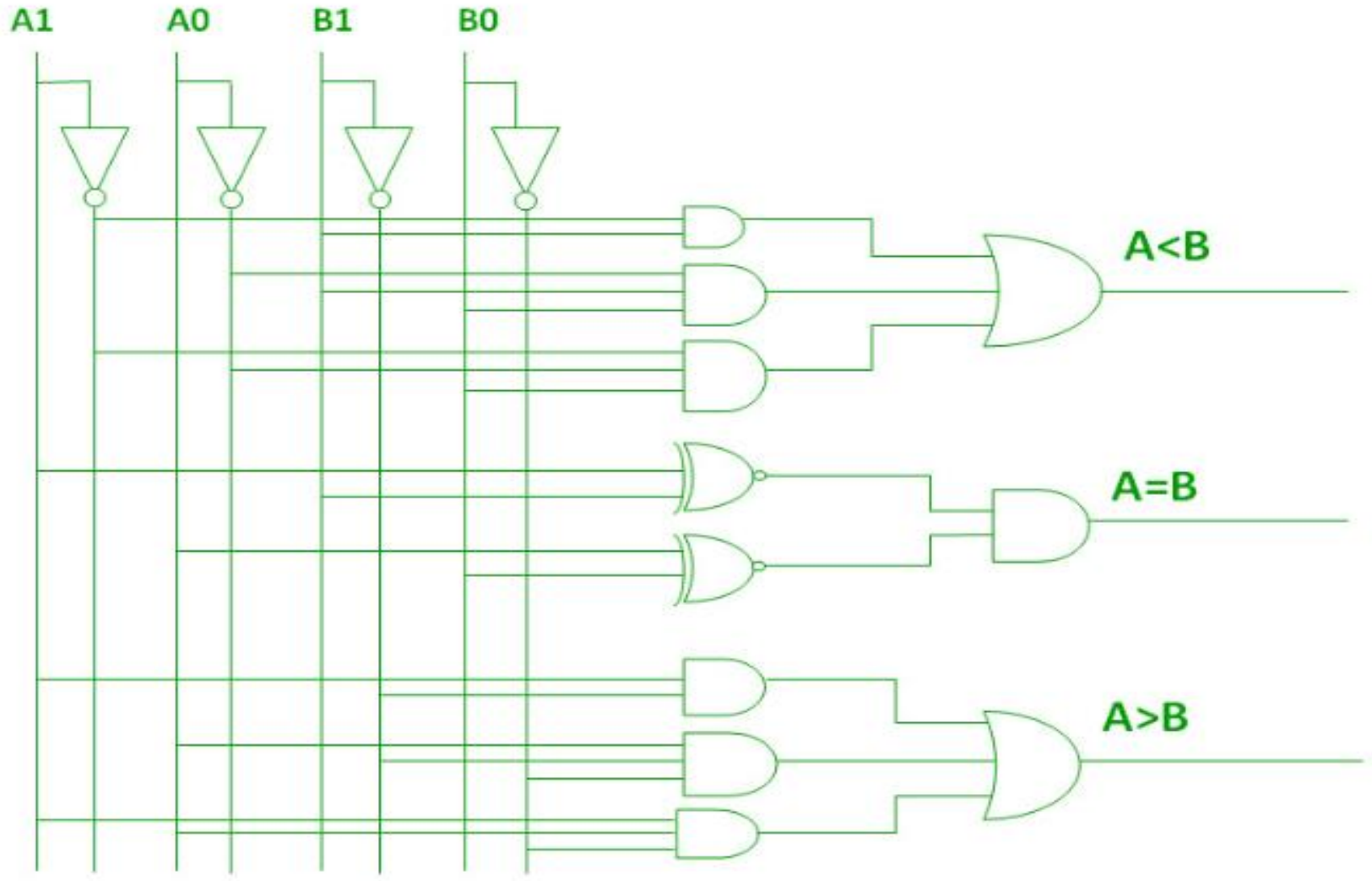
2-bit comparator







2-bit Comparator Circuit diagram



4-bit comparator

The condition of A>B can be possible in the following four cases:

1. If A3 = 1 and B3 = 0
2. If A3 = B3 and A2 = 1 and B2 = 0
3. If A3 = B3, A2 = B2 and A1 = 1 and B1 = 0
4. If A3 = B3, A2 = B2, A1 = B1 and A0 = 1 and B0 = 0

Similarly the condition for A<B can be possible in the following four cases:

1. If A3 = 0 and B3 = 1
1. If A3 = B3 and A2 = 0 and B2 = 1
2. If A3 = B3, A2 = B2 and A1 = 0 and B1 = 1
3. If A3 = B3, A2 = B2, A1 = B1 and A0 = 0 and B0 = 1

The condition A = B,
The condition of A=B is possible only when all the individual bits of one number exactly coincide with corresponding bits of another number.

A=B: (A3 Ex-Nor B3) (A2 Ex-Nor 82) (Al Ex-Nor BI) (AO Ex-Nor BO)

4-BIT COMPARISION PROCESS BIT BY BIT

Numerical:

The Comparator compare the number bit by bit from MSB to LSB.
The function table of 4-bit comparator is:

Comparing Input				Output		
A_3, B_3	A_2, B_2	A_1, B_1	A_0, B_0	A > B	A = B	A < B
$A_3 > B_3$	X	X	X	1	0	0
$A_3 < B_3$	X	X	X	0	0	1
$A_3 = B_3$	$A_2 > B_2$	X	X	1	0	0
$A_3 = B_3$	$A_2 < B_2$	X	X	0	0	1
$A_3 = B_3$	$A_2 = B_2$	$A_1 > B_1$	X	1	0	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 < B_1$	X	0	0	1
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 > B_0$	1	0	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 < B_0$	0	0	1
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	0	1	0

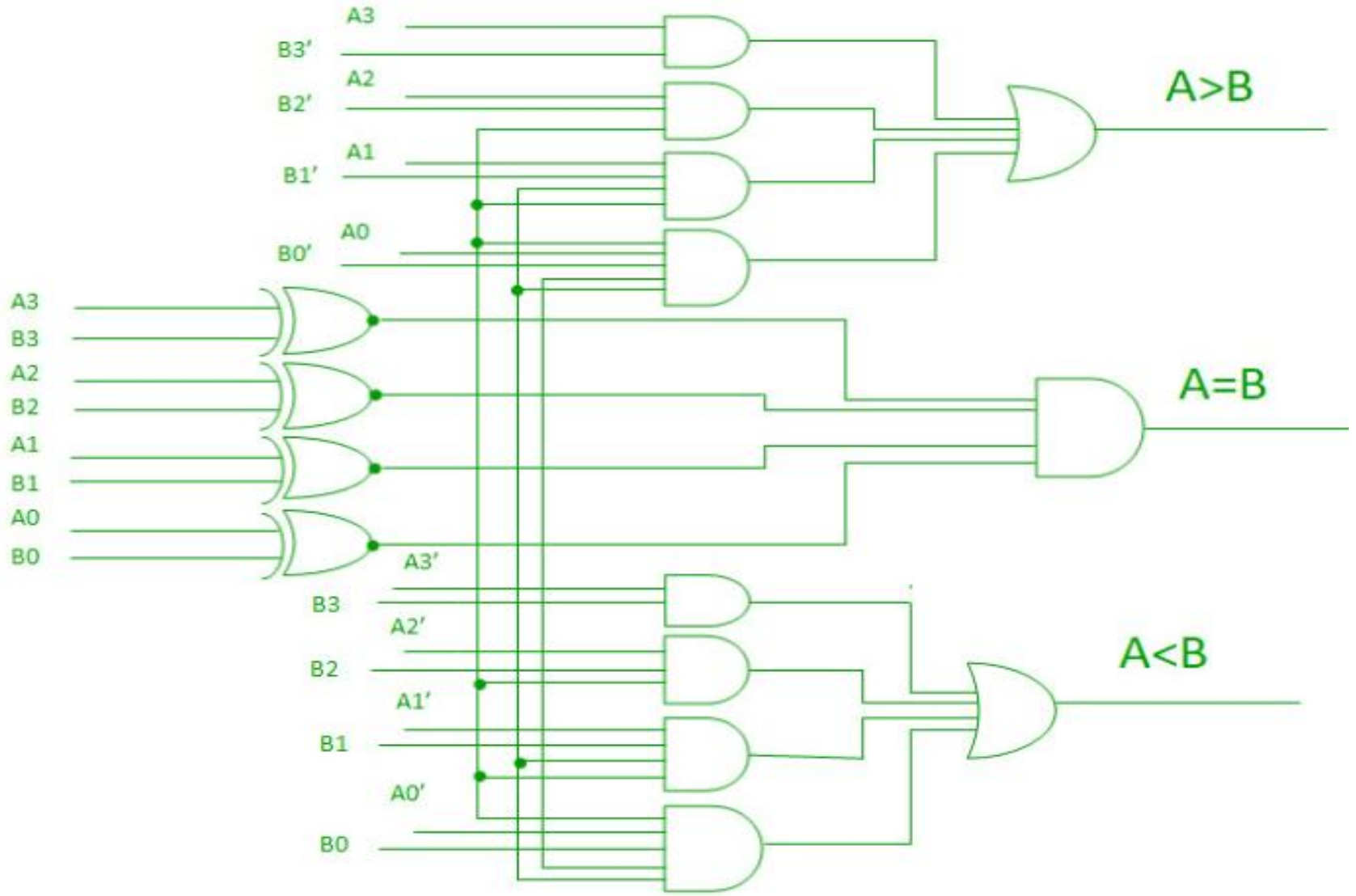
4-BIT COMP

Function Table

Comparing Inputs				Cascading Inputs			Outputs		
A3, B3	A2, B2	A1, B1	A0, B0	A > B	A < B	A = B	A > B	A < B	A = B
A3 > B3	X	X	X	X	X	X	H	L	L
A3 < B3	X	X	X	X	X	X	L	H	L
A3 = B3	A2 > B2	X	X	X	X	X	H	L	L
A3 = B3	A2 < B2	X	X	X	X	X	L	H	L
A3 = B3	A2 = B2	A1 > B1	X	X	X	X	H	L	L
A3 = B3	A2 = B2	A1 < B1	X	X	X	X	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 > B0	X	X	X	H	L	L
A3 = B3	A2 = B2	A1 = B1	A0 < B0	X	X	X	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	H	L	L	H	L	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	H	L	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	L	H	L	L	H
A3 = B3	A2 = B2	A1 = B1	A0 = B0	X	X	H	L	L	H
A3 = B3	A2 = B2	A1 = B1	A0 = B0	H	H	L	L	L	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	L	L	H	H	L

H = High Level, L = Low Level, X = Don't Care

4-bit Comparator

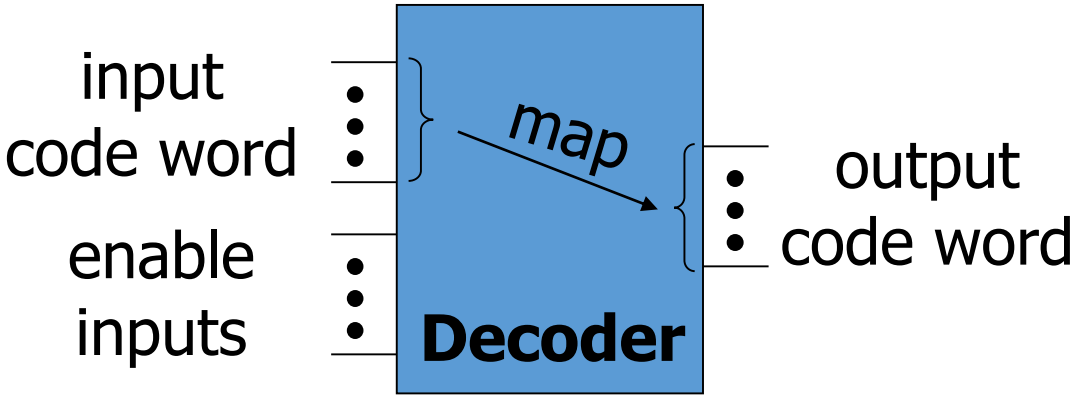


N- bit comparator

- For n- the bit comparator then, the number of combinations for which
- A = B is 2^n
- A > B or A < B is $(2^{2n} – 2^n)/2$

Decoders

- A decoder is a multiple-input, multiple-output logic circuit that converts coded inputs into coded outputs, where the input and output codes are different. The input code generally has fewer bits than the output code, and there is one-to-one mapping from input code words into output code words.



Return Next

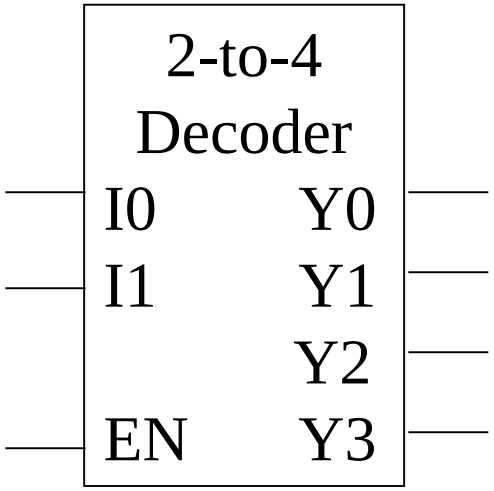
Decoders

- Binary decoder
  - The most common decoder circuit is an n-to-2^n decoder or binary decoder. Such a decoder has an n-bit binary input code and a 1-out-of-2^n output code.

Truth table for a 2-to-4 binary decoder

Inputs			Outputs			
EN	I1	I0	Y3	Y2	Y1	Y0
0	X	X	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

"don't-care" notation

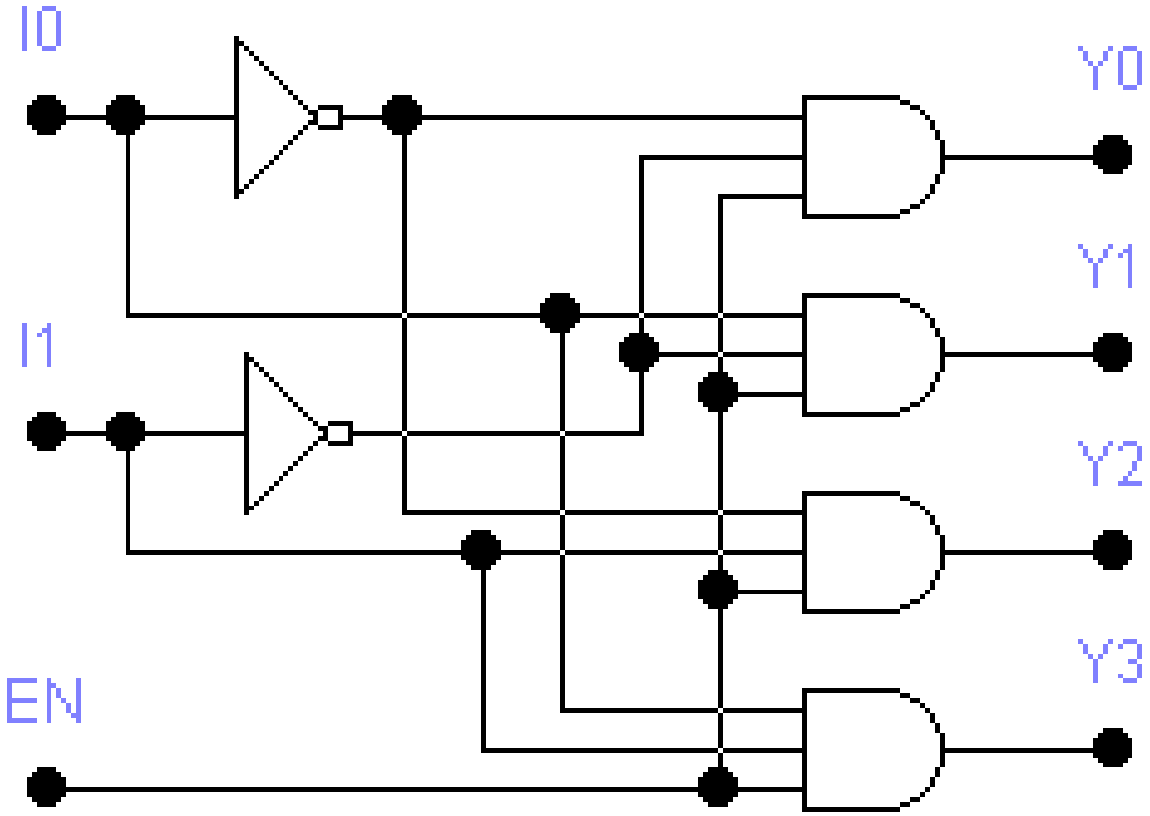


Return Back Next

Decoders

$$Y0 = \overline{I1} \cdot \overline{I0} \cdot EN \qquad Y1 = \overline{I1} \cdot I0 \cdot EN$$

$$Y2 = I1 \cdot \overline{I0} \cdot EN \qquad Y3 = I1 \cdot I0 \cdot EN$$



Simulation

Return Back Next

Decoders

- It is not necessary to use all of the outputs of a decoder, or even to decode all possible input combinations, e.g. a decimal or BCD decoder.

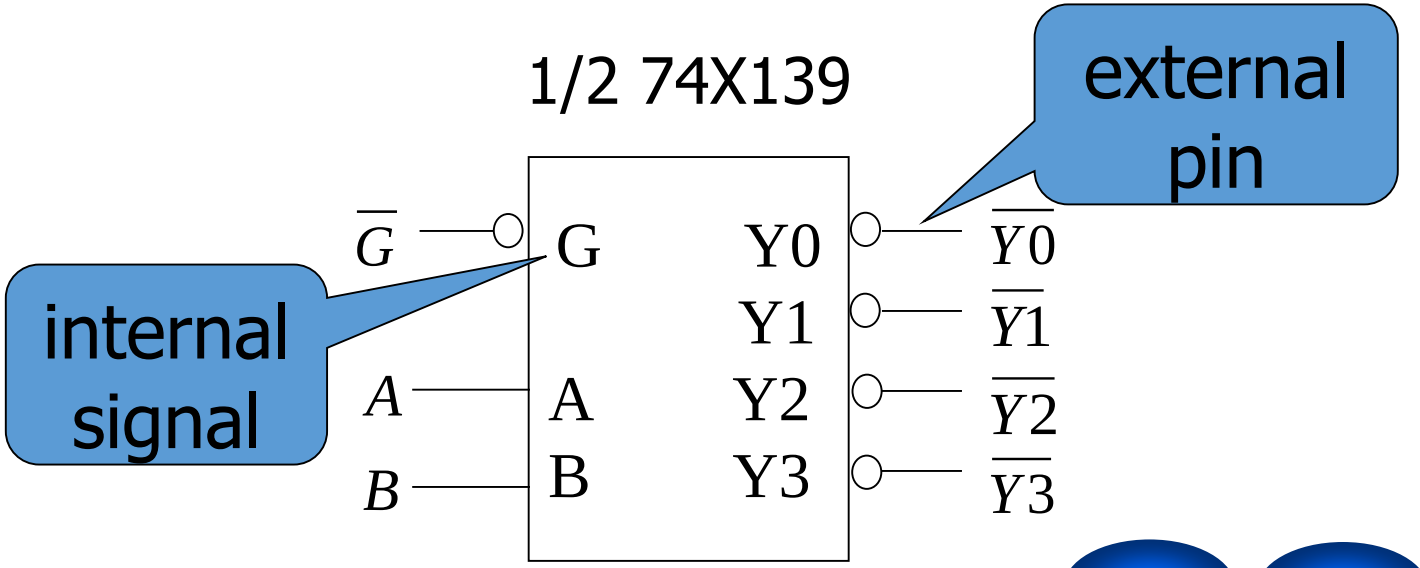
Logic Symbols for Lager-Scale Elements

- The most basic rule is that logic symbols are drawn with inputs on the left and outputs on the right. The top and bottom edges of a logic symbol are not normally used for signal connections. However, explicit power and ground connections are sometimes shown at the top and bottom, especially if these connections are made on "nonstandard" pins.

Return Back Next

Decoders

- We use an inversion bubble to indicate an active-low pin and the absence of a bubble to indicate an active-high pin.
- Active-high pins are given the same name as the internal signal, while active-low pins have the internal signal name with an overbar.



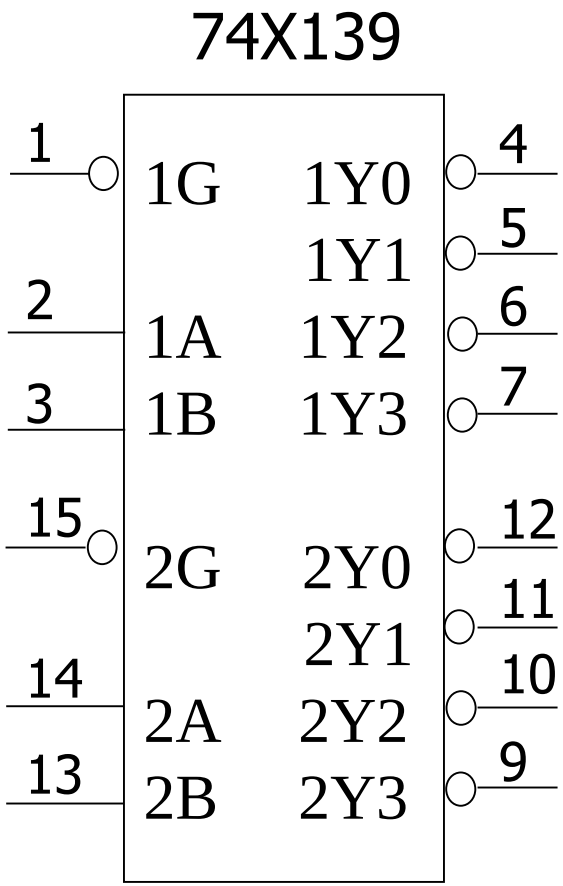
Return Back Next

Decoders

- The 74x139 Dual 2-to-4 Decoder

Truth table for one-half of a 74x139 dual 2-to-4 decoder

Inputs			Outputs			
$\overline{G}$	B	A	$\overline{Y3}$	$\overline{Y2}$	$\overline{Y1}$	$\overline{Y0}$
1	X	X	1	1	1	1
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1



(Logic diagram see P355 Figure 5-35)

Return Back Next

Decoders

- The 74x138 3-to-8 Decoder

Inputs						Outputs							
G1	$\overline{G2A}$	$\overline{G2B}$	C	B	A	$\overline{Y7}$	$\overline{Y6}$	$\overline{Y5}$	$\overline{Y4}$	$\overline{Y3}$	$\overline{Y2}$	$\overline{Y1}$	$\overline{Y0}$
0	X	X	X	X	X	1	1	1	1	1	1	1	1
X	1	X	X	X		1	1	1	1	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1	1	1	1
1	0	0	0	0	1	1	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	1	1	1	1	0	1
1	0	0	0	1	1	1	1	1	1	1	0	1	1
1	0	0	1	0	0	1	1	1	1	0	1	1	1
1	0	0	1	0	1	1	1	1	0	1	1	1	1
1	0	0	1	1	0	1	1	0	1	1	1	1	1
1	0	0	1	1	1	1	0	1	1	1	1	1	1
						0	1	1	1	1	1	1	1

(Logic diagram see P358 Figure 5-37)

Return Back Next

Decoders

- It has three enable inputs, G1, $\overline{G2A}$, $\overline{G2B}$, all of which must be asserted for the selected output to be asserted.
- The equation for the internal output signal Y5:

$$Y5 = G1 \cdot G2A \cdot G2B \cdot C \cdot \overline{B} \cdot A$$

- The equation for the external output signal $\overline{Y5}$:

$$\overline{\overline{Y5}} = \overline{G1 \cdot G2A \cdot G2B \cdot C \cdot \overline{B} \cdot A}$$

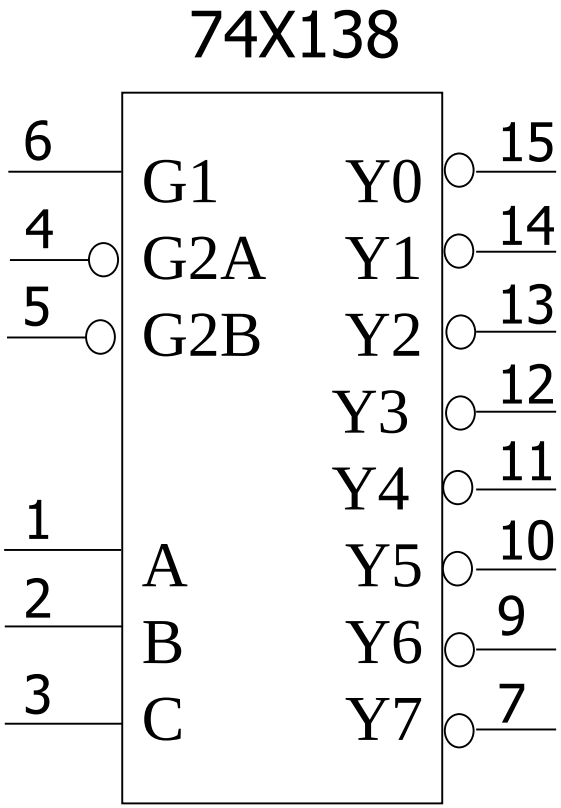
$$= \overline{G1} + \overline{\overline{G2A}} + \overline{\overline{G2B}} + \overline{C} + B + \overline{A}$$

Active-low

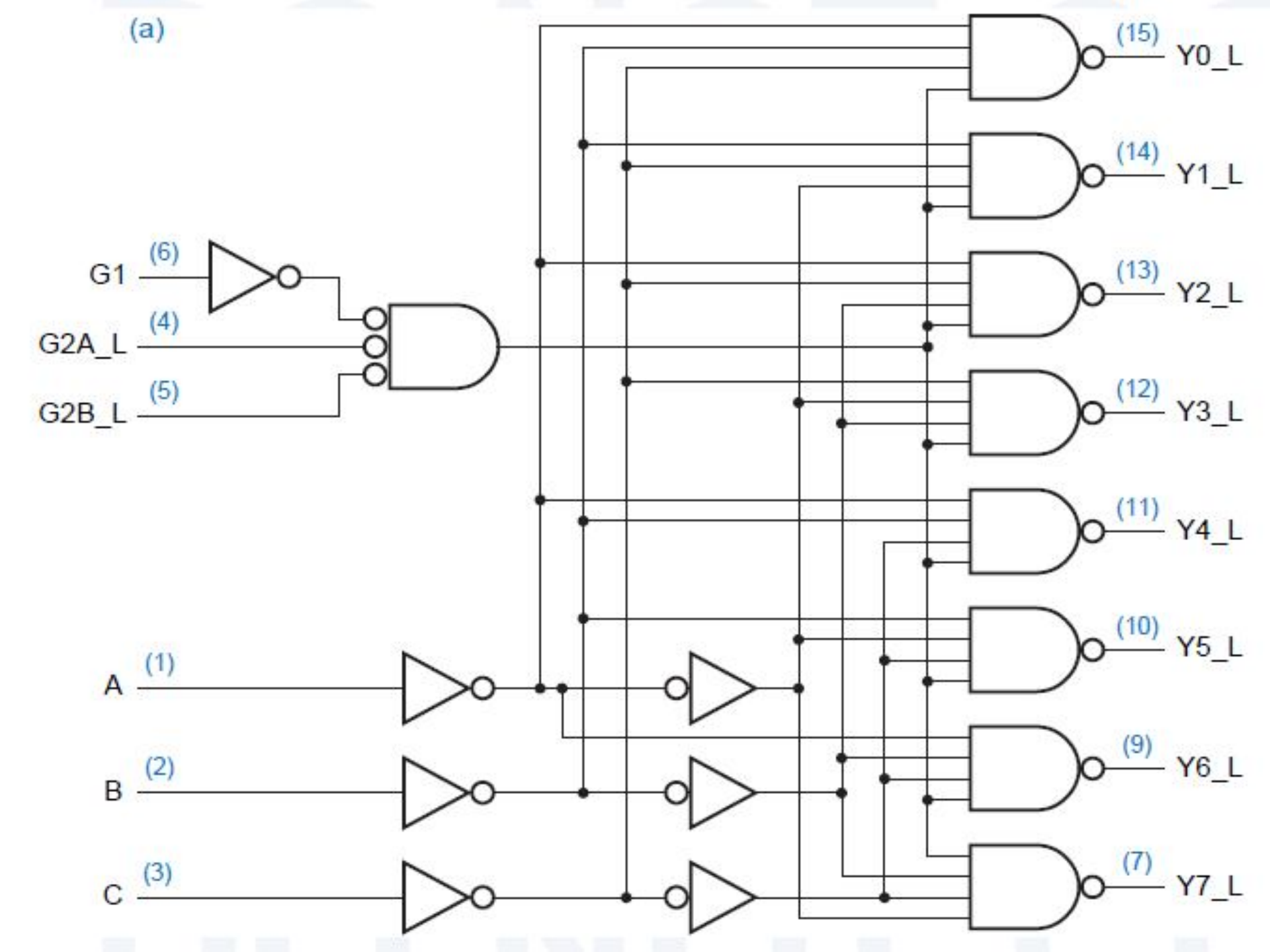
Return Back Next

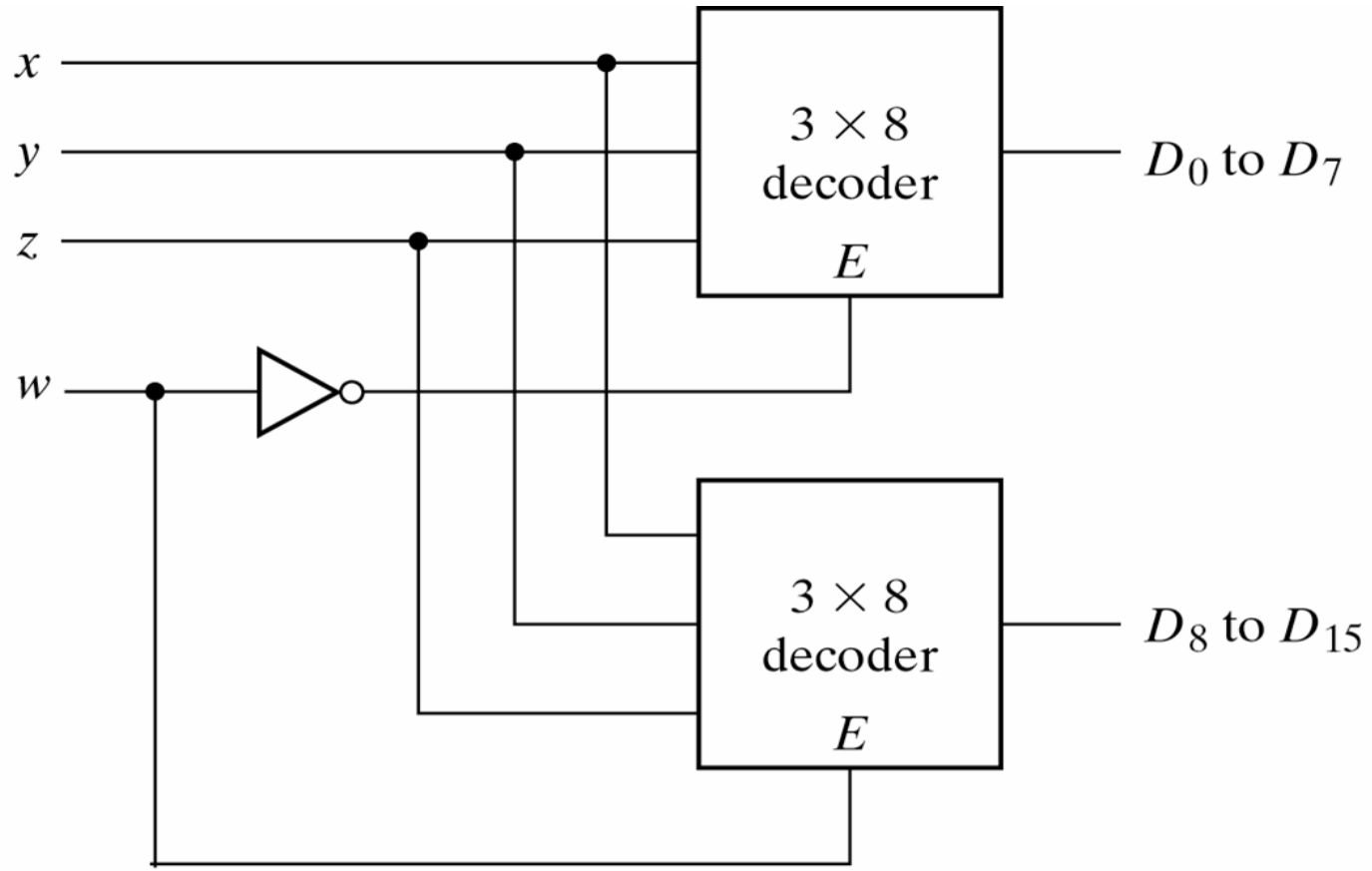
Decoders

- Traditional logic symbol of the 74x138



Return Back Next

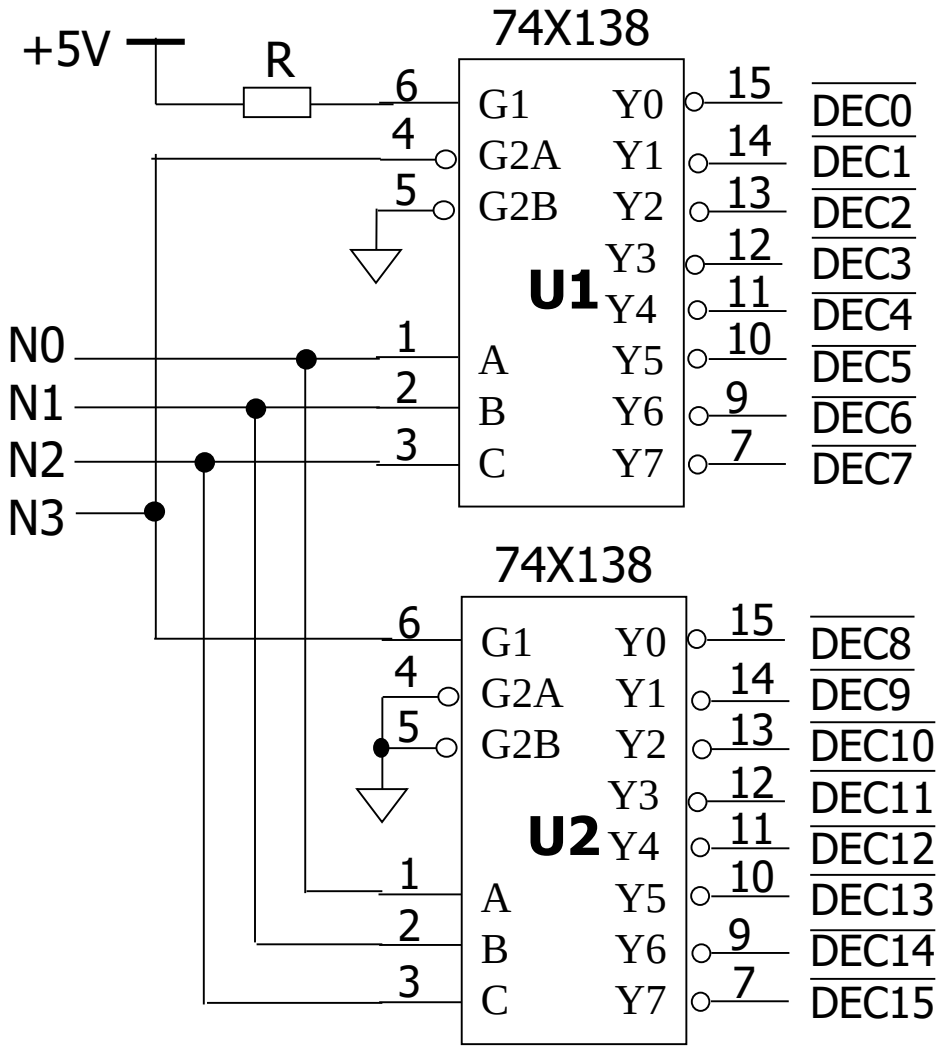

(a)
G1
(6)
G2A_L
(4)
G2B_L
(5)
A
(1)
B
(2)
C
(3)
(15)
Y0_L
(14)
Y1_L
(13)
Y2_L
(12)
Y3_L
(11)
Y4_L
(10)
Y5_L
(9)
Y6_L
(7)
Y7_L



4 -16 DECODER USING 3TO 8 DECODER CASCADING:

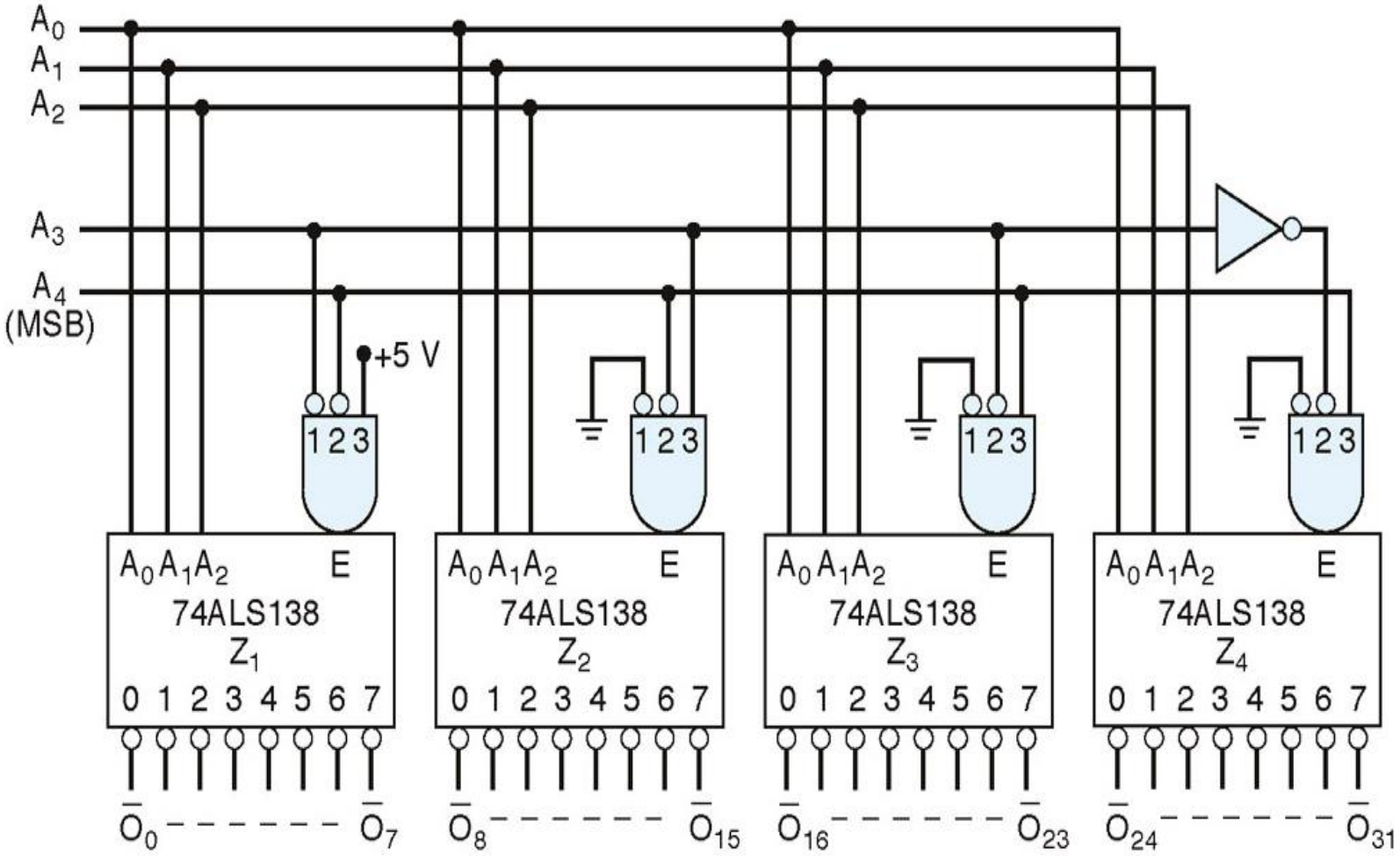
Decoders

- Cascading Binary Decoders



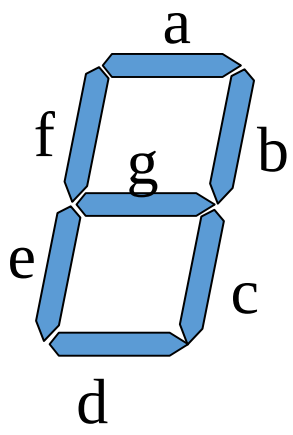
- The top decoder (U1) is enabled when N3 is 0, and the bottom one (U2) is enabled when N3 is 1.

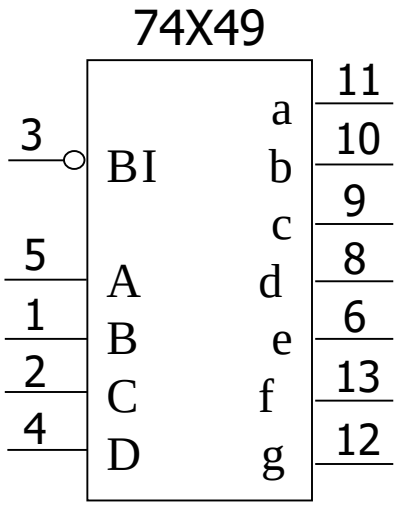
Example of a 5 to 32 Bit Decoder



Decoders

- Seven-Segment Decoders



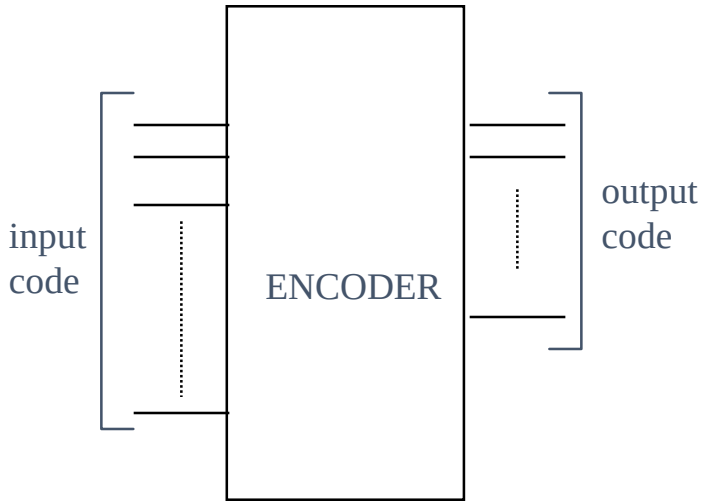


- Seven-segment display normally uses light-emitting diodes(LEDs) or liquid-crystal display(LCD) elements.
- A seven-segment decoder has 4-bit BCD as its input code and the "seven-segment code"
- Truth table for a 74x49 seven-segment decoder(See P374 table 5-21)
- Logic diagram for a 74x49 seven-segment decoder(See P373 figure 5-45)

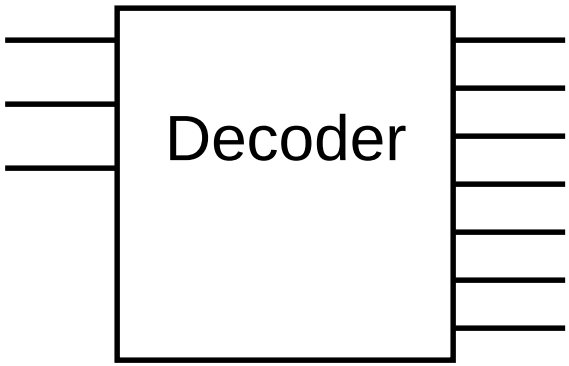
Return Back

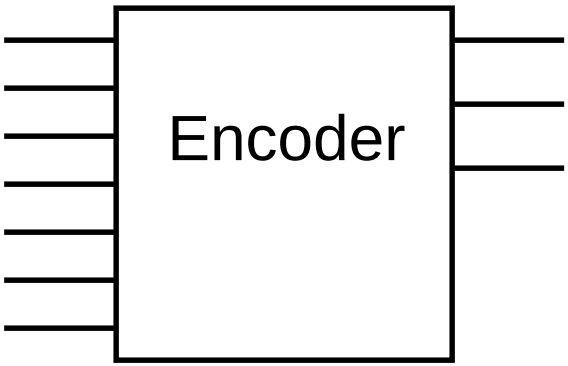
ENCODERS

- Multiple-input/multiple-output device.
- Performs the inverse function of a Decoder.
- Outputs (m) are less than inputs (n).
- Converts input code words into output code words.



Encoders vs. Decoders



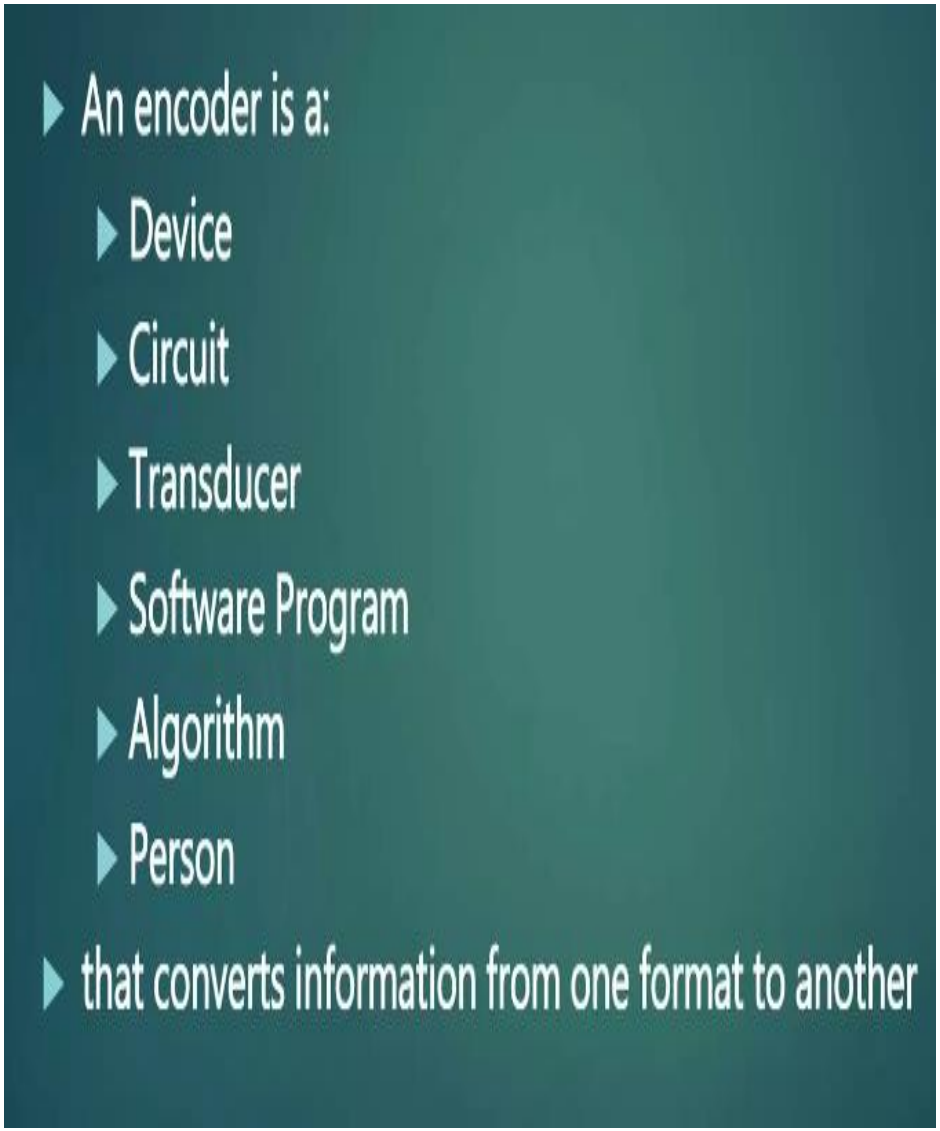


Binary decoders/encoders

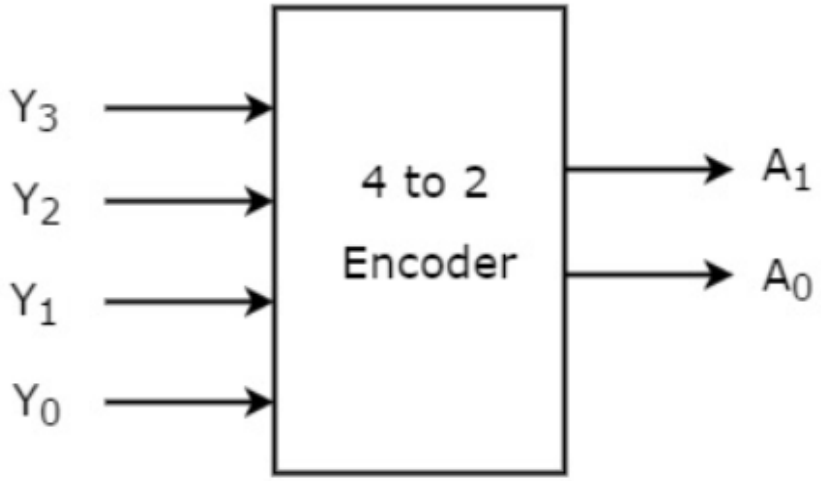
- n-to-2^n
- Input code : Binary Code
- Output code :1-out-of-2^n.

- 2^n-to-n encoder
- Input code : 1-out-of-2^n.
- Output code : Binary Code

Encoders and Decoders



- The combinational circuits that modify the binary data into N output lines are known as Encoders. The combinational circuits that convert the binary data into 2N output lines are called Decoders.
- In digital electronic projects, the encoder and decoder play an important role. It is used **to convert the data from one form to another form**.



At any time, only one of these 4 inputs can be '1' in order to get the respective binary code at the output. The **Truth table** of 4 to 2 encoder is shown below.

Inputs				Outputs	
Y_3	Y_2	Y_1	Y_0	A_1	A_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

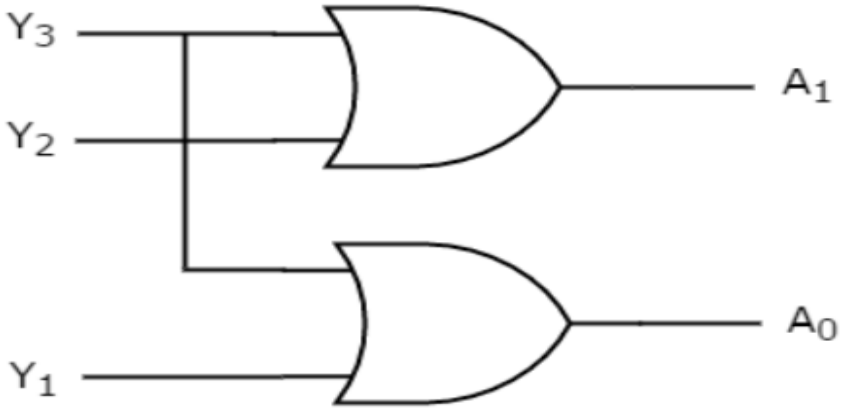
4-2 Encoder

From Truth table, we can write the **Boolean functions** for each output as

$$A_1 = Y_3 + Y_2$$

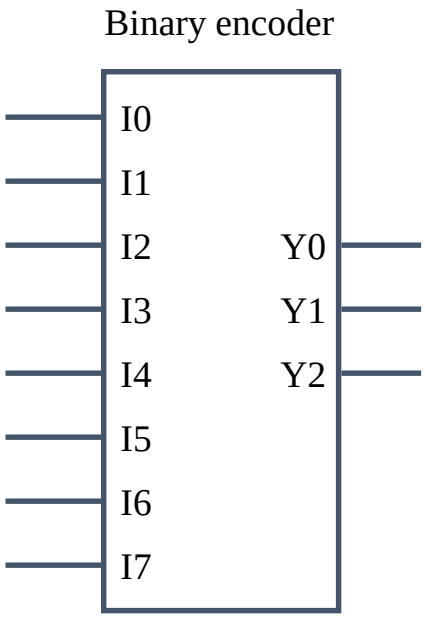
$$A_0 = Y_3 + Y_1$$

We can implement the above two Boolean functions by using two input OR gates. The **circuit diagram** of 4 to 2 encoder is shown in the following figure.



The above circuit diagram contains two OR gates. These OR gates encode the four inputs with two bits

Binary Encoder



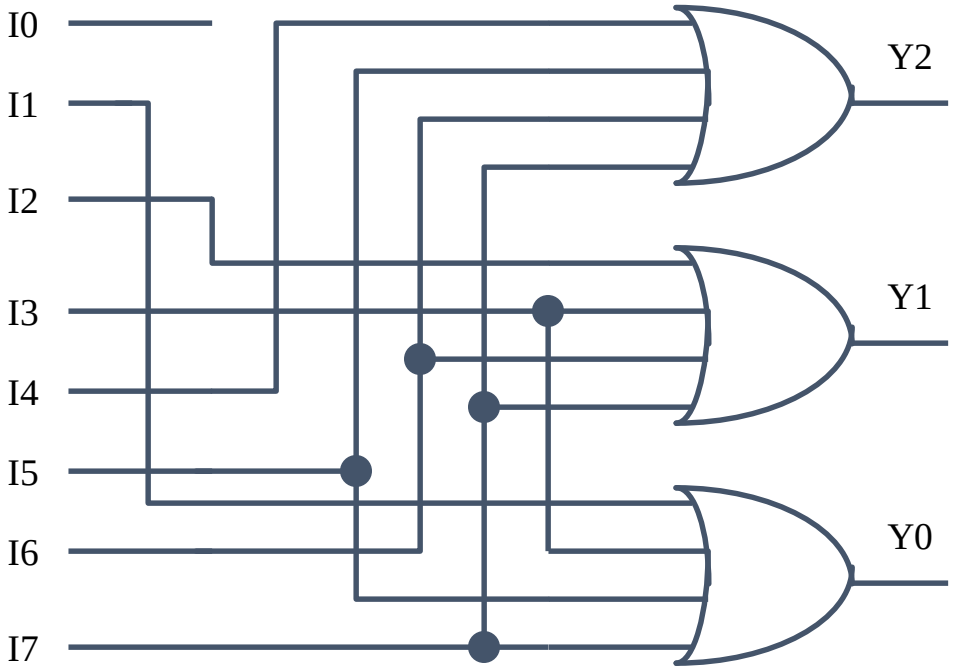
- 2^n-to-n encoder : 2^n inputs and n outputs.
- Input code : 1-out-of-2^n.
- Output code : Binary Code

- Example : n=3, 8-to-3 encoder
- Inputs Outputs

I0	I1	I2	I3	I4	I5	I6	I7	Y2	Y1	Y0
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

8-to-3 encoder Implementation

- Simplified implementation:
  - From the truth table
    Y0 = I1 + I3 + I5 + I7
    Y1 = I2 + I3 + I6 + I7
    Y2 = I4 + I5 + I6 + I7
- Limitations :
  - I0 has no effect on the output
  - Only one input can be activated
- Application:
  Handling multiple devices requests
  But, no simultaneous requests



- Establishing priorities solve the problem of multiple requests

Need of Priority Encoder

So, to overcome these difficulties, we should assign priorities to each input of encoder. Then, the output of encoder will be the $binary$ code corresponding to the active High input s , which has higher priority. This encoder is called as **priority encoder**.

Priority Encoder

A 4 to 2 priority encoder has four inputs Y_3, Y_2, Y_1 & Y_0 and two outputs A_1 & A_0. Here, the input, Y_3 has the highest priority, whereas the input, Y_0 has the lowest priority. In this case, even if more than one input is '1' at the same time, the output will be the $binary$ code corresponding to the input, which is having **higher priority**.

We considered one more **output, V** in order to know, whether the code available at outputs is valid or not.

- If at least one input of the encoder is '1', then the code available at outputs is a valid one. In this case, the output, V will be equal to 1.
- If all the inputs of encoder are '0', then the code available at outputs is not a valid one. In this case, the output, V will be equal to 0.

The **Truth table** of 4 to 2 priority encoder is shown below.

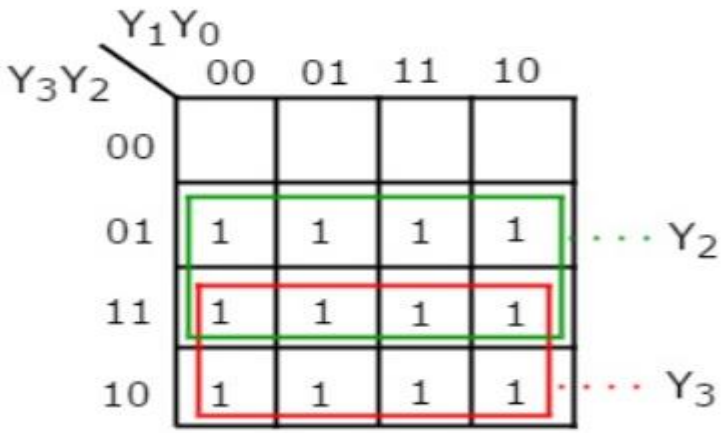
Inputs				Outputs		
Y_3	Y_2	Y_1	Y_0	A_1	A_0	V
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	x	0	1	1
0	1	x	x	1	0	1
1	x	x	x	1	1	1

Use **4 variable K-maps** for getting simplified expressions for each output.

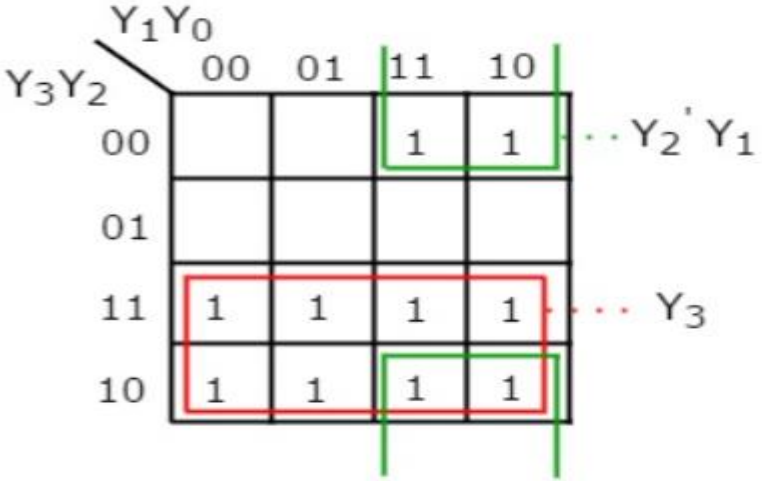
K-Map for A_1

K-Map for A_0

K-Map for A_1



K-Map for A_0



The simplified **Boolean functions** are

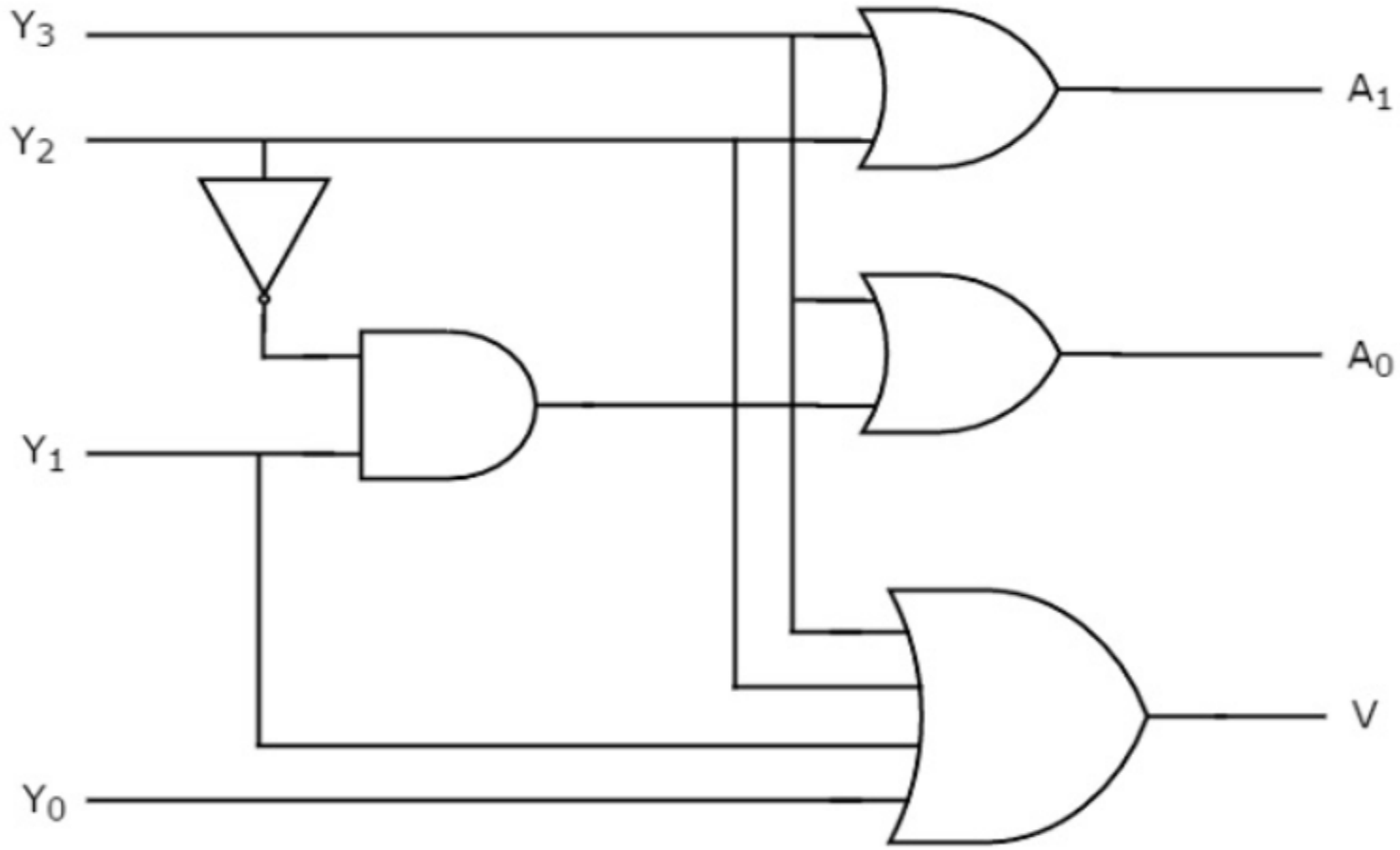
$$A_1 = Y_3 + Y_2$$

$$A_0 = Y_3 + Y_2'Y_1$$

Similarly, we will get the Boolean function of output, V as

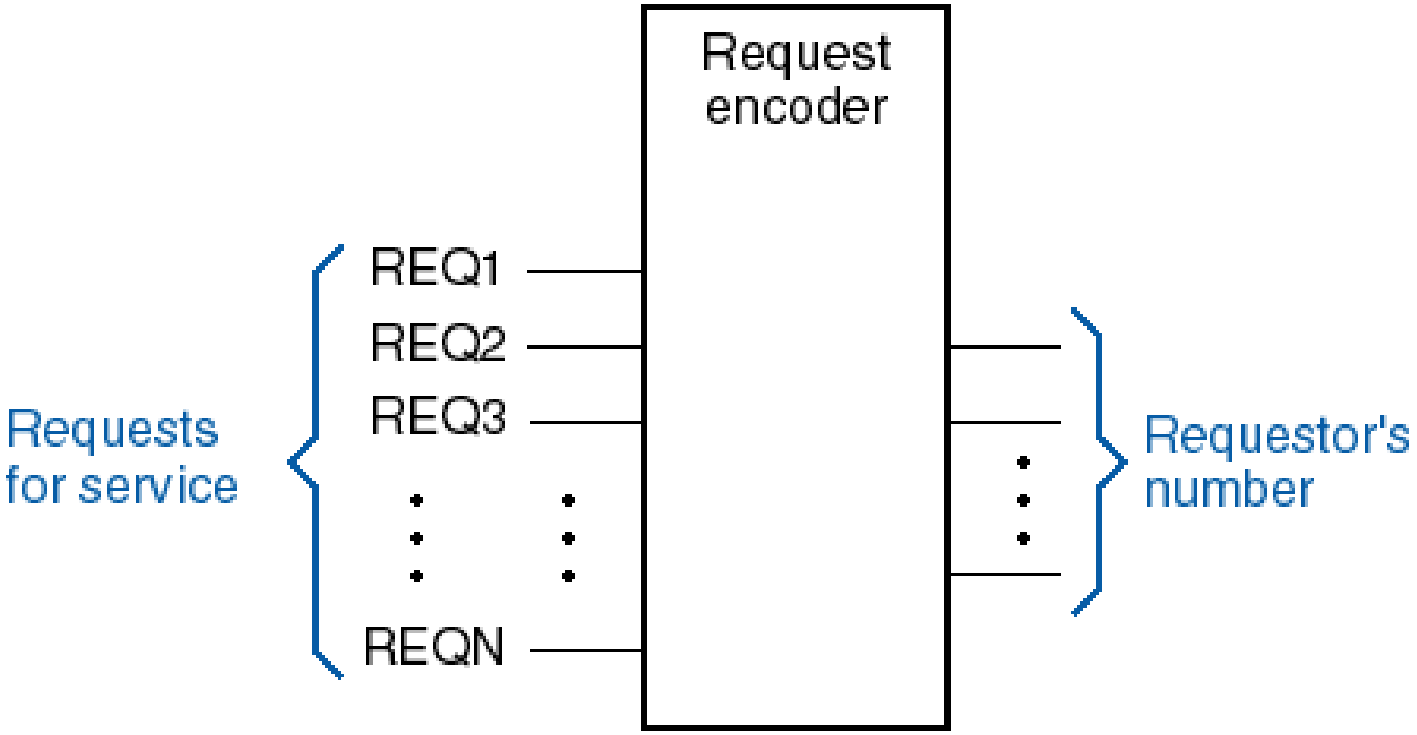
$$V = Y_3 + Y_2 + Y_1 + Y_0$$

4-2 Priority encoder



The above circuit diagram contains two 2-input OR gates, one 4-input OR gate, one 2input AND gate & an inverter. Here AND gate & inverter combination are used for producing a valid code at the outputs, even when multiple inputs are equal to '1' at the same time. Hence, this circuit encodes the four inputs with two bits based on the **priority** assigned to each input.

Need priority in most applications



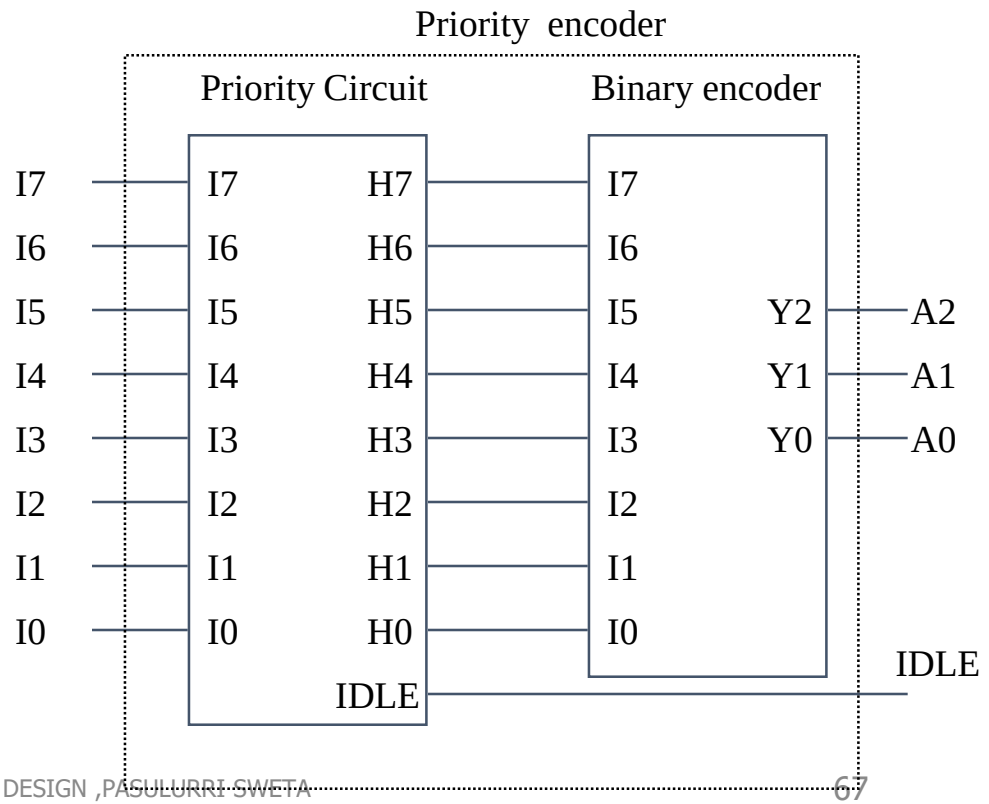
Priority Encoder

- Assign priorities to the inputs
- When more than one input are asserted, the output generates the code of the input with the highest priority
- Priority Encoder :

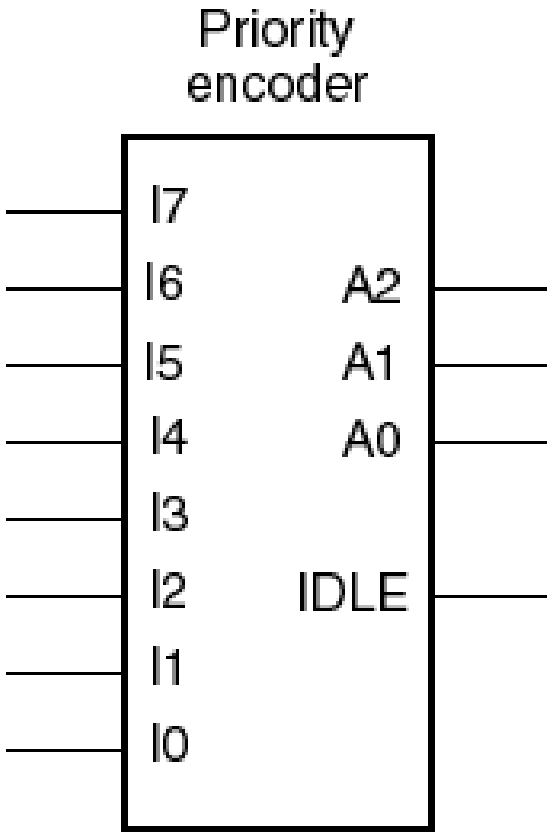
H7=I7 (Highest Priority)
H6=I6.I7'
H5=I5.I6'.I7'
H4=I4.I5'.I6'.I7'
H3=I3.I4'.I5'.I6'.I7'
H2=I2.I3'.I4'.I5'.I6'.I7'
H1=I1. I2'.I3'.I4'.I5'.I6'.I7'
H0=I0.I1'. I2'.I3'.I4'.I5'.I6'.I7'
IDLE= I0'.I1'. I2'.I3'.I4'.I5'.I6'.I7'

- Encoder

A0=Y0 = H1 + H3 + H5 + H7
A1=Y1 = H2 + H3 + H6 + H7
A2=Y2 = H4 + H5 + H6 + H7

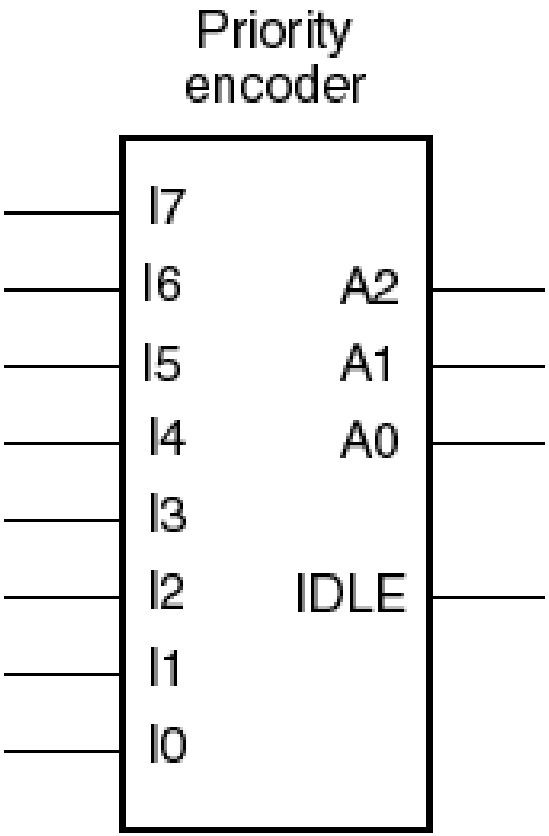


8-input priority encoder

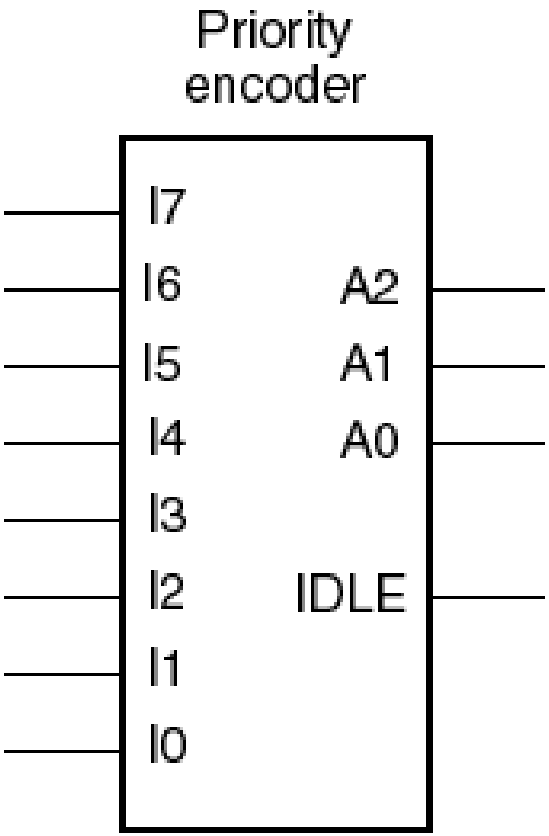


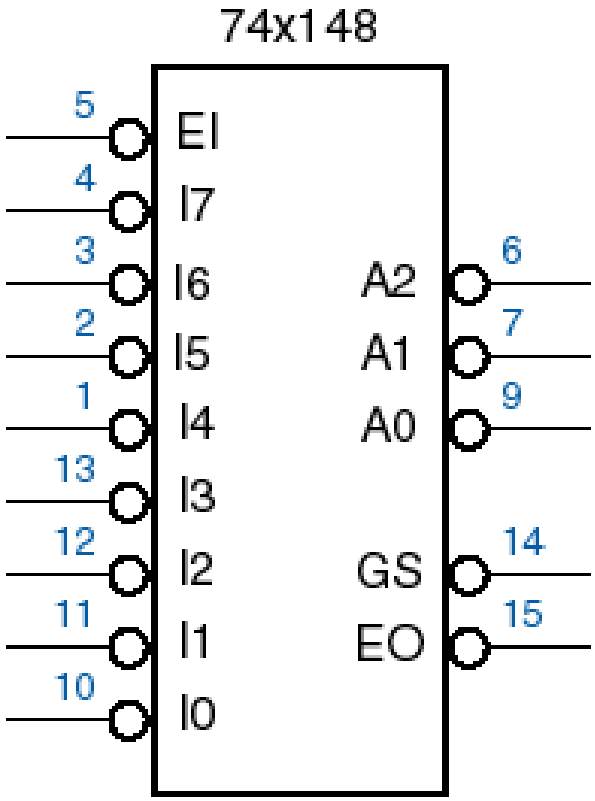
- I7 has the highest priority,I0 least
- A2-A0 contain the number of the highest-priority asserted input if any.
- IDLE is asserted if no inputs are asserted.

74x148 8-input priority encoder



74x148 8-input priority encoder



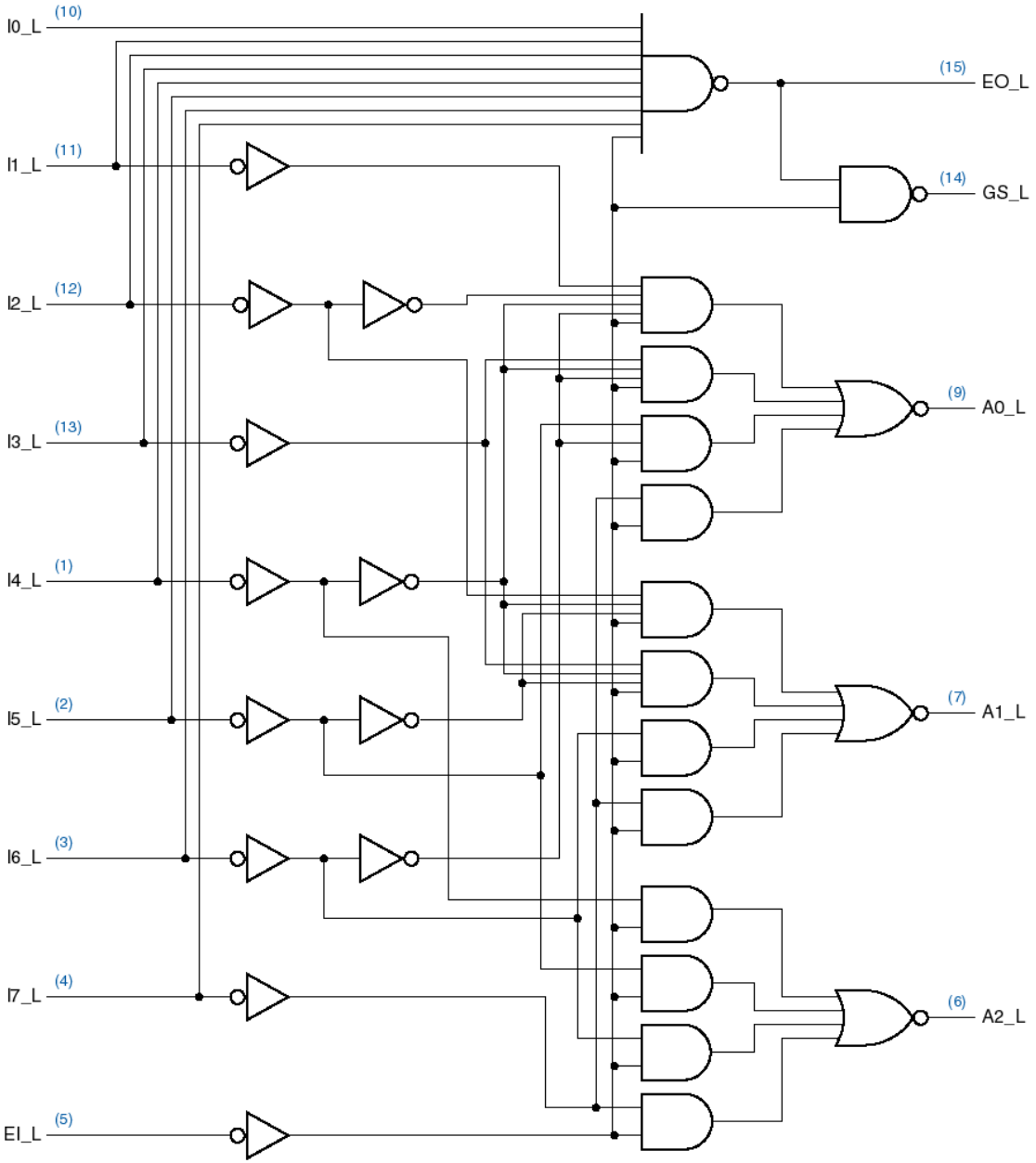


- Active-low I/O
- Enable Input
- “Got Something": Group Select
- Enable Output

74x148 Truth Table

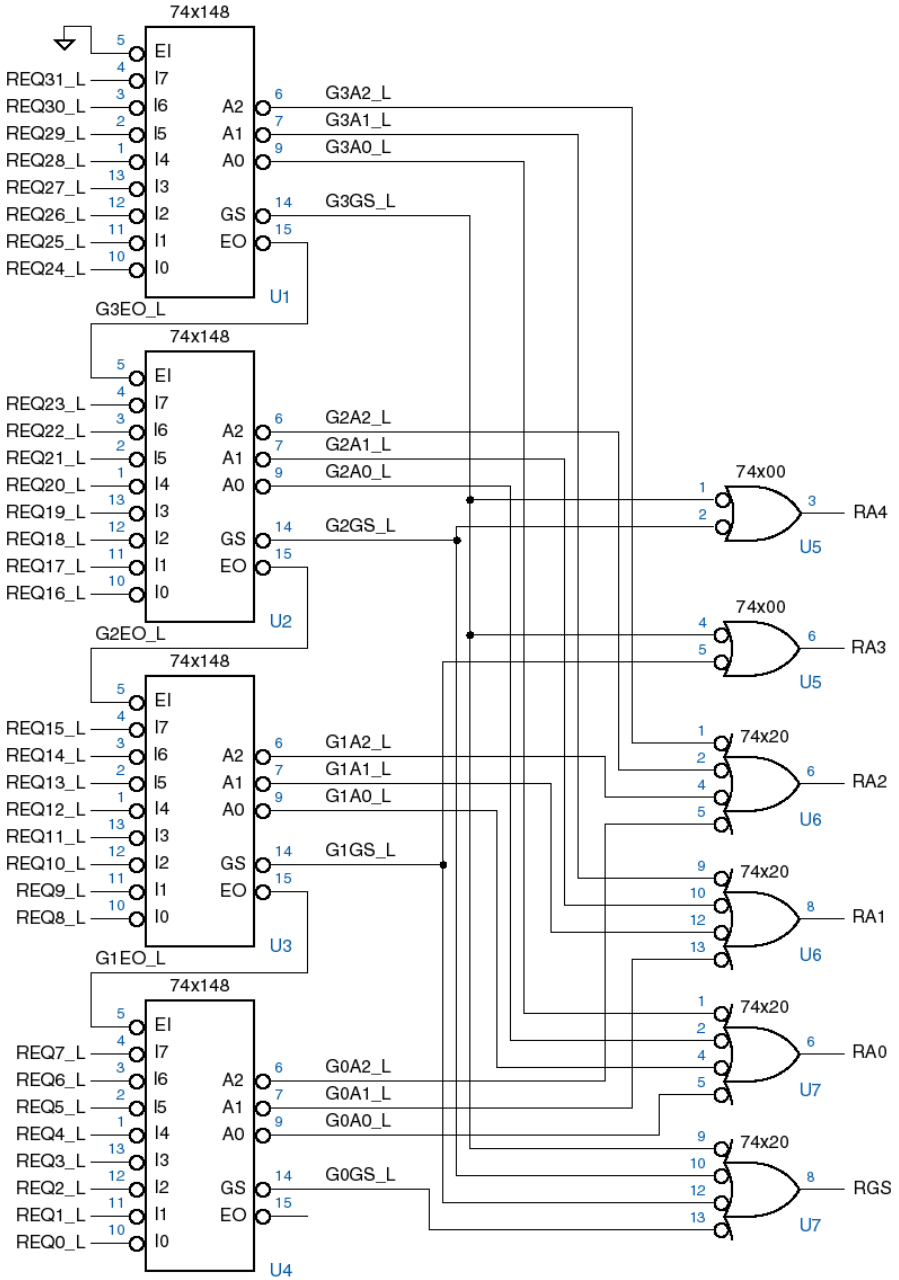
Inputs									Outputs				
EI_L	I0_L	I1_L	I2_L	I3_L	I4_L	I5_L	I6_L	I7_L	A2_L	A1_L	A0_L	GS_L	EO_L
1	x	x	x	x	x	x	x	x	1	1	1	1	1
0	x	x	x	x	x	x	x	0	0	0	0	0	1
0	x	x	x	x	x	x	0	1	0	0	1	0	1
0	x	x	x	x	x	0	1	1	0	1	0	0	1
0	x	x	x	x	0	1	1	1	0	1	1	0	1
0	x	x	x	0	1	1	1	1	1	0	0	0	1
0	x	x	0	1	1	1	1	1	1	0	1	0	1
0	x	0	1	1	1	1	1	1	1	1	0	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0

74x148 circuit



Cascading priority encoders

- 32-input priority encoder



Multiplexers

- A multiplexer is a digital switch

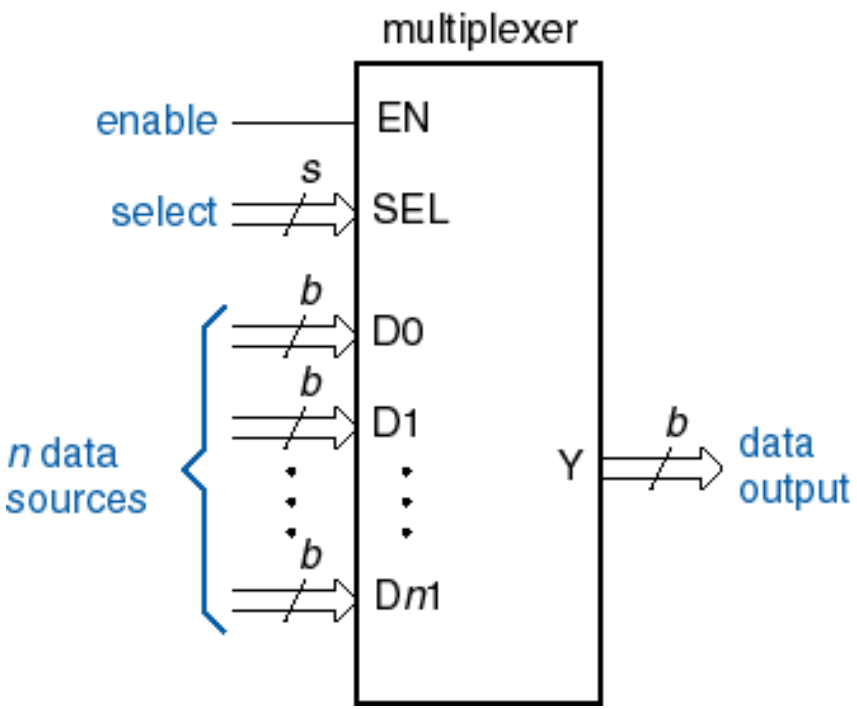
  - it connects data from one of *n* sources to its output.

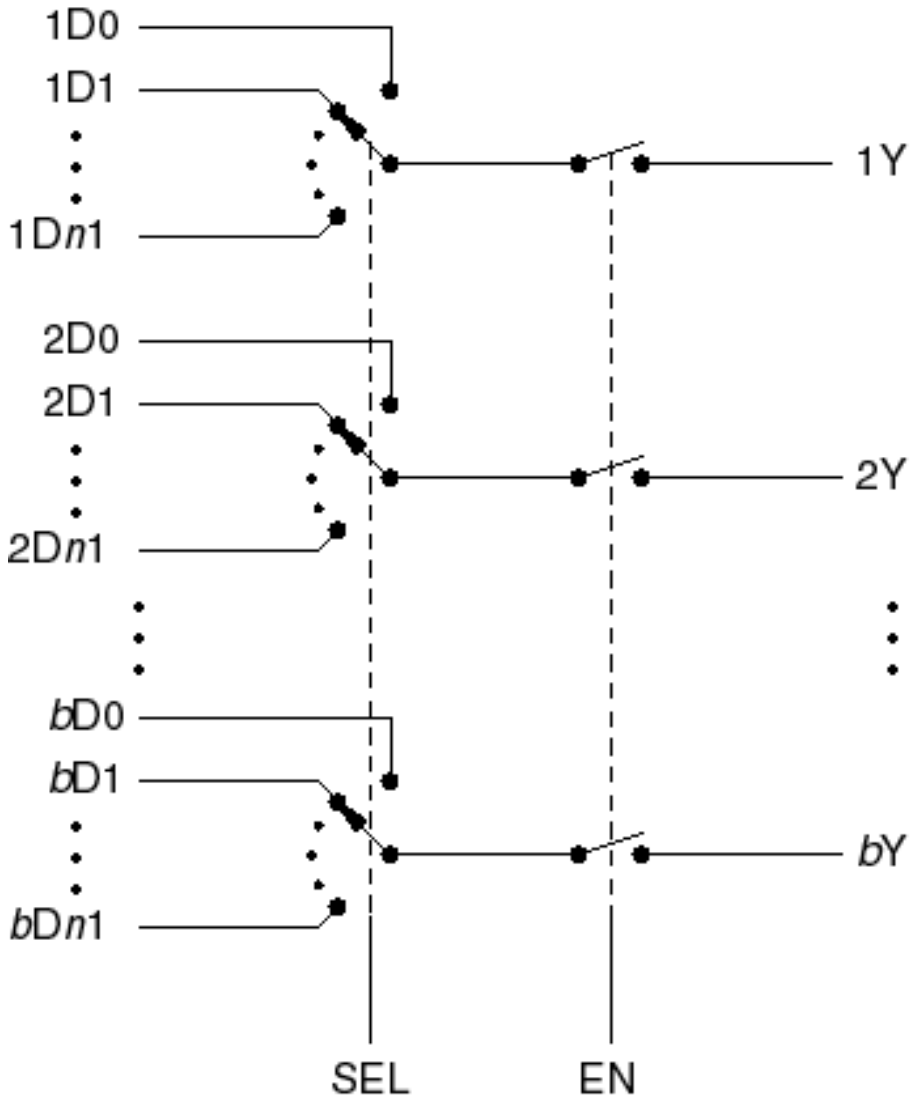
Multiplexers or Data Selectors

- An n-input and b-bit multiplexer has n soureces of data, each of which b bits wide, and there are b output bits.
- A multiplexer is a unidirectional device.
- Multiplexers are used in any application in which data must be switched from multiple sources to a destination.

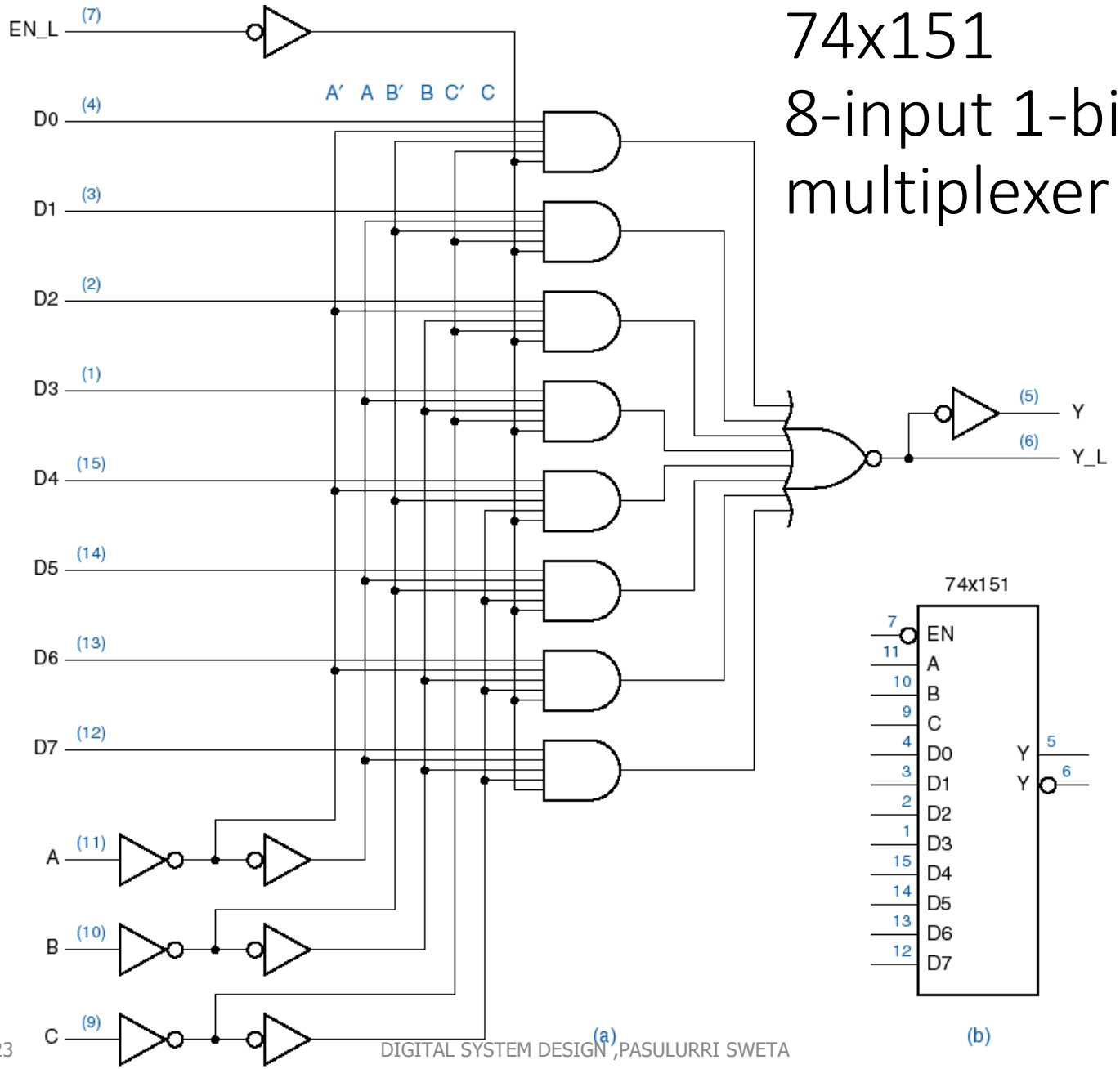
  e.g., processor's registers to ALU

Multiplexers





74x151
8-input 1-bit multiplexer



Multiplexers IC Specifications

74x151

- 8-input, 1-bit-wide Multiplexer
- 8-input, 1-bit Multiplexer
- 1-bit, 8-to-1 Multiplexer
- 1, 8-to-1 Multiplexer
- 1, 1-of-8 Data Selector
- Single, 1-of-8 Data Selector

74x157

- 2-input, 4-bit-wide Multiplexer
- 2-input, 4-bit Multiplexer
- 4-bit, 2-to-1 Multiplexer
- 4, 2-to-1 Multiplexer
- Quadruple 2-line to 1-line Data Selector/Multiplexer
- Quad, 1-of-2 Data Selector

74x153

- 4-input, 2-bit-wide Multiplexer
- 2-bit, 4-to-1 Multiplexer
- 2, 1-of-4 Data Selector

74x151 truth table

Inputs				Outputs	
EN_L	C	B	A	Y	Y_L
1	x	x	x	0	1
0	0	0	0	D0	D0′
0	0	0	1	D1	D1′
0	0	1	0	D2	D2′
0	0	1	1	D3	D3′
0	1	0	0	D4	D4′
0	1	0	1	D5	D5′
0	1	1	0	D6	D6′
0	1	1	1	D7	D7′

Other multiplexer varieties

- 2-input, 4-bit-wide
  - 74x157

Inputs		Outputs			
G_L	S	1Y	2Y	3Y	4Y
1	x	0	0	0	0
0	0	1A	2A	3A	4A
0	1	1B	2B	3B	4B

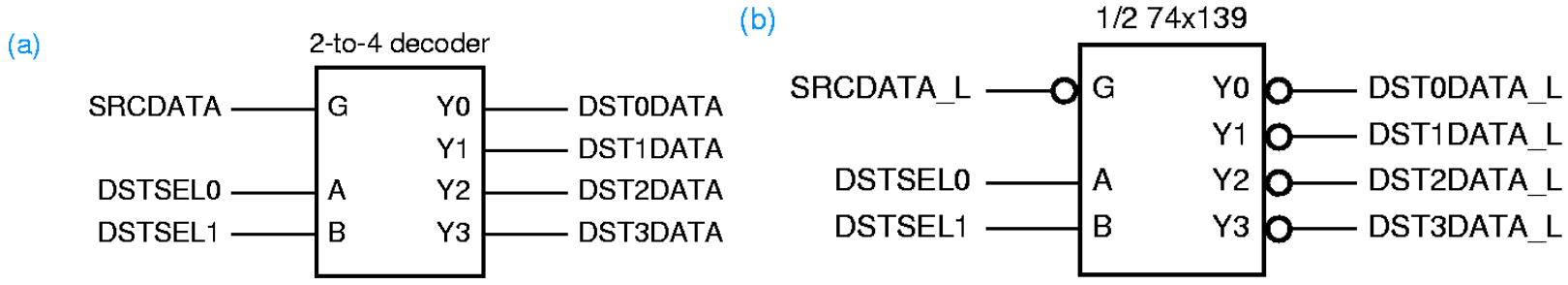
- 4-input, 2-bit-wide
  - 74x153

Multiplexers, De-multiplexers

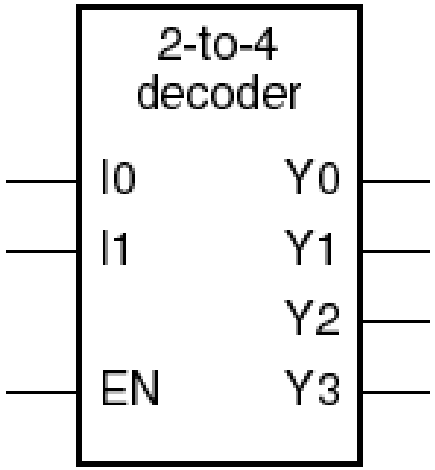
- A multiplexer is used to select one of n sources of data to transmit on a bus.
- A demultiplexer is opposite of a multiplexer.
- A 1-bit, n-output demultiplexer has one data input and s inputs to select one of $n = 2^s$ data outputs.
- A b-bit, n-output demultiplexer has b-data input and s inputs to select one of $n = 2^s$ sets of b-data outputs.

Decoder/Demultiplexers

- A binary decoder with an enable input can be used as a demultiplexer.
- The decoder's enable input is connected to the data line, and its select inputs determine which of its output lines is driven with the data bit.



Binary 2-to-4 decoder

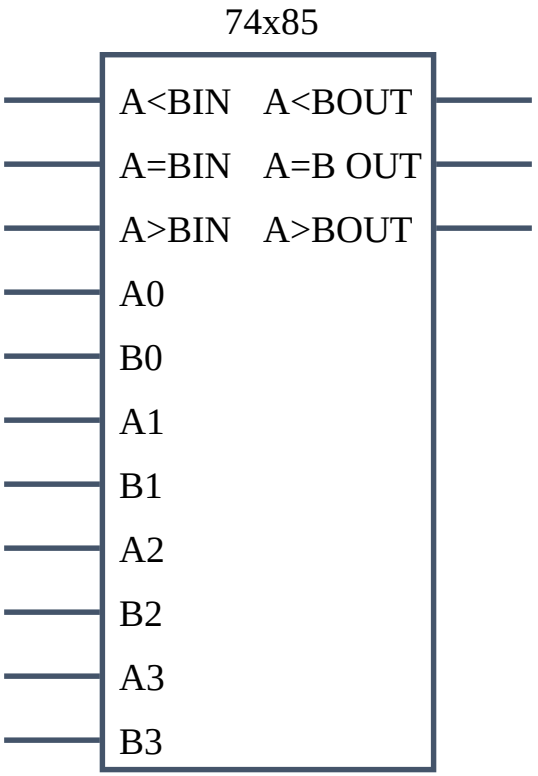


Inputs			*Outputs*			
EN	I1	I0	Y3	Y2	Y1	Y0
0	x	x	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

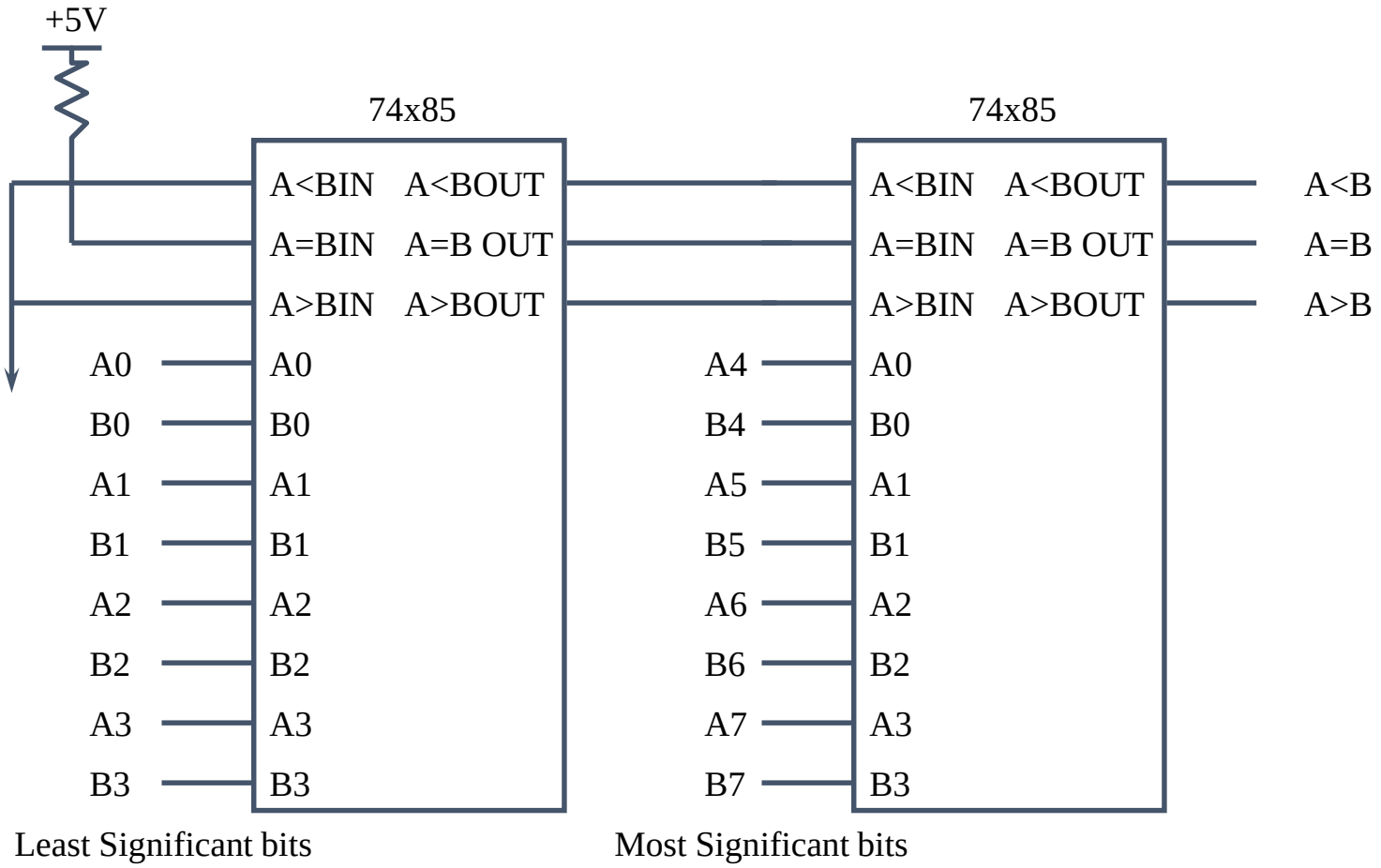
Note “x” (don’t care) notation.

MSI Comparator : 74x85

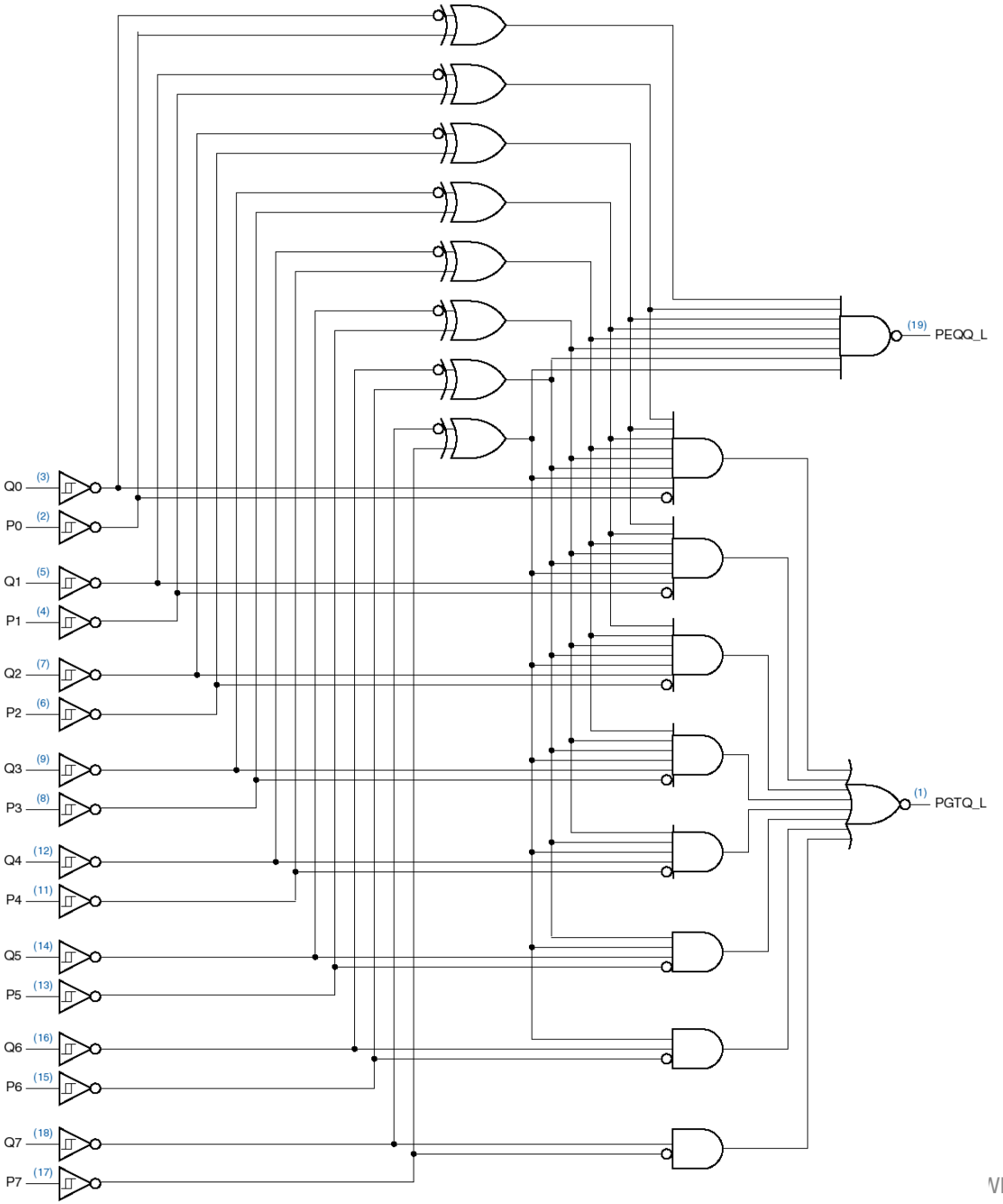
- 4 bit comparator
- 3 outputs : A=B, A<B, A>B
- 3 Cascading inputs
- Functional Output equations :
  (A>B OUT)= (A>B)+(A=B).(A>B IN)
  (A<B OUT)= (A<B)+(A=B).(A<B IN)
  (A=B OUT)= (A=B).(A=B IN)
- Cascading inputs initial values :
  (A=B IN) =1
  (A>B IN) =0
  (A<B IN) =0

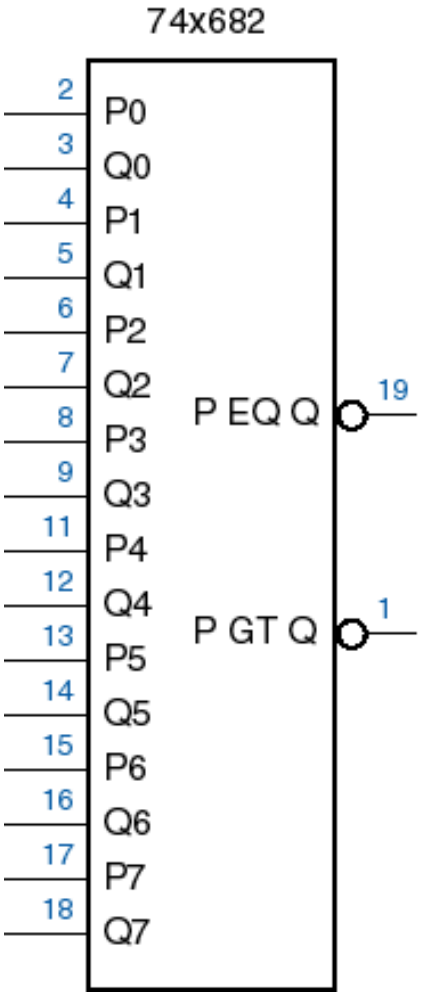


8 bit Comparator



8-bit Magnitude Comparator





Hazards

- A hazard, if exists, in a digital circuit **causes a temporary fluctuation in output of the circuit** or a hazard is a temporary disturbance in ideal operation of the circuit which if given some time, gets resolved itself.
- Hazards are unwanted switching transients that appear at the output of a circuit due to different propagation delays of different paths. i.e., the unwanted switching which appears in the output, will be very short in duration, like a glitch that will be removed after some time. Such a transient is also called a glitch or a spike that occurs due to the Hazardous behaviour of a circuit.
- Static hazard occur when an input changes and it causes the output to change at the same moment before output becomes stable.
- Dynamic hazard occur when output changes for two adjacent inputs while the output should change only once.

Types of Hazards in Digital Electronics

1.Static Hazards

- Static ‘1’-SOP & Static- ‘0’ -POS

2.Dynamic Hazards

- Happened in multilevel circuits

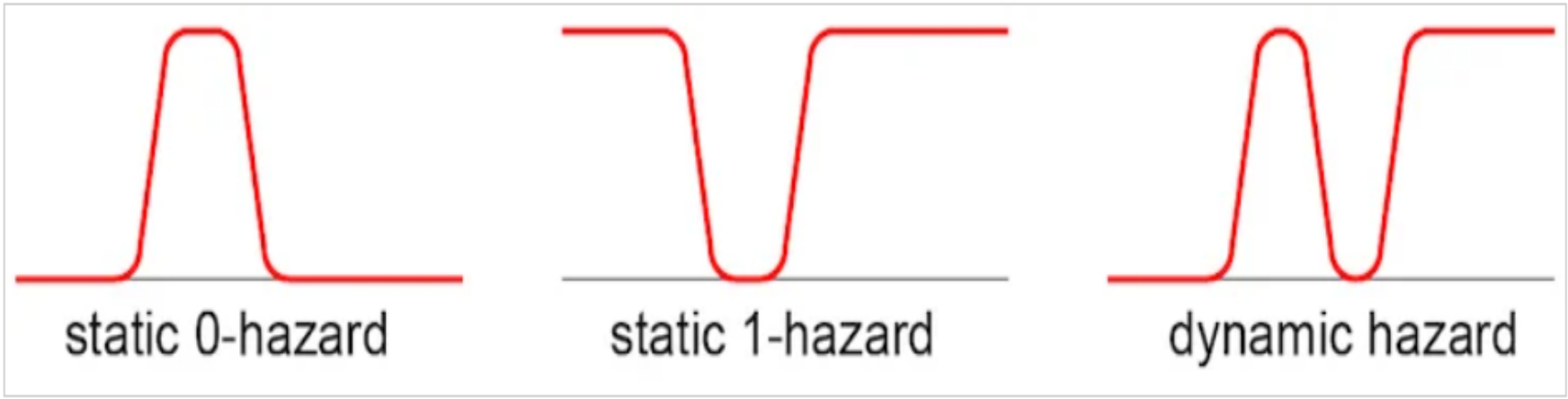
3.Essential Hazards

- Happened in sequential circuits

What Is a Logic Hazard?

In complex logic circuits, undesired and temporary switching events may occur at outputs. Figure 1 illustrates three types of hazards that can occur in combinational logic circuits:

- **Static 0-hazard:** The output temporarily changes to 1 when it should have stayed a 0
- **Static 1-hazard:** The output temporarily changes to 0 when it should have stayed a 1
- **Dynamic hazard:** The output changes multiple times when it should have made a single logic transition; either from 0 to 1 or 1 to 0



Static Hazards

- S-Hazards is those where the signal level should have been constant but it changes for a small amount of time.
- If the signal is ‘1’ for all time but due to some static hazard it will become from 1 to 0 for a small amount of time, or If a signal is ‘0’ for all time but due to some static hazards it will become from ‘0’ to ‘1’.
- **if any static hazard comes in digital circuits then both static ‘1’ and static ‘0’ hazard will come in the circuit simultaneously. Only static ‘1’ or only static ‘0’ hazard will not generate in a digital circuit.**

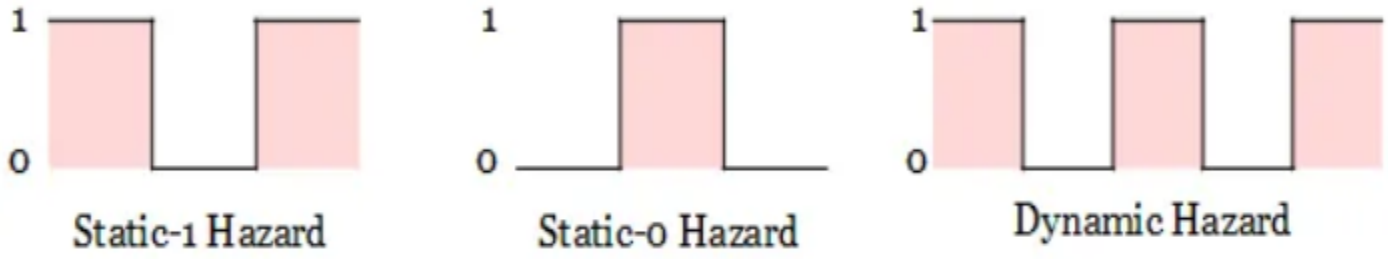
- **Static 1 Hazard:**
- Static 1 hazard occurs due to different delays experienced by the signal through the Gates connected in circuits.
- Static 1 hazard always occurs in SOP (Sum of Product) terms.

- **Static 0 Hazard:**
- Static 0 hazard occurs due to different delays experienced by the signal through the Gates connected in circuits.
- Static 0 hazard always occurs in POS (Product of Sum) terms.
- **Static hazard can be eliminated by adding redundant terms which increases the hardware but removes the glitch.**

Static and Dynamic Hazard

When the input to the logic circuit changes, the output has to remain at a particular logic, but instead it may change its value momentarily. If this happens in the logic circuit, then there exists a **static hazard**.

The static hazard can be classified as a static-0 hazard and static-1 hazard. Instead of remaining at logic 1, the output goes to logic 0 momentarily, then it is a **static-1 hazard**. Similarly, if the output monetarily goes to logic 1, instead of remaining at 0, it is a **static-0 hazard**.



A hazard is said to be **dynamic** if the output changes two or more times when it should change from 1 to 0 or from 0 to 1.

Dynamic Hazards

- Dynamic Hazard occurs during a multilevel circuit where the output must make a transition from 0 to 1 or from 1 to 0 but the output makes multiple transitions then settles to a final value.
- Dynamic hazard occurs when the output changes for 2 adjacent input combinations while changing, the output should change on just one occasion . But it's going to change three or more times briefly intervals due to different delays in several paths.
- Dynamic hazards occur only in multilevel circuits.
- Dynamic hazards are more complex to resolve but note that if all static hazards are eliminated from a circuit, then dynamic hazards cannot occur

Static & Dynamic Hazards

- Static hazard occur when an input changes and it causes the output to change at the same moment before output becomes stable.
- Dynamic hazard occur when output changes for two adjacent inputs while the output should change only once.
- A static hazard is **a change of a signal state twice in a row when the signal is expected to stay constant**. When one input signal changes, the output changes momentarily before stabilizing to the correct value.
- "A dynamic hazard is **the possibility of an output changing more than once as a result of a single input change**" Dynamic hazards often occur in larger logic circuits where there are different routes to the output (from the input).

Essential Hazards

- This type of hazard is caused by unequal delays along two or more paths that originate from an equivalent input.
- An excessive delay through an inverter circuit as compared to the delay related to the feedback path may cause such a hazard.
- Essential hazards can’t be corrected by adding redundant gates as in static hazards.
- To avoid essential hazards, each feedback circuit must be handled with individual care to make sure that the delay within the feedback path is long enough compared with delays of other signals that originate from the input terminals.
- https://www.allaboutcircuits.com/technical-articles/hazards-in-combinational-logic/

Unit- 2 last topic
Programmable Logic Devices (PLD'S)

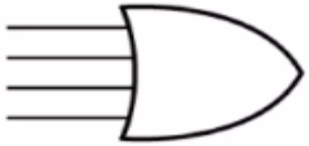
- Programmable Read-Only Memories (PROM)
- Programmable Logic Arrays (PLAs)
- Programmable Array Logic (PAL)

- A combinational PLD is **an integrated circuit with programmable gates divided into an AND array and an OR array to provide an AND-OR sum of product implementation**. There are three major types of combinational PLDs and they differ in the placement of the programmable connections in the AND-OR array.

PLD's

- A programmable logic device (PLD) is **an electronic component used to build reconfigurable digital circuits**. Unlike digital logic constructed using discrete logic gates with fixed functions, a PLD has an undefined function at the time of manufacture.

- In order to show the internal logic diagram in a concise form, it is necessary to employ a special gate symbology applicable to array logic.



(a) Conventional symbol

(b) Array logic symbol

General Structure of PLD

- Inputs to the PLD are applied to a set of buffer/inverters. These devices have both the true value of the input as well as the complemented value of the input as its outputs.
- Outputs from these devices are the inputs to an array of and-gates. The AND array generates a set of p product terms.
- The product terms are inputs to an array of or-gates to realize a set of m sum-of-product expressions.

General Structure of PLD

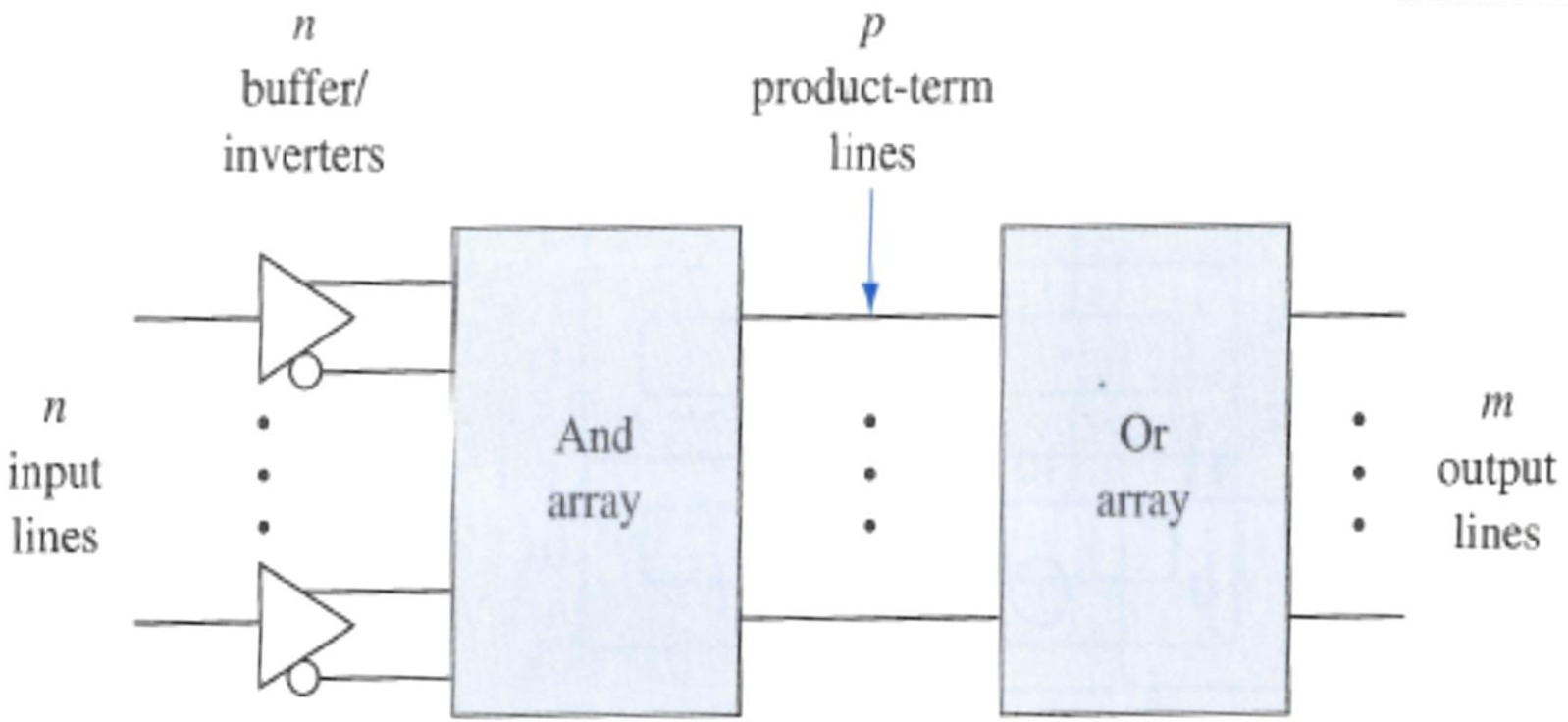


Figure 5.48 General structure of PLDs.

General Structure of PLD

- One or both of the gate arrays are programmable.
- The logic designer can specify the connections within an array.
- PLDs serve as general circuits for the realization of a set of Boolean functions.

Device	AND-array	OR-array
PROM	Fixed	Programmable
PLA	Programmable	Programmable
PAL	Programmable	Fixed

Programming a PLD

- In a programmable array, the connections to each gate can be modified.
- Simple approach is to have each of the gate inputs connected to a fuse.

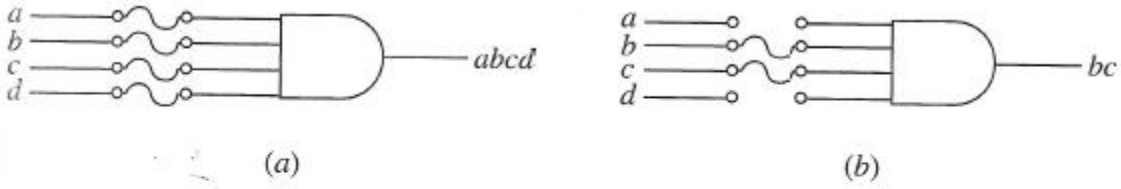


Figure 5.50 Programming by blowing fuses. (*a*) Before programming. (*b*) After programming.

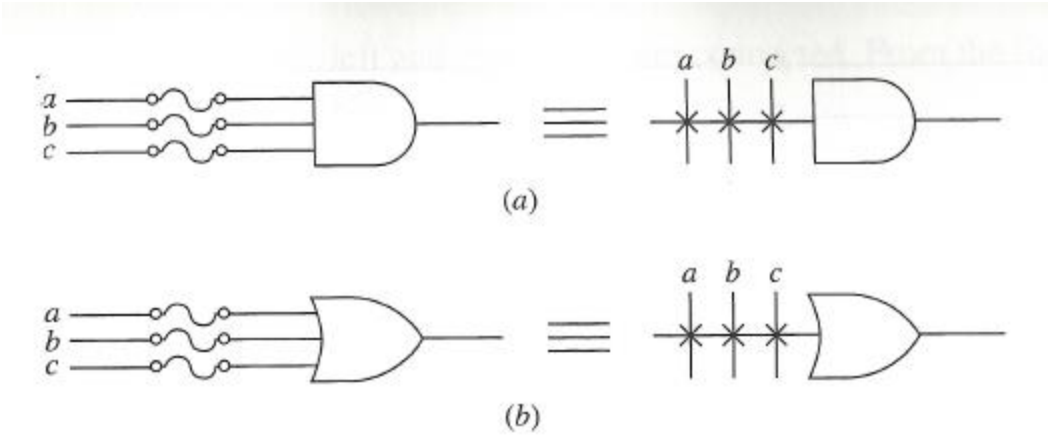
- Gate realizes the product term $abcd$.
- To generate the product term bc we remove the a, d connections by blowing the corresponding fuses.
- Thus, programming is a hardware procedure. Specialized equipment called programmers is needed to carry out the programming of a PLD.

Programming a PLD

- Erasable PLD—connections can be reset to their original conditions and then reprogrammed.
  - Can be achieved by exposing the PLD to ultraviolet light or using electrical signals
- PLDs programmed by a user are called field programmable.
- User can also specify the desired connections and supply the information to the manufacturer. Manufacturer prepares an overlay that is used to complete the connections as the last step in the fabrication process.
- Such PLDs are called mask programmable.

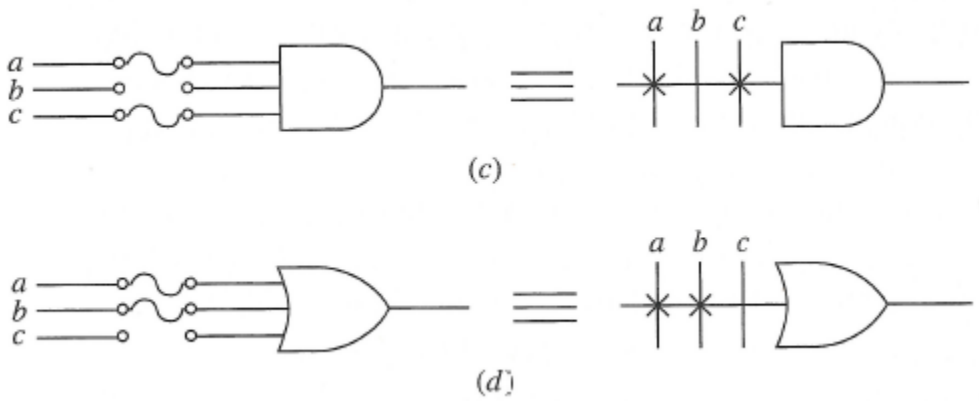
PLD Notation

- Simplified notation. Each gate has only a single input line.
- Inputs are indicated by lines at right angles to the single gate lines.
- A cross at the intersection denotes a fusible link is intact.



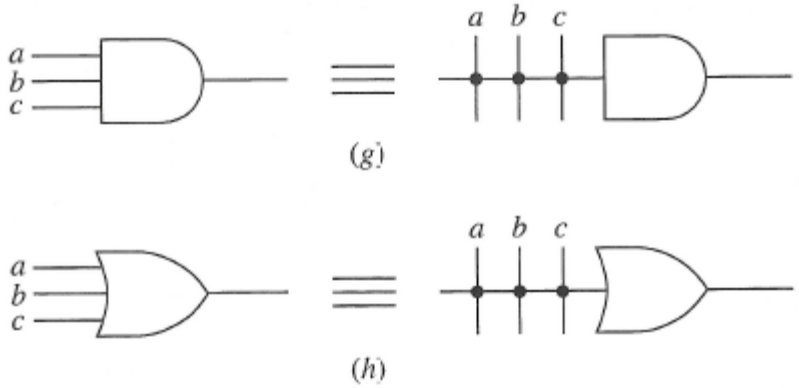
PLD Notation

- Lack of cross indicates the fuse is blown or no connection exists.

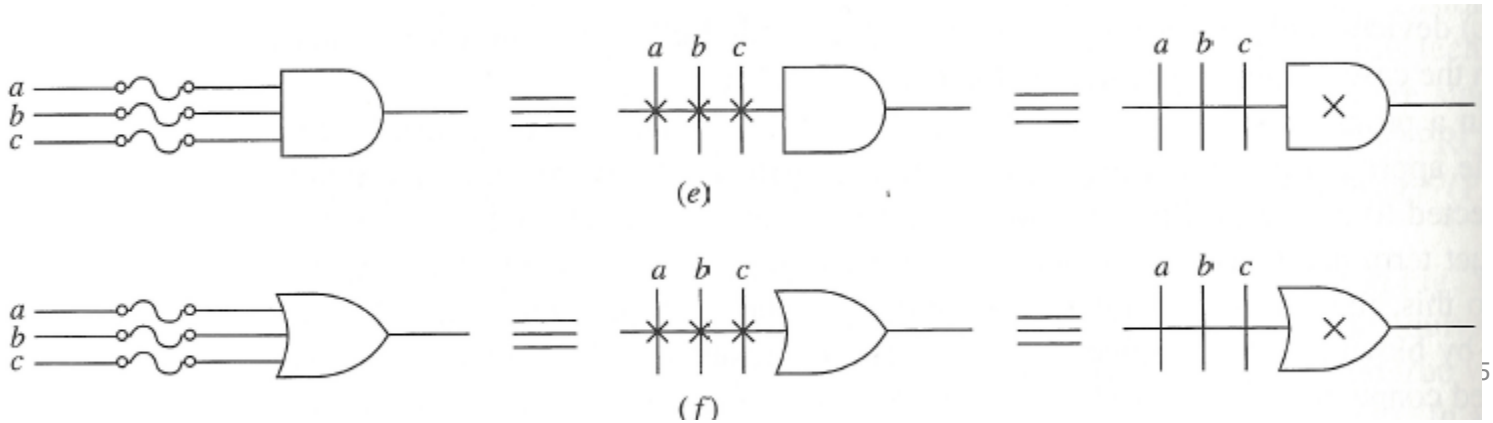


PLD Notation

- The occurrence of a hard-wired connection that is not fusible is indicated by a junction dot.



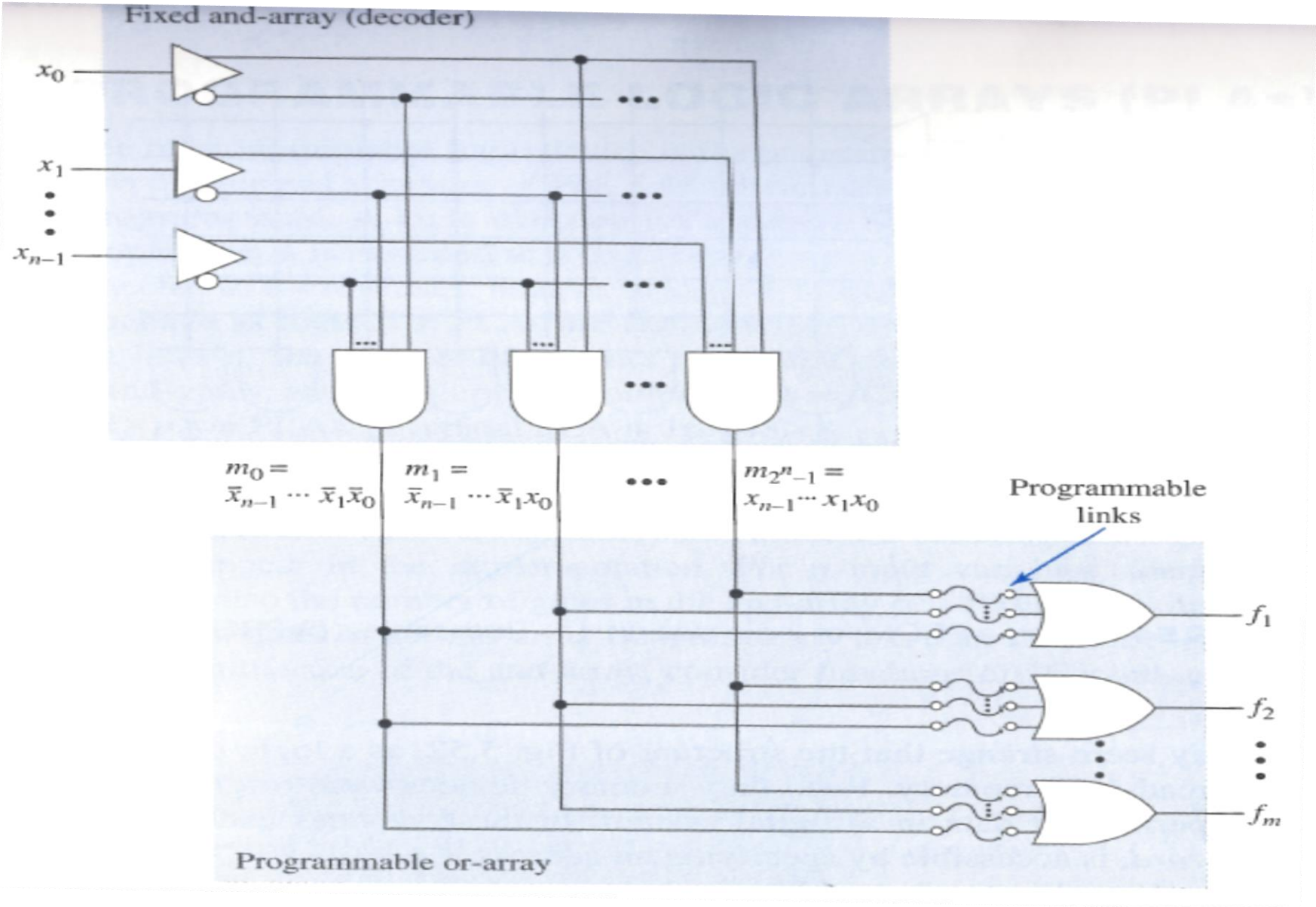
- For the special case when all the input fuses to a gate are kept intact, a cross is placed inside the gate symbol.



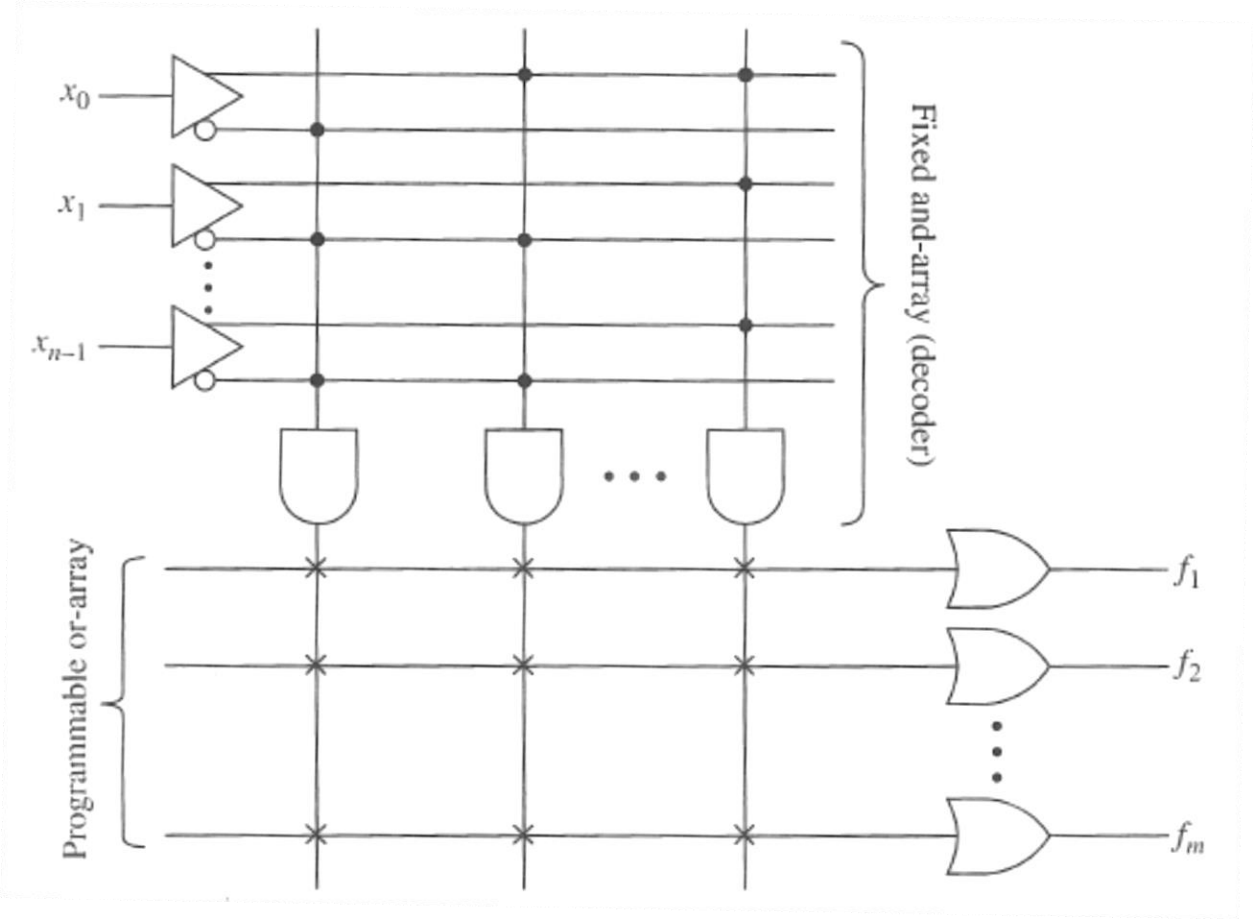
Programmable Read-Only Memory (PROM)

- AND-array with buffer/inverter is an n-to-2^n-line decoder.
- OR-array is a collection of programmable or-gates.
- Decoder is a min-term generator.
- n-variable minterms appear on the 2^n lines at the decoder output. These are also known as word lines.
- n input lines called address lines, m output lines called bit lines.
- $2^n \times m$ PROM.
- Realization of Boolean expressions same as realization using decoder discussed previously.

PROM Structure

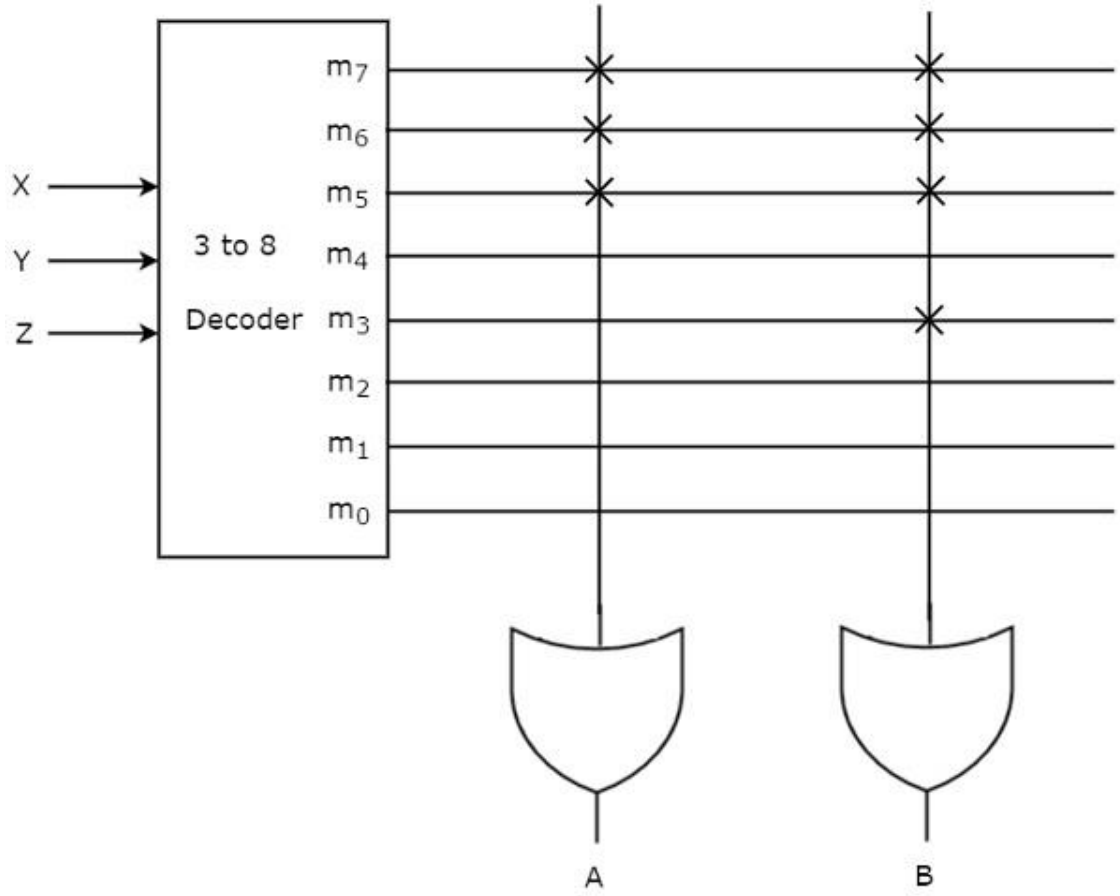


PROM Structure



PLD Notation

PROM example: 3-8 Decoder A, B obtained from fixed AND array



NOT, OR, NAND & XOR using PROM device?

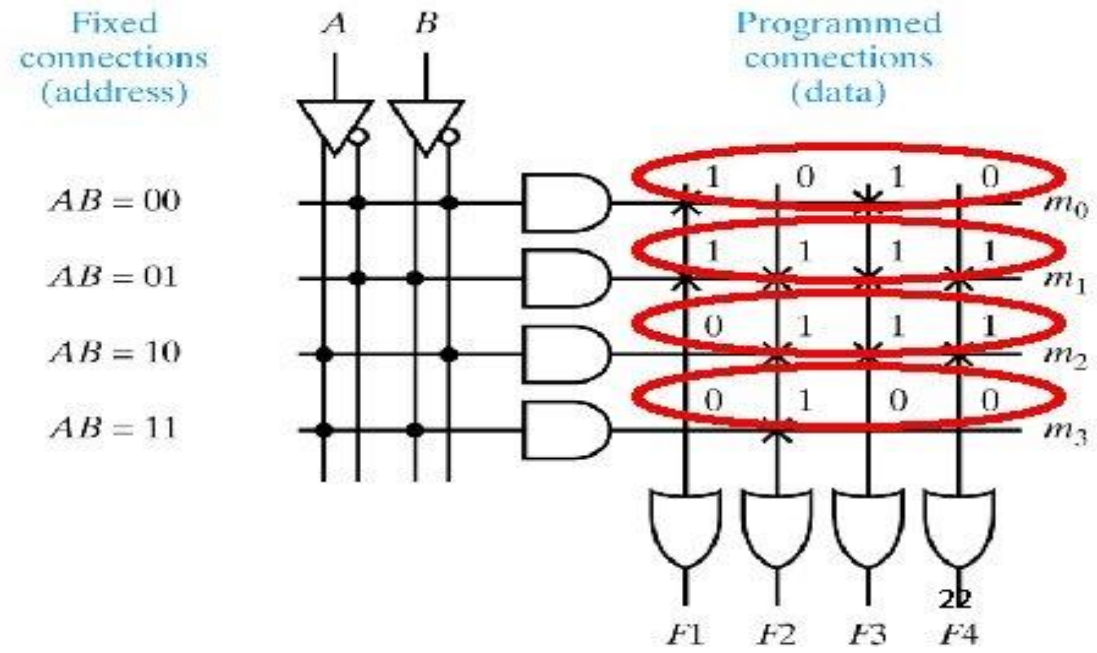
PROM Implementation

NOT, OR, NAND, XOR

A	B	F1	F2	F3	F4
0	0	1	0	1	0
0	1	1	1	1	1
1	0	0	1	1	1
1	1	0	1	0	0

Fuse map

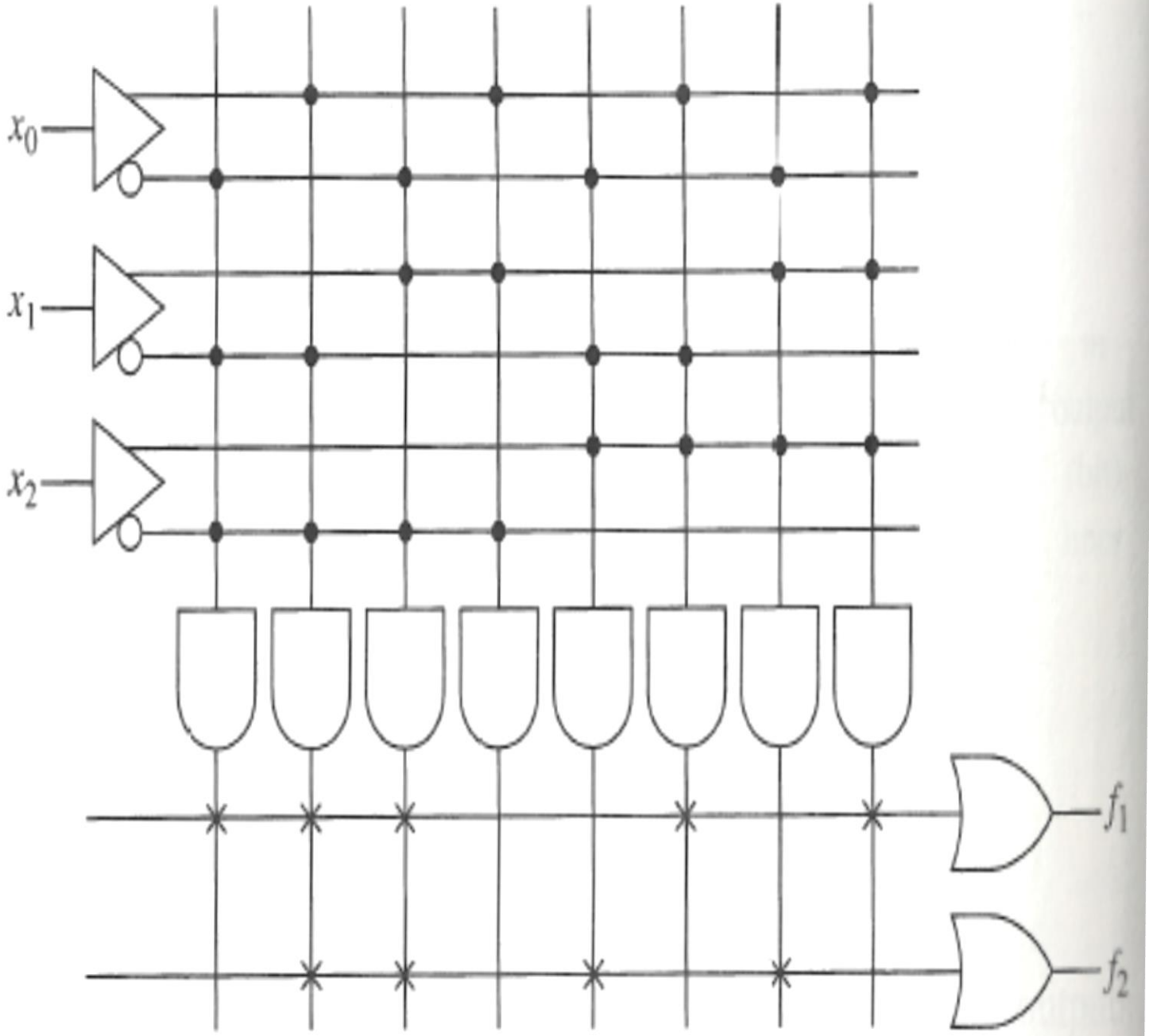
Address	Data
0	A
1	F
2	7
3	4



Realizing $f_1(x_2, x_1, x_0) = \sum m(0,1,2,5,7)$

$f_2(x_2, x_1, x_0) = \sum m(1,2,4,6)$

x_2	x_1	x_0	f_1	f_2
0	0	0	1	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	0
1	0	0	0	1
1	0	1	1	0
1	1	0	0	1
1	1	1	1	0

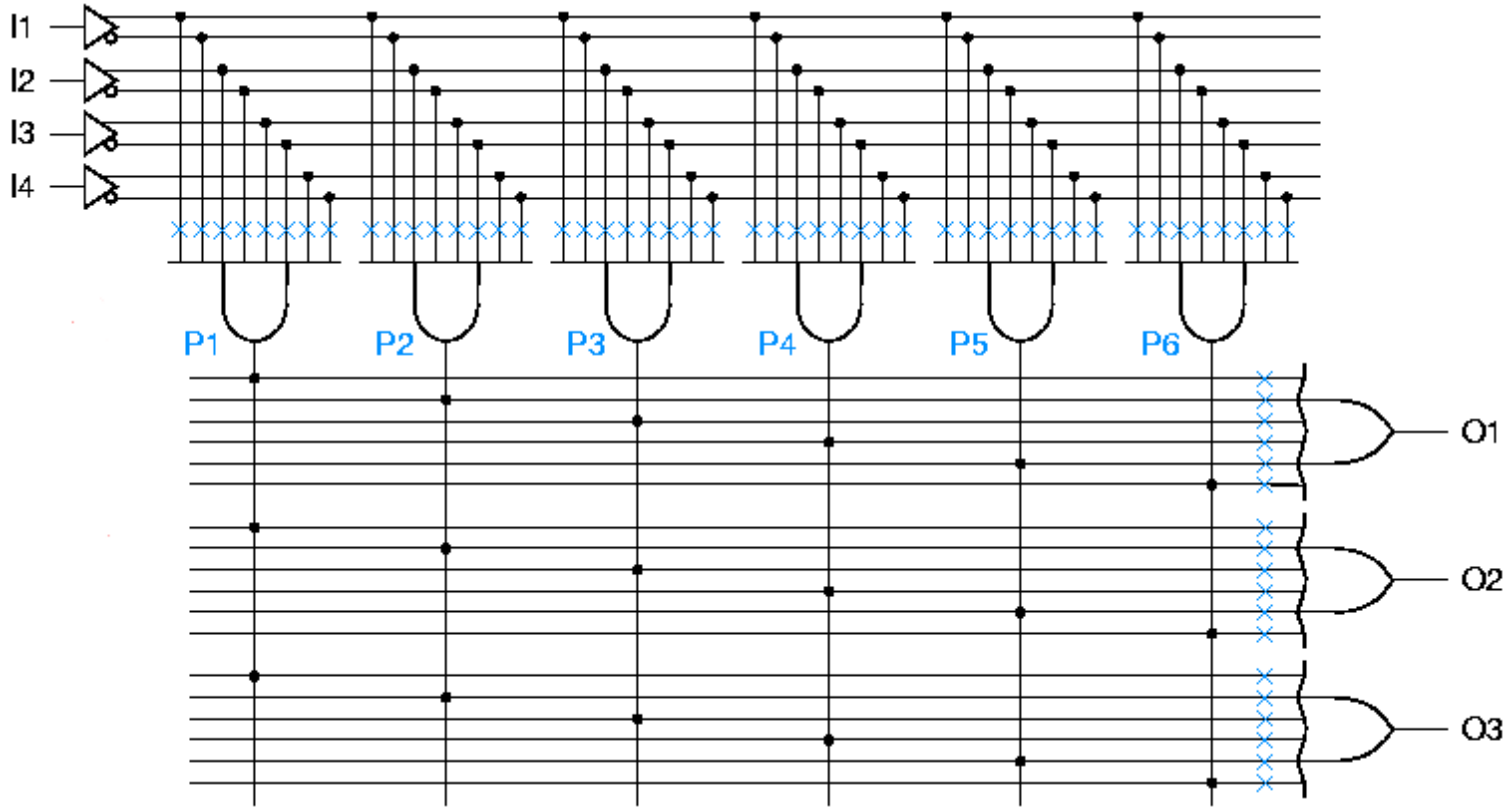


- The first PLDs were Programmable Logic Arrays (**PLAs**).
- A PLA is a combinational, 2-level AND-OR device that can be programmed to realise any sum-of-products logic expression.
- A PLA is limited by:
  - the number of inputs (n)
  - the number of outputs (m)
  - the number of product terms (p)
- We refer to an “n x m PLA with p product terms”. Usually, $p << 2^n$.
- An n x m PLA with p product terms contains p 2n-input AND gates and m p-input OR gates.

PLA

- **Each input is connected to a buffer that produces a true and a complemented version of the signal.**

A 4x3 PLA with 6 product terms.

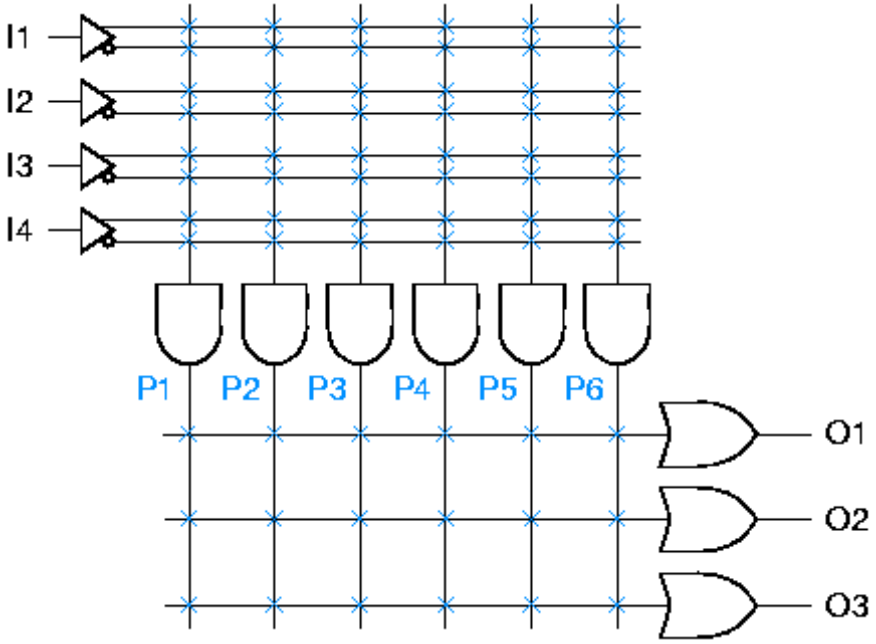


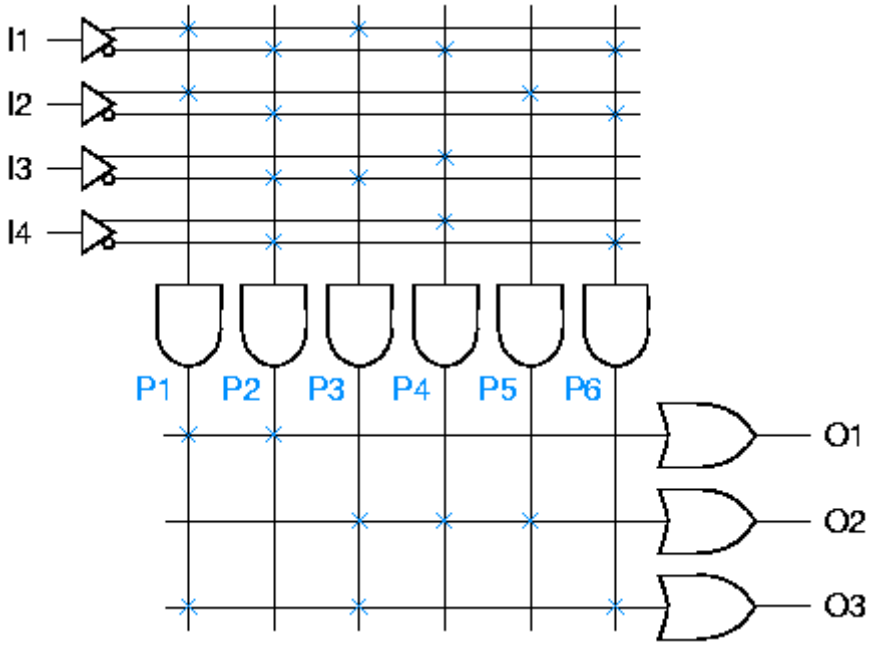
- Potential connections are indicated by Xs.
- The device is programmed by establishing the needed connections.
- The connections are made by fuses.

PLA

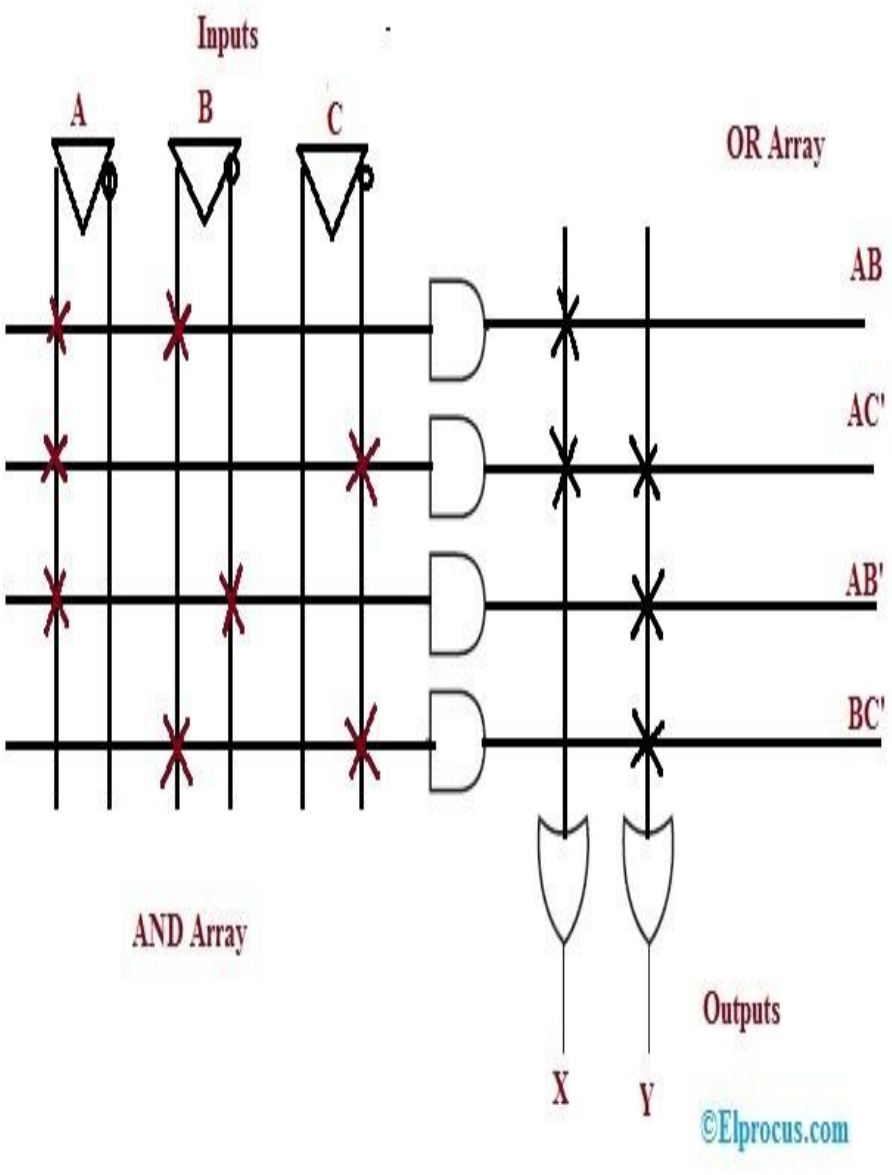
- Compact representation of the 4x3 PLA with 6 product terms.

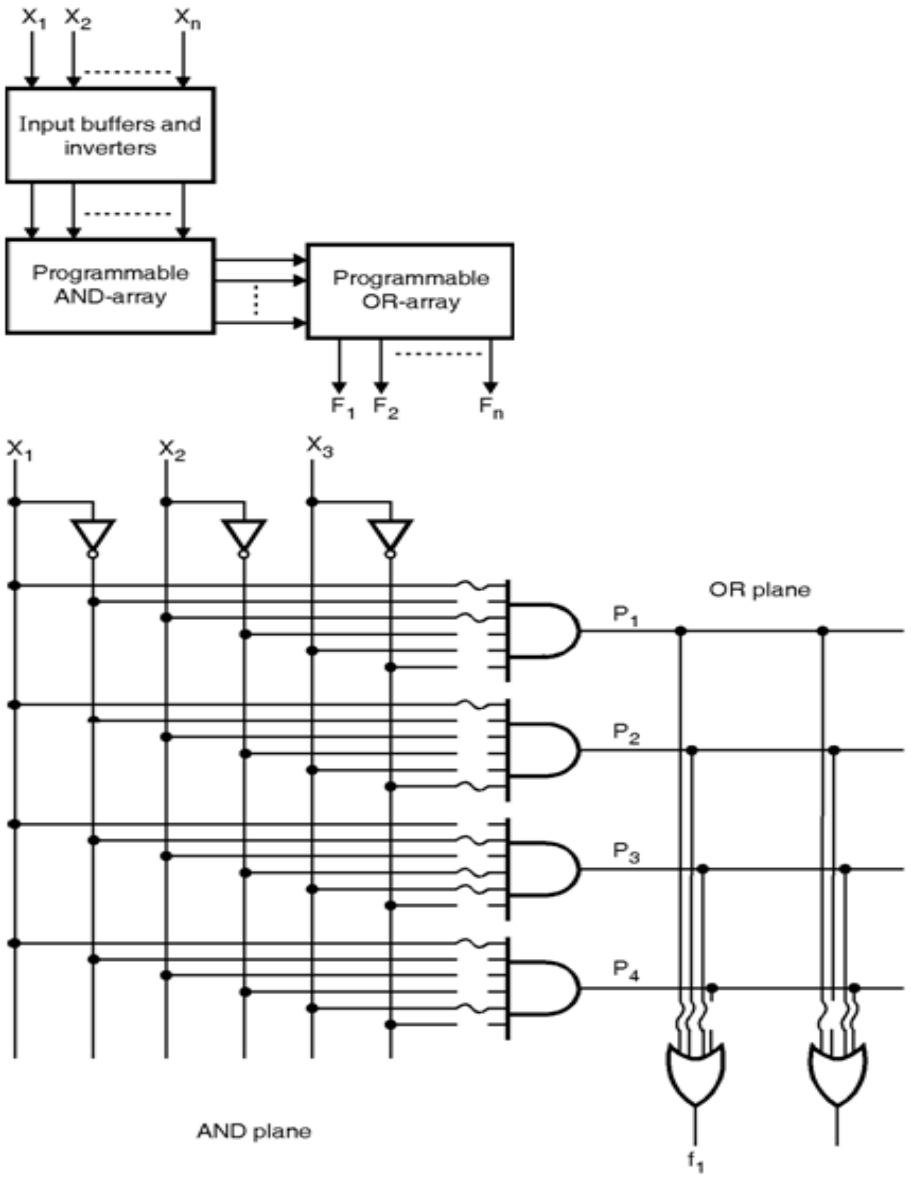
- O1 = I1·I2 + I1'·I2'·I3'·I4'
  O2 = I1·I3' + I1'·I3·I4 + I2
  O3 = I1·I2 + I1·I3' + I1'·I2'·I4'





PLA

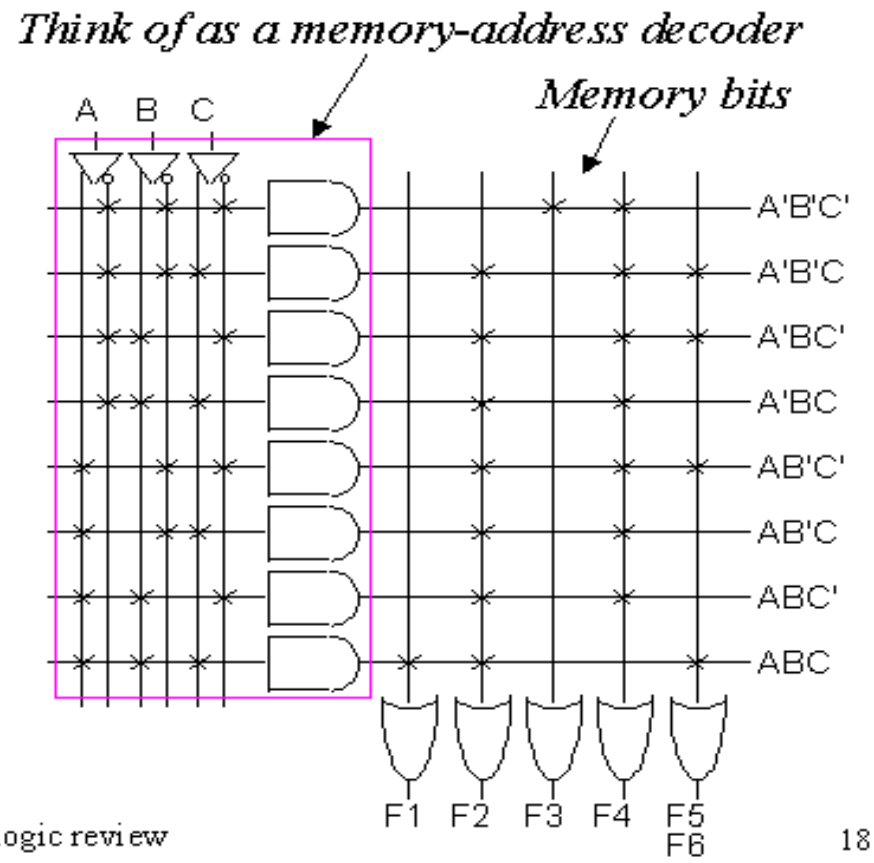

Inputs
A
B
C
OR Array
AB
AC'
AB'
BC'
AND Array
Outputs
X
Y
©Elprocus.com


X1 X2 Xn
Input buffers and inverters
Programmable AND-array
Programmable OR-array
F1 F2 Fn
X1 X2 X3
OR plane
P1
P2
P3
P4
AND plane
f1

PLA example

F1 = ABC
F2 = A + B + C
F3 = A' B' C'
F4 = A' + B' + C'
F5 = A xor B xor C
F6 = A xnor B xnor C

A	B	C	F1	F2	F3	F4	F5	F6
0	0	0	0	0	1	1	0	0
0	0	1	0	1	0	1	1	1
0	1	0	0	1	0	1	1	1
0	1	1	0	1	0	1	0	0
1	0	0	0	1	0	1	1	1
1	0	1	0	1	0	1	0	0
1	1	0	0	1	0	1	0	0
1	1	1	1	1	0	0	1	1



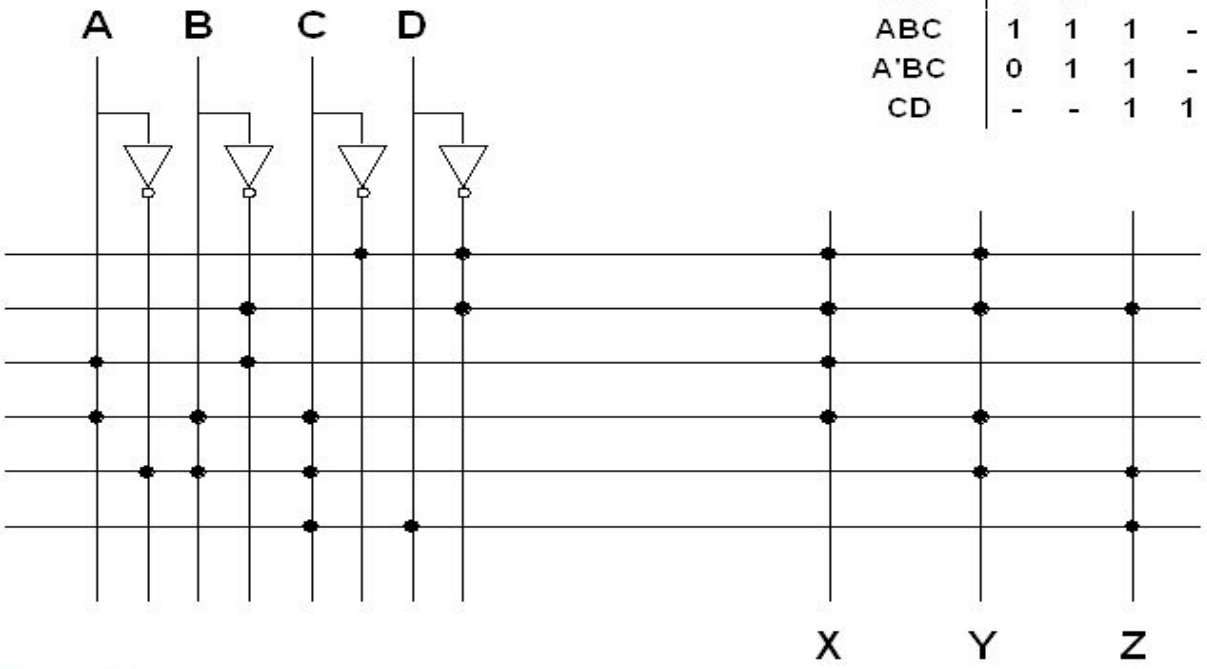
PLA Example

X = C'D' + B'D' + AB' + ABC

Y = C'D' + B'D' + ABC + A'BC

Z = B'D' + CD + A'BC

Product Term	Input A	B	C	D	Output X	Y	Z
C'D'	-	-	0	0	1	1	0
B'D'	-	0	-	0	1	1	1
AB'	1	0	-	-	1	0	0
ABC	1	1	1	-	1	1	0
A'BC	0	1	1	-	0	1	1
CD	-	-	1	1	0	0	1



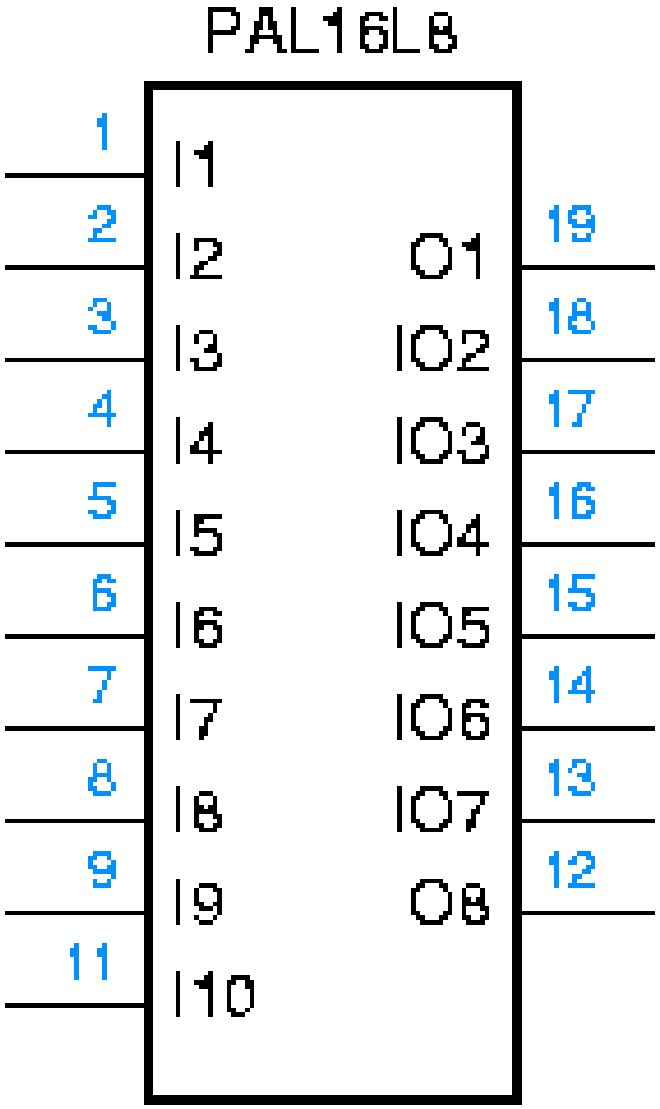
BYU
Computer Engineering
Electrical Engineering

ECEn 224

PLA
Page 31

PAL

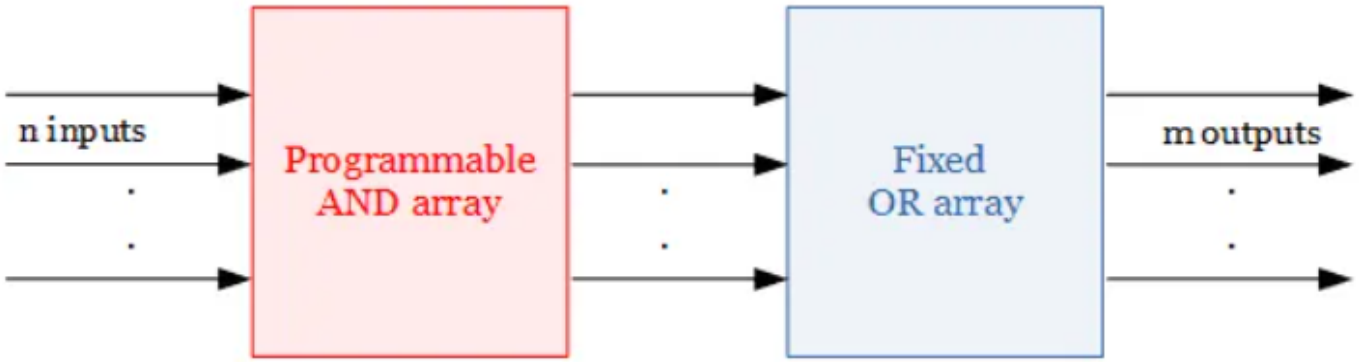
- Another PLD is PAL (Programmable Array Logic).
- A PAL device has a fixed OR array.
- In a PAL, product terms are not shared by the outputs.
- A PAL is usually faster than a similar PLA.



PAL

- Programmable Array Logic (PAL) is **a type of semiconductor used to implement logic functions in digital circuits**. PAL is a type of programmable logic device, which is a term for an integrated circuit that can be programmed in a laboratory to perform complex functions.
- PAL is **a programmable logic device that has Programmable AND array & fixed OR array**. The advantage of PAL is that we can generate only the required product terms of Boolean function instead of generating all the min terms by using programmable AND gates. The block diagram of PAL is shown in the following figure.

PAL

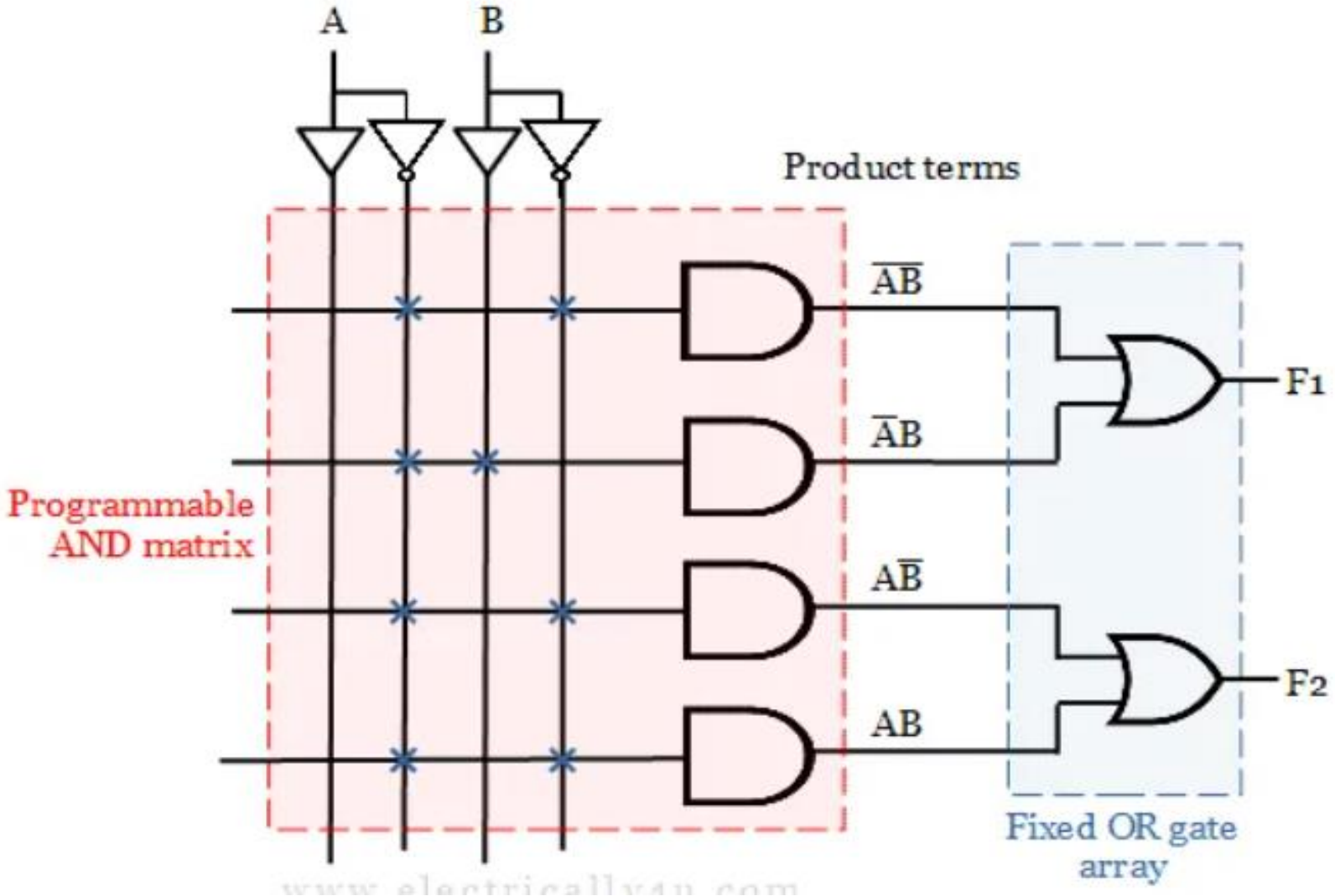


The following figure shows the internal structure of Programmable Array Logic. The product terms can be programmed through the fuse link.

It means the user can decide the connection between the inputs and the AND gates. If a particular input line is to be connected to the AND gate, then the fuse link must be placed at the interconnection.

The AND gate outputs are then fed as an input to the fixed OR gate. Depending upon the required function, the output line of the AND gate is connected to the corresponding input of the OR gate.

PAL structure



Implement Boolean Functions F, F2 using PAL?

$$F1 = \bar{A}B\bar{C}D + \bar{A}BC\bar{D} + AB\bar{C}D$$

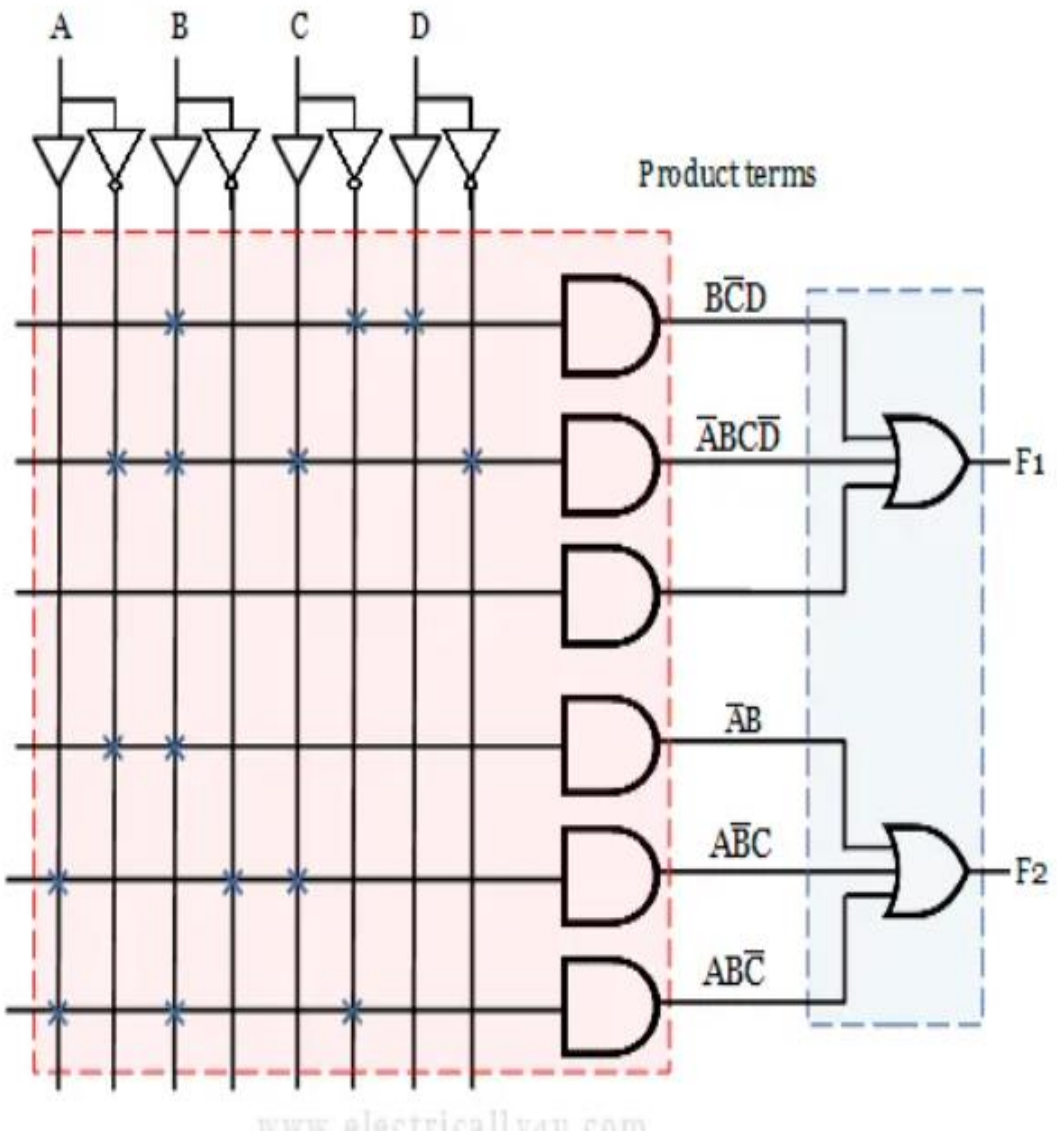
$$F1 = (\bar{A} + A)\,B\bar{C}D + \bar{A}BC\bar{D}$$

$$F1 = B\bar{C}D + \bar{A}BC\bar{D}$$

$$F2 = \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}C + AB\bar{C}$$

$$F2 = \bar{A}B\,(\bar{C} + C) + A\bar{B}C + AB\bar{C}$$

$$F2 = \bar{A}B + A\bar{B}C + AB\bar{C}$$



Realize BCD- Gray code converter using PAL?

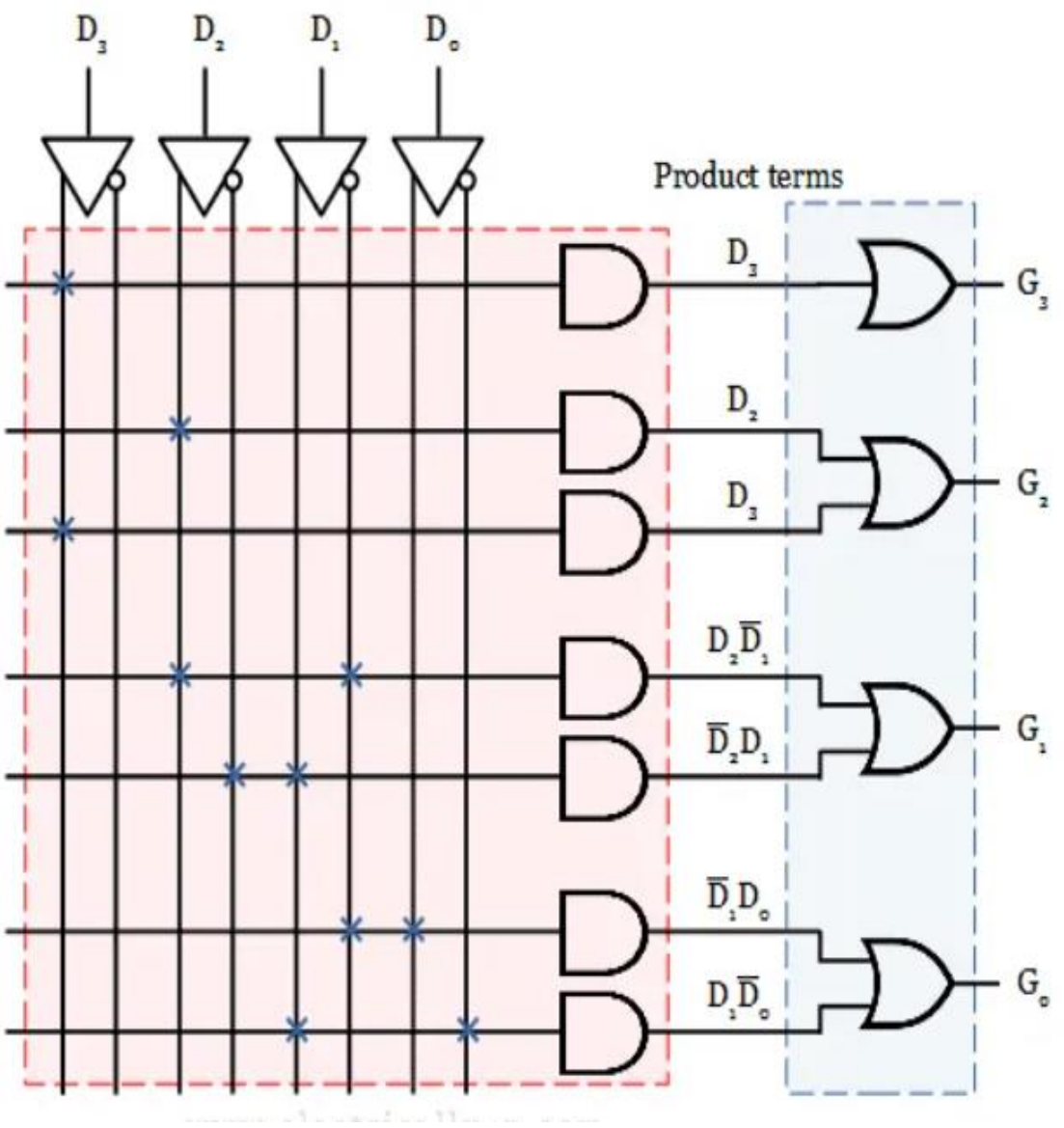
- Use K-maps to solve for expressions and as follows:

$$G_3 = D_3$$

$$G_2 = D_2 + D_3$$

$$G_1 = D_2\overline{D}_1 + \overline{D}_2 D_1$$

$$G_0 = \overline{D}_1 D_0 + D_1\overline{D}_0$$



- Kindly Refer suggested DSD course Text Books and other online like NPTEL resources for more design practice with full focus…

- Happy & Safe Learning…
- Pasulurri Sweta
- Binduswetha.ece@gmail.com

Thanking You