UNIVERSITI TEKNOLOGI MALAYSIA

www.utm.my

SCSR1013 DIGITAL LOGIC

MODULE 6:
FUNCTIONS OF COMBINATIONAL LOGIC

FACULTY OF COMPUTING

FUNCTIONS OF COMBINATIONAL LOGIC

- Basic Adders
- Parallel Binary Adder
- Comparators
- Decoders
- Encoders
- Multiplexers (Data Selector)
- DeMultiplexers (Data Distributor)
- Code Converter
- Parity Generators / Checkers

Basic Adders

Parallel Binary Adders

Comparators

Basic Adder

- Basic adder operation is fundamental to the study of digital system.

- Types of basic adder :
  - a) Half Adder
  - b) Full Adder

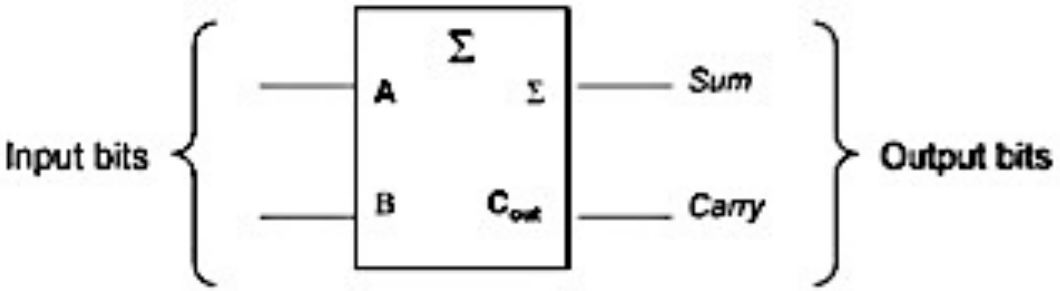
- Basic rules for binary addition

0+0= 0
0+1= 1
1+0= 1
1+1=10

1+ 1+1 = 11

Basic Adder: Half Adder

- Accepts two binary digits on its inputs and produces two binary digits on its outputs, a sum bit and a carry bit
- It meant for adding 1-bit number

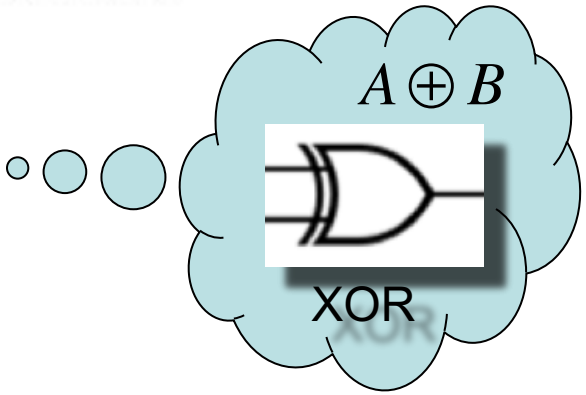


$$Sum = A \oplus B$$

- Sum output is 1, if A and B are not equal.

$$C_{out} = AB$$

- Output carry is 1 only when A and B are 1s.



(Module v3: Page 97)

The Design:

AB $A \oplus B$

A	B	Carry, C_{out}	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

X-OR truth table

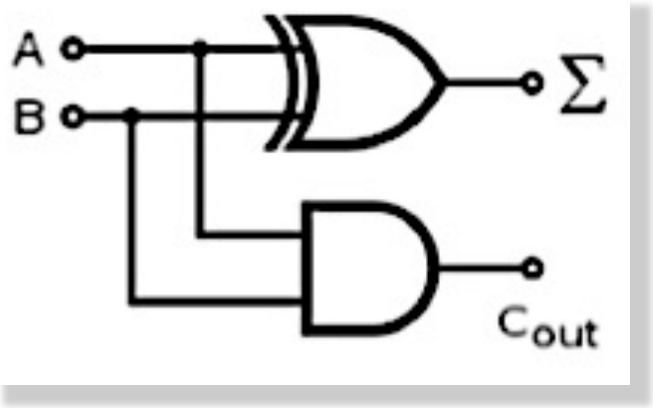
(Module v3: Page 97)

From the truth table

$$\text{Sum}, \Sigma = \overline{A}B + A\overline{B} = A \oplus B$$

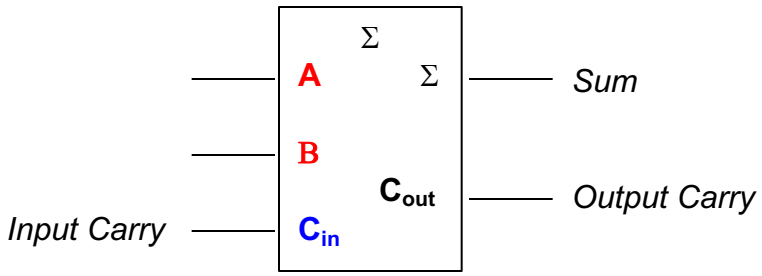
and Carry or C_{out}

$$\text{Carry}, C_{out} = AB$$

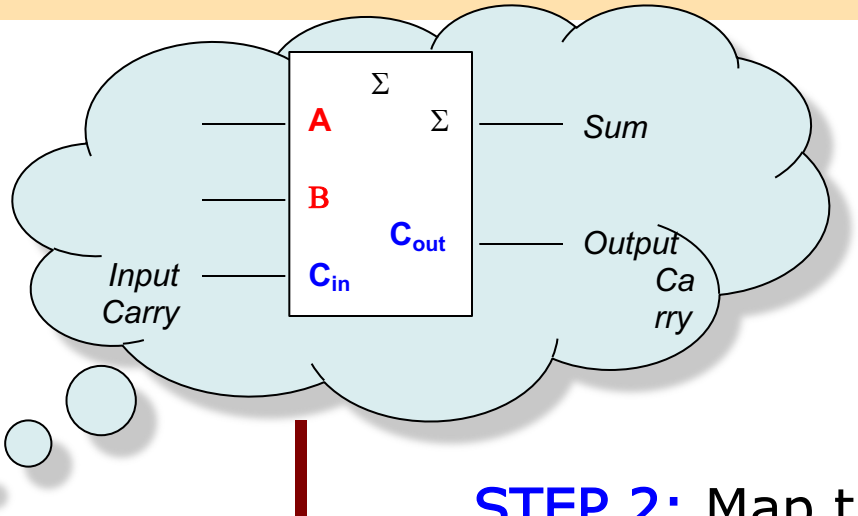


Basic Adder: Full Adder

- Accepts two input bits and an input carry and generates a sum output and an output carry.
- Two or more full adders are connected to form parallel adders



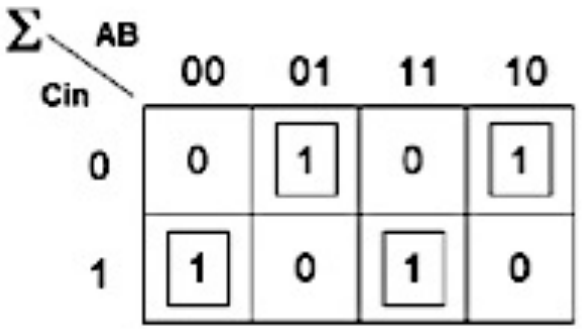
The design:



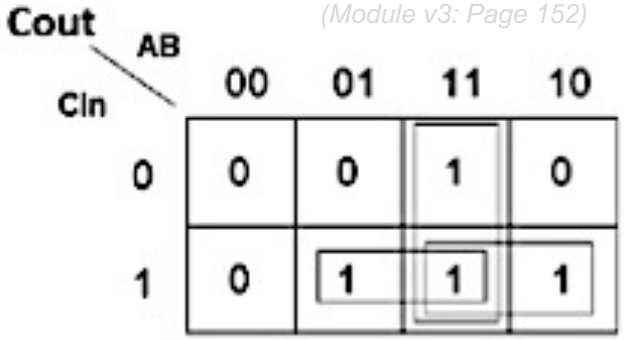
STEP 1: Produce a truth table

Inputs			Outputs	
A	B	CIN	∑	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

STEP 2: Map the K-Map



Checker Board
(Module v3: Page 152)



STEP 3: Simplify K-Map & get the equation

(Module v3: Page 97-98)

$$\sum = \overline{A}B\overline{C}_{in} + A\overline{B}\overline{C}_{in} + \overline{A}\overline{B}C_{in} + ABC_{in}$$

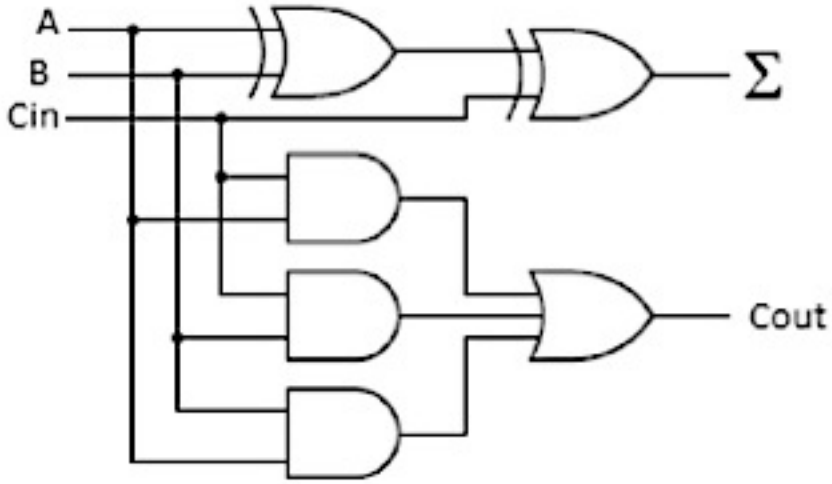
$$= (\overline{A}B + A\overline{B})\overline{C}_{in} + (\overline{A}\overline{B} + AB)C_{in}$$

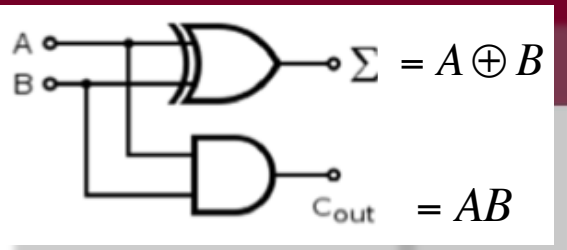
$$= (A \oplus B)\overline{C}_{in} + (\overline{A \oplus B})C_{in}$$

$$\sum = (A \oplus B) \oplus C_{in}$$

$$C_{out} = AB + AC_{in} + BC_{in}$$

STEP 4: Draw the circuit

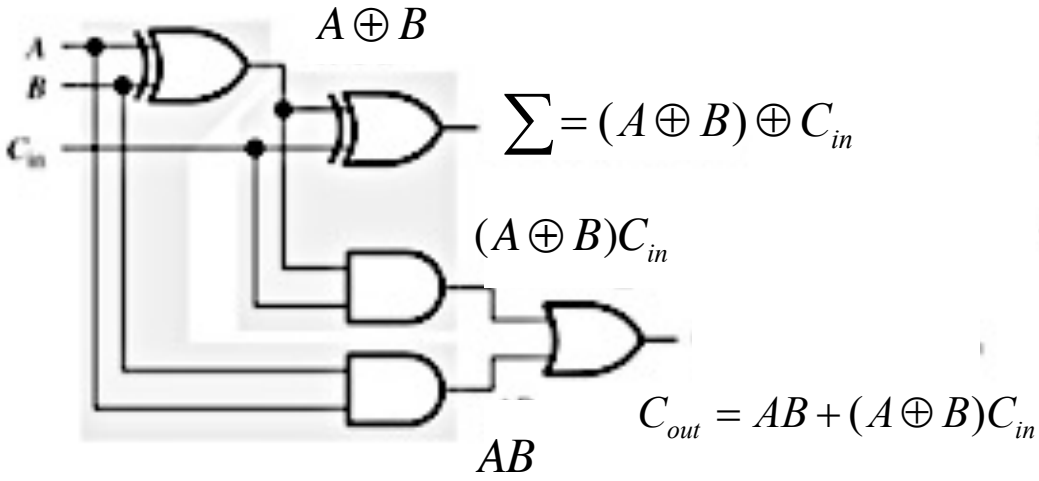


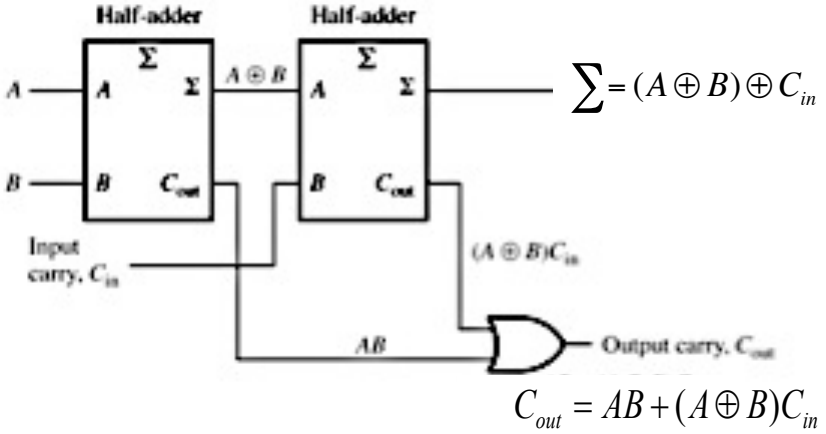


Half Adder

..Full Adder

- Full Adder actually can be built by using 2 Half Adder
  - Half Adder is the one shaded

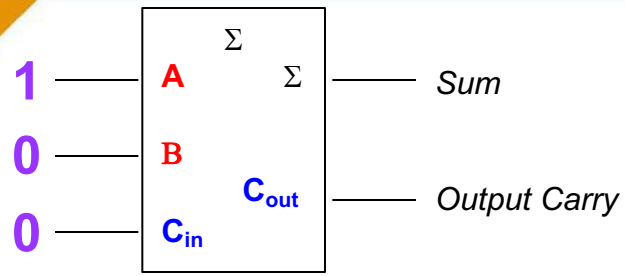


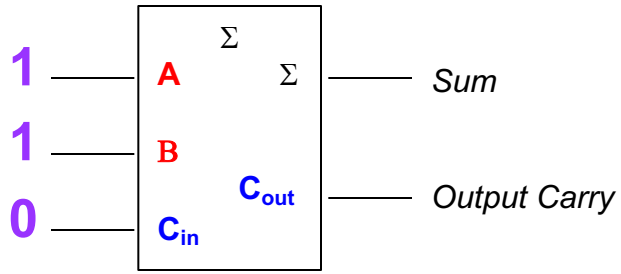


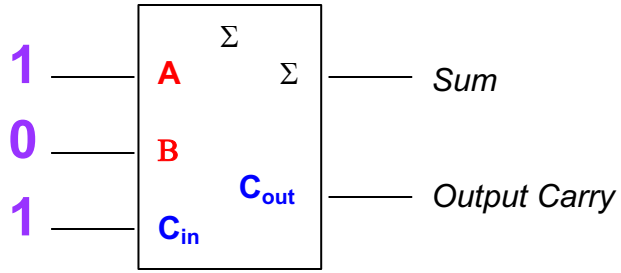
(Half Adder with 2 half adder)

Example:

For each of the three full adders, determine the outputs for the inputs shown.



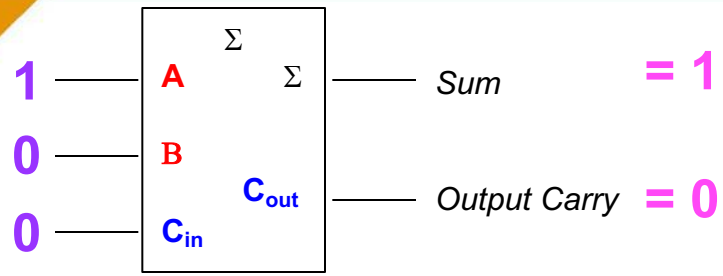




Solution:

$$\sum = (A \oplus B) \oplus C_{in}$$

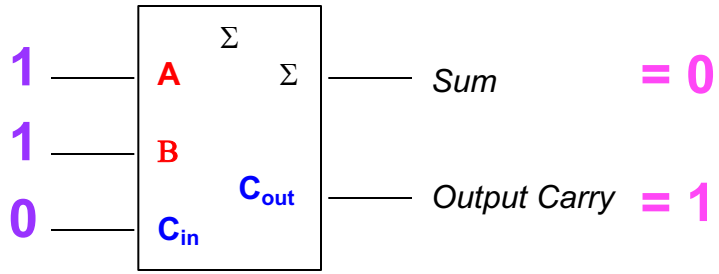
$$C_{out} = AB + (A + B)C_{in}$$



A = 1, B = 0, and C_{in} = 0

1 + 0 + 0 = 1 with no carry

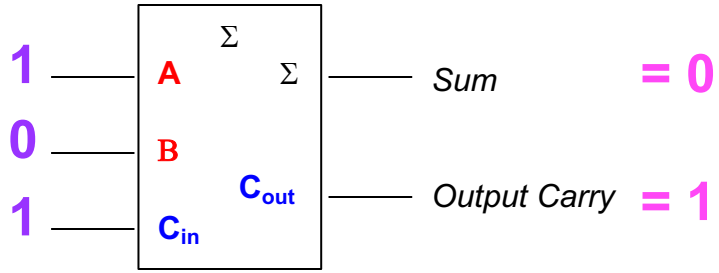
$\sum$ = 1, C_{out} = 0



A = 1, B = 1, and C_{in} = 0

1 + 1 + 0 = 0 with carry 1

$\sum$ = 0, C_{out} = 1



A = 1, B = 0, and C_{in} = 1

1 + 0 + 1 = 0 with carry 1

$\sum$ = 0, C_{out} = 1

Extra

Exercise 6.1: Determine the sum ($\sum$) and the output carry (C_{out}) of a half adder for each set of input bits:

Solution:

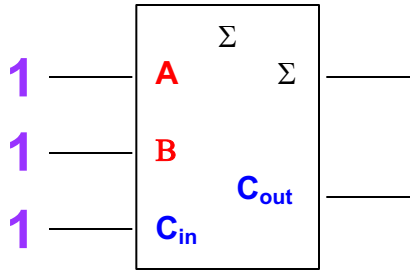
$$\sum = A \oplus B \qquad C_{out} = AB$$

	Input, A	Input, B	Sum ($\sum$)	Output carry (C_{out})
i)	0	1	1	0
ii)	0	0	0	0
iii)	1	0	1	0
iv)	1	1	0	1

Extra

Exercise 6.2: A full adder has $C_{in}=1$. What are the sum ($\sum$) and the output carry (C_{out}) when A=1 and B=1?

Solution 6.2:



$$\sum = (A \oplus B) \oplus C_{in} \quad = 1$$

$$C_{out} = AB + (A + B)C_{in} \quad = 1$$

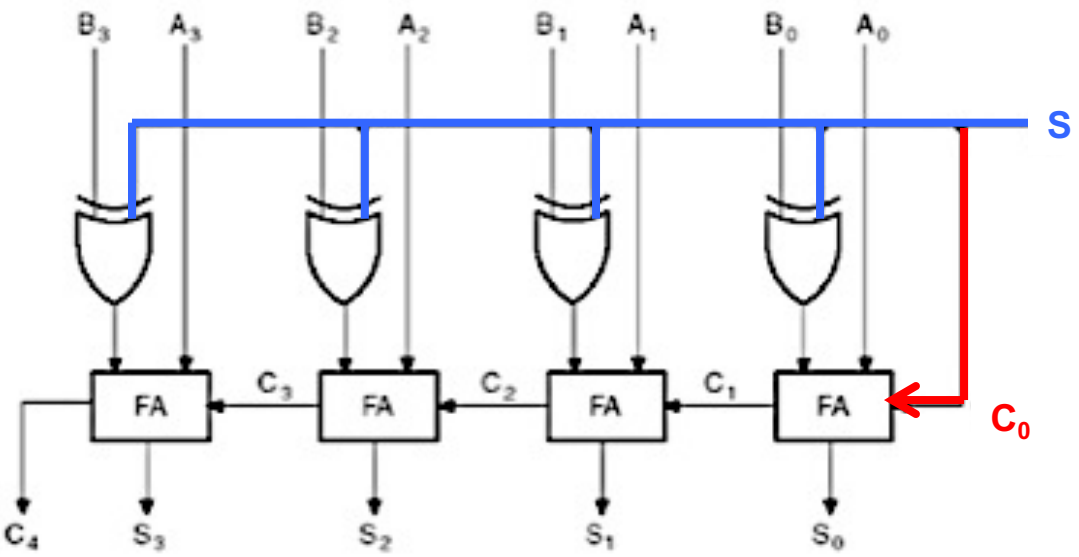
A = 1, B = 1, and C_{in} = 1

1 + 1 + 1 = 1 with carry 1

$\sum$ = 1, C_{out} = 1

Adder/Subtractor Circuit

- The circuit implement the 2's complement arithmetic
- If S=0 implement **A + B**, no changes to B because B⊕S = B⊕0 = B, therefore full adder will do A+B
- If S =1 implement **A - B**, -B will be the 2's complement of B because B⊕1 = $\overline{B}$ which is the 1's complement of B. To get the 2's complement of B we have to add 1 by connecting C_0 to S so that C_0 = S = 1, therefore full adder will do ***A + 2's complement of B*** which is **A-B**



Note:
FA □ Full Adder;
C □ Carry

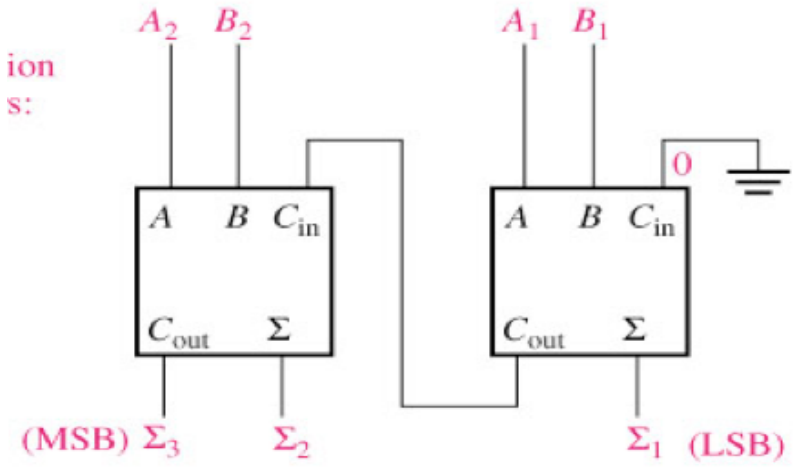
Input: A_i and B_i
Output: S_i

Parallel Adder

- Two or more full adders are connected to form parallel adders

General format, addition of two 2-bit numbers:

$$\begin{array}{r} A_2A_1 \\ +\ B_2B_1 \\ \hline \Sigma_3\Sigma_2\Sigma_1 \end{array}$$

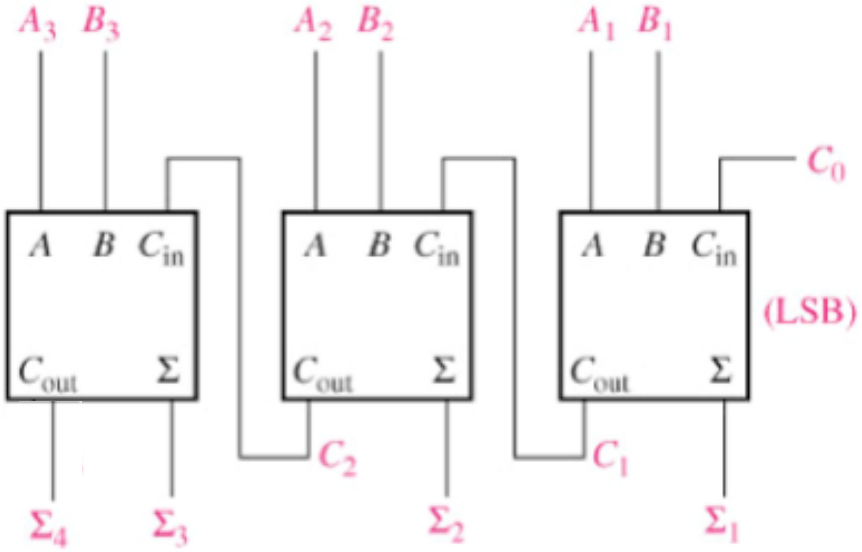


$$\begin{array}{rcccc} & A_3 & A_2 & A_1 \\ + & B_3 & B_2 & B_1 \\ \hline & \Sigma_4 & \Sigma_3 & \Sigma_2 & \Sigma_1 \end{array}$$

Example: Determine the sum generated by the 3-bit parallel adder below and show the intermediate carries when the binary numbers 101 and 011 are being added.

A_3	A_2	A_1
1	0	1

B_3	B_2	B_1
0	1	1



$$\begin{array}{cccc} & A_3 & A_2 & A_1 \\ + & B_3 & B_2 & B_1 \\ \hline & \Sigma_4 & \Sigma_3 & \Sigma_2 & \Sigma_1 \end{array}$$

Example: Determine the sum generated by the 3-bit parallel adder below and show the intermediate carries when the binary numbers 101 and 011 are being added.

Solution:

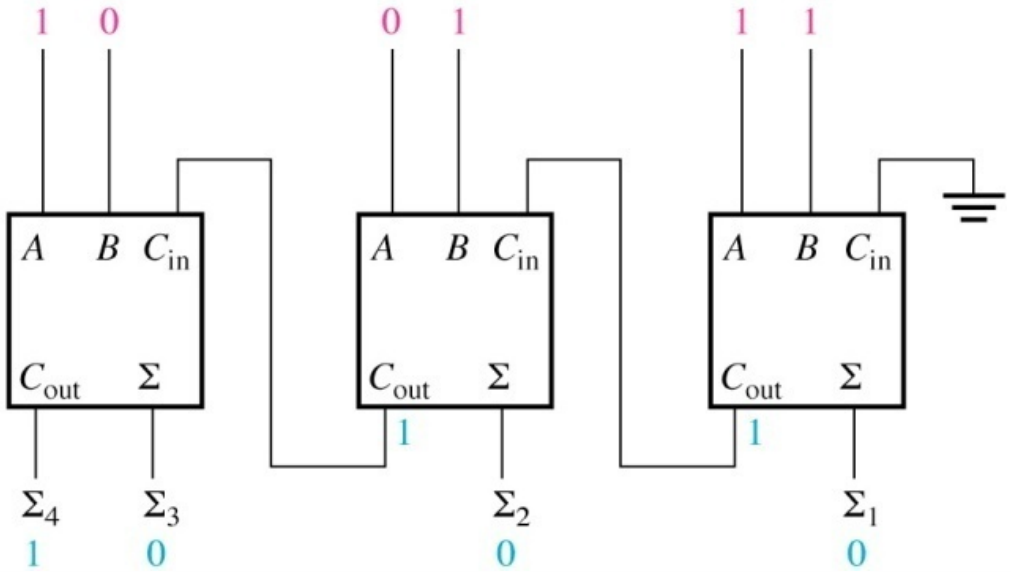
A_3	A_2	A_1
1	0	1

B_3	B_2	B_1
0	1	1

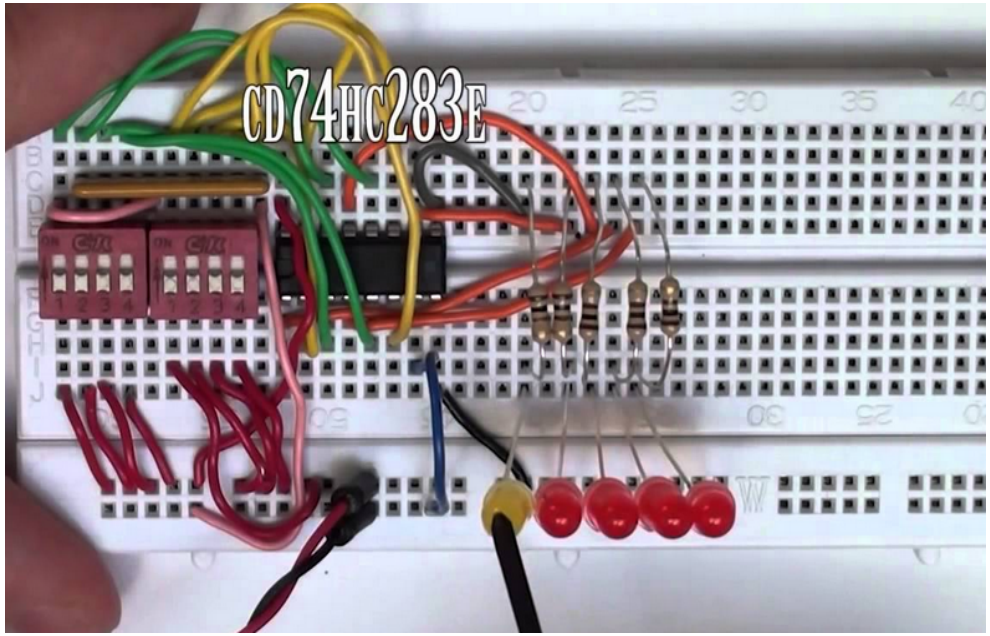
$$\begin{array}{cccc} & 1 & 0 & 1 \\ + & 0 & 1 & 1 \\ \hline 1 & 0 & 0 & 0 \end{array}$$

$$\sum = (A \oplus B) \oplus C_{in}$$

$$C_{out} = AB + (A + B)C_{in}$$



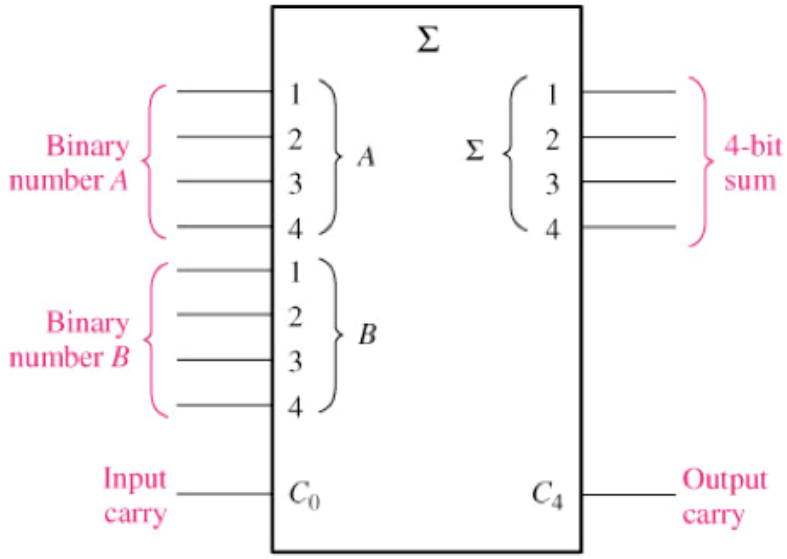
Extra



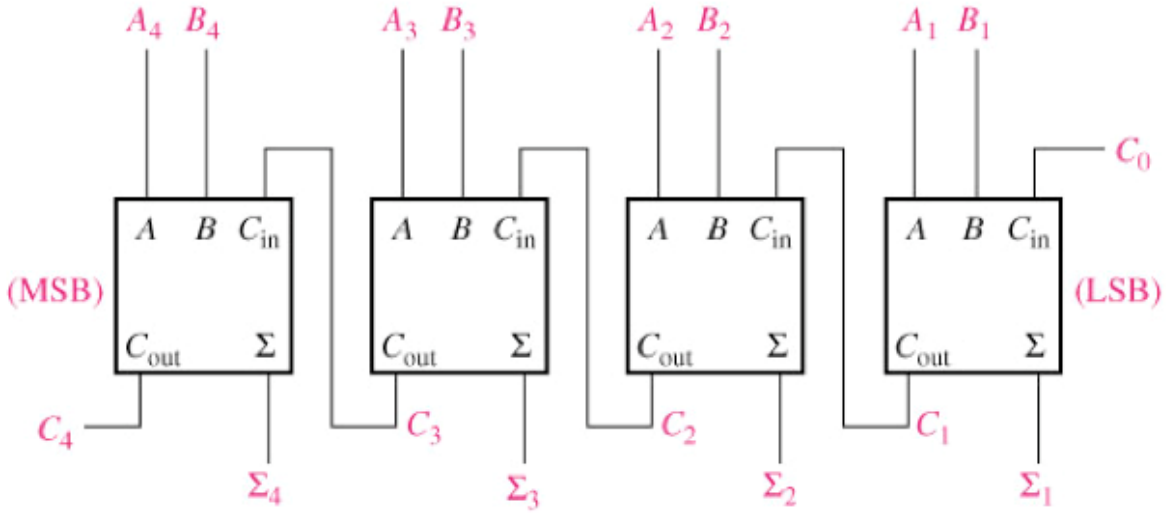
MSI chip: (74LS283)
4-bit parallel adder

MSI chip: (74LS283)

4-bit parallel adder



(b) Logic symbol



(a) Block diagram

$$\begin{array}{r} A_4\ A_3\ A_2\ A_1 \\ +\ B_4\ B_3\ B_2\ B_1 \\ \hline \Sigma_5\ \Sigma_4\ \Sigma_3\ \Sigma_2\ \Sigma_1 \end{array}$$

- Two type of parallel adder, based on how it handle a carry

(Module: Page 180)

1. Ripple Carry Adder
   - Carry output of each full-adder is connected to carry in of the next order stage of full adder
   - Carry accumulate from the lowest order bit to the highest order bit
   - Speed of addition limited by the time required for the carries to propagate (or ripple) through all stages
   - Circuit simpler than look-ahead carry adder

2. Look-Ahead Carry Adder
   - Speed up the addition process by eliminating ripple carry delay
   - It anticipates the output carry of each stage , and based on the inputs, produces the output carry by either **carry generation** or **carry propagation**

Comparator

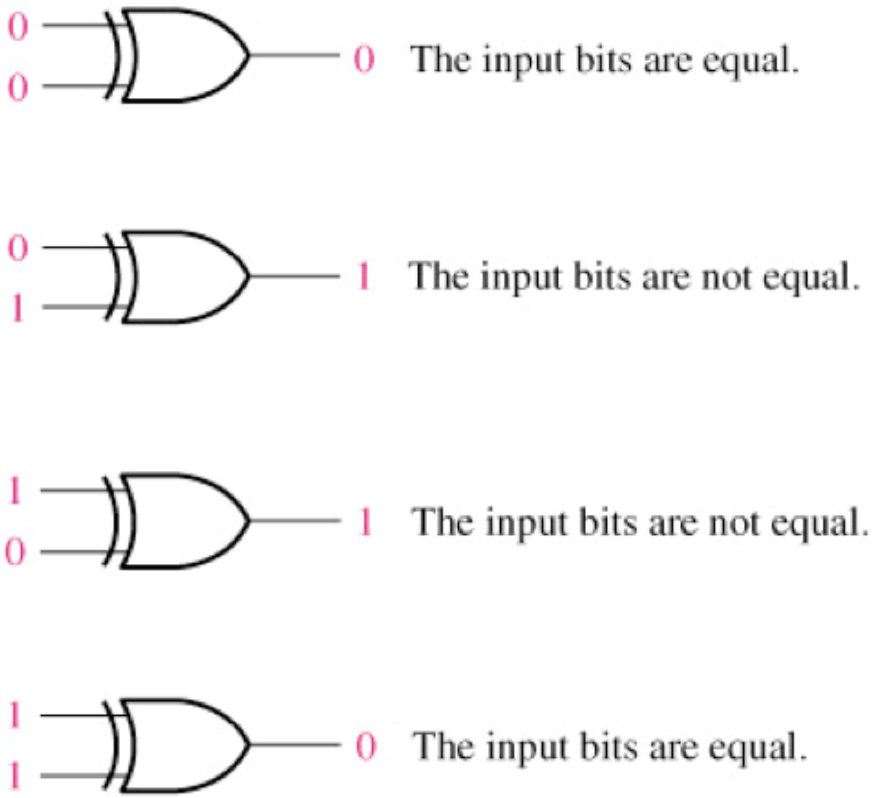
Compares two *n*-bit binary values to determine which one is larger.

- Inputs : 2 single-bit inputs (X and Y).
- Outputs : 3 lines: $X > Y$, $X = Y$, $X < Y$.

XNOR gate can be used as a basic comparator (Equality)

Use XOR to check for single bit Inequality.

XOR gate as a 2-bit comparator:



continue...

To check for N bit equality (XNOR), combine the result of comparing an N single bit comparator.

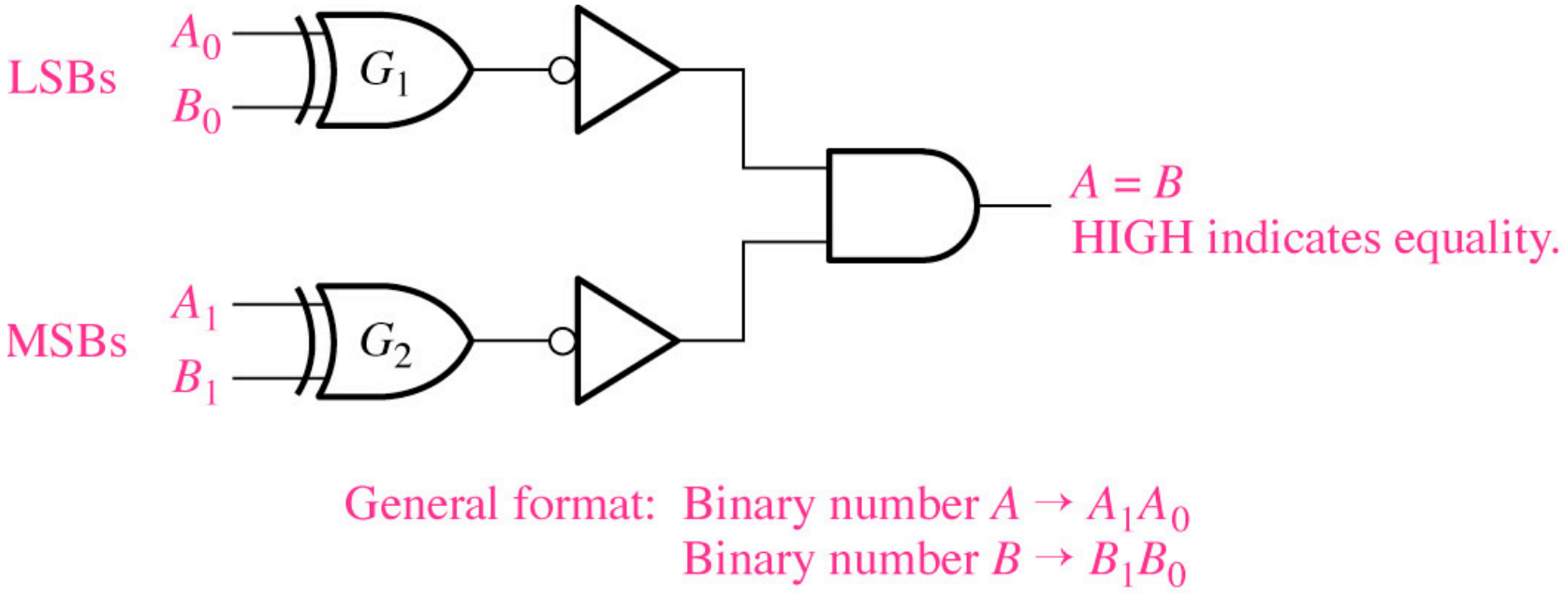
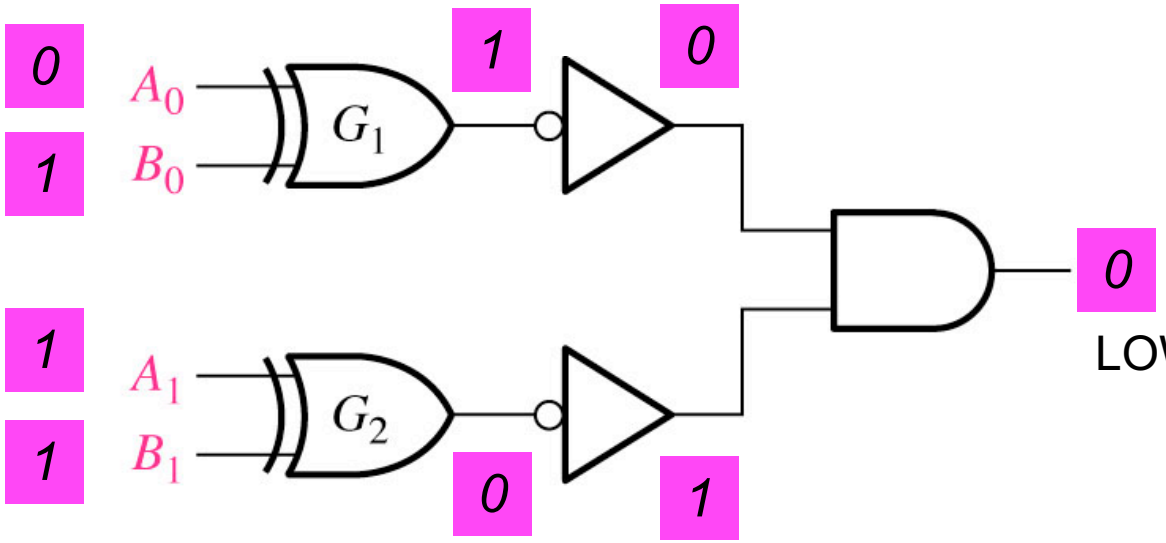


Figure: Logic diagram for equality comparison of two 2-bit numbers.

Extra

Example: Compare the equality for these 2-bits binary $A=10_2$ and $B=11_2$

General format: Binary number $A \rightarrow A_1A_0$
Binary number $B \rightarrow B_1B_0$



LOW: Indicate Not Equal

Example: Design a single comparator that compares X and Y. Determine whether X=Y, X<Y or X>Y.

Solution:

The truth table:

X_i	Y_i	$X > Y$	$X = Y$	$X < Y$
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

From the truth table, the output equations:

$$X > Y \rightarrow X\overline{Y}$$

$$X = Y \rightarrow \overline{X}\ \overline{Y} + XY = X \odot Y$$

$$X < Y \rightarrow \overline{X}Y$$

continue...

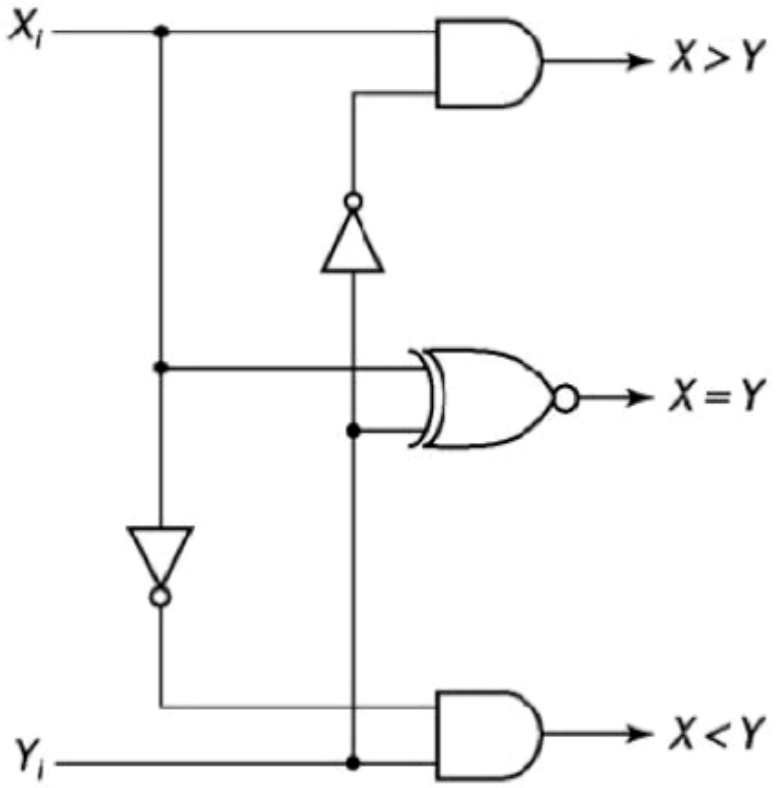
The circuit:

Design a separate circuit for each output and finally combine them.

$$X > Y \rightarrow X\overline{Y}$$

$$X = Y \rightarrow \overline{X}\ \overline{Y} + XY = X \odot Y$$

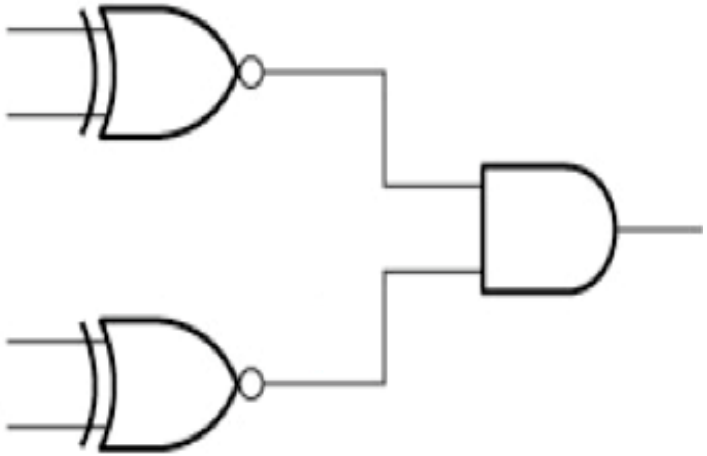
$$X < Y \rightarrow \overline{X}Y$$



Extra

Example: Apply each of the following sets of binary numbers to the comparator inputs in the Figure, and determine the output (equlity) by following the logic levels through the circuit.

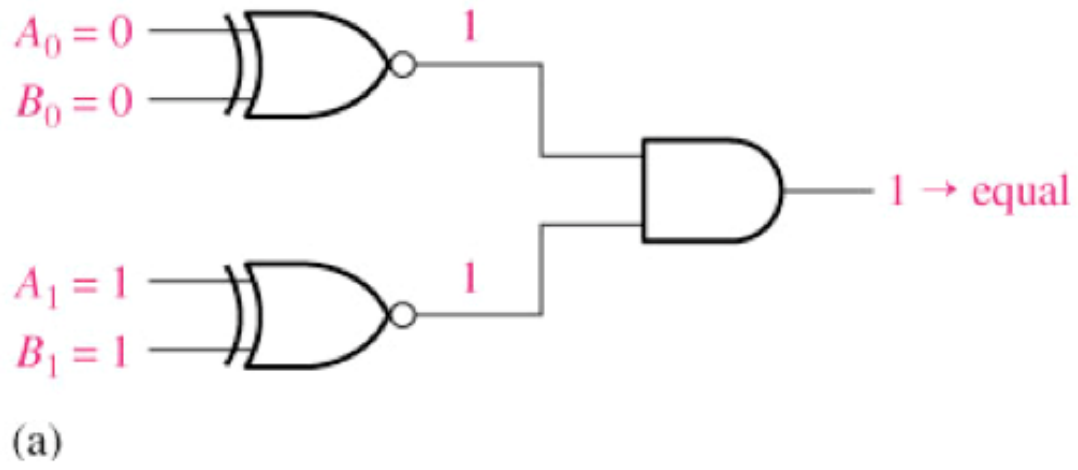
(a) 10 and 10 (b) 11 and 10



Extra

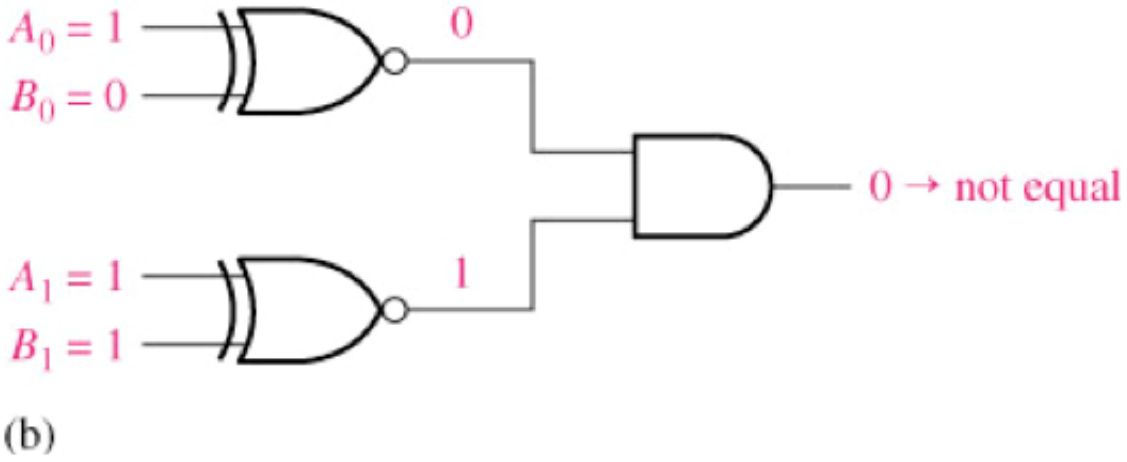
Solution:

(a) 10 and 10



(a)

(b) 11 and 10



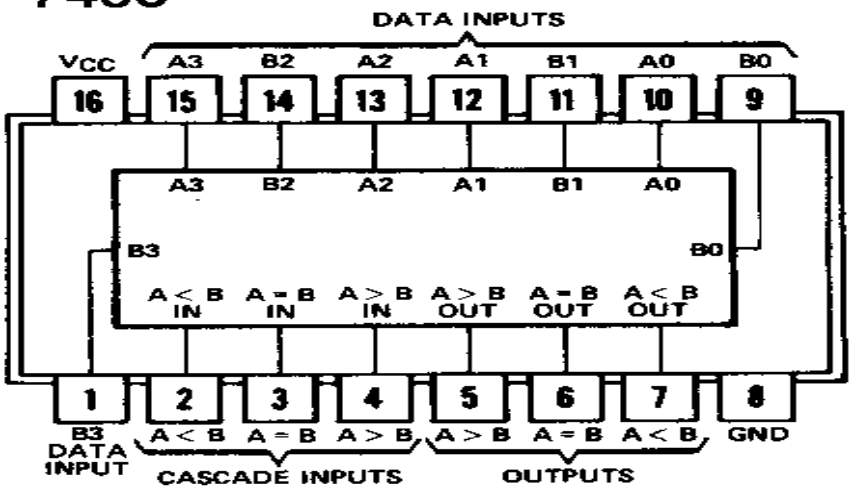
(b)

Extra

MSI chip: (74LS85)

4-bit Comparator

7485



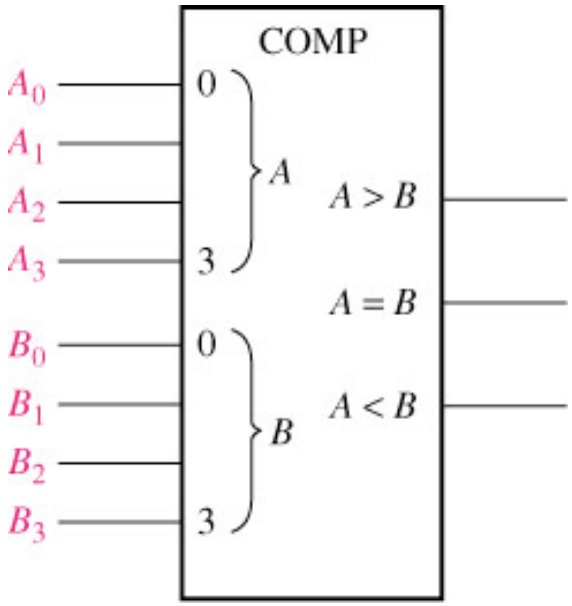


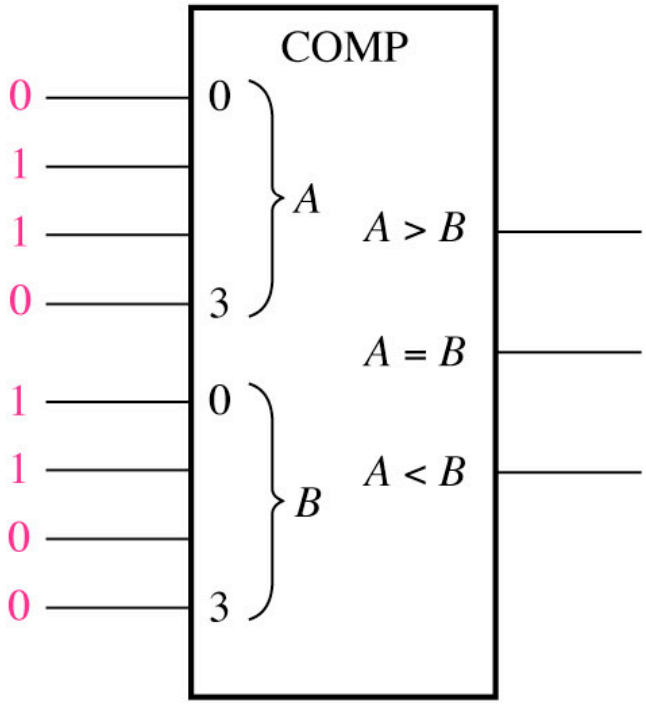
Figure: Logic symbol for a 4-bit comparator with inequality indication.

Its operation: compare the MSB first

- If A_3= 1 and B_3 = 0 then A > B set without comparing the lower bit
- If A_3= 0 and B_3 = 1 then A < B set without comparing the lower bit
- If A_3 = B_3 then A2 and B2 will be compared, until A_0 and B_0, if all bits are equal, only then A= B set

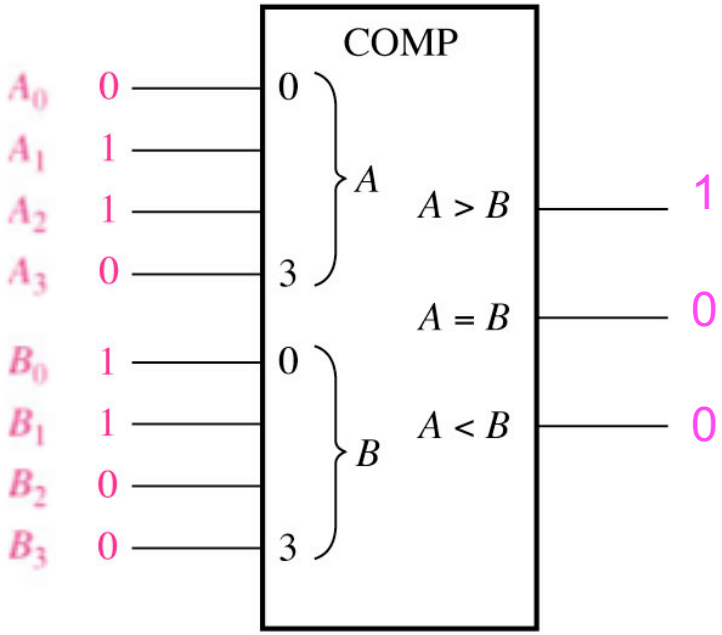
Extra

Example: Determine the A = B, A > B, and A < B outputs for the input numbers shown on the comparator in the Figure.



Extra

Solution:



A > B output is **HIGH** and the other outputs are **LOW**

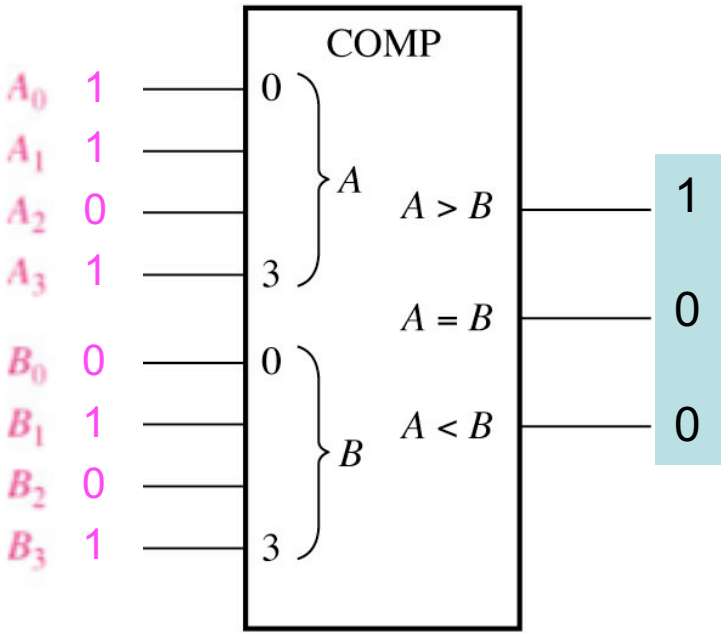
Step 1: Compare A_3 and B_3 □ Number A = B; compare next bits

Step 2: Compare A_2 and B_2 □ Number A > B; stop comparing.

Extra

Exercise 6.3: What are the comparator outputs when binary number of A = 1011 and B = 1010 applied as the inputs ?

Solution 6.3:



Step 1: Compare A_3 and B_3 □ Number A = B; compare next bits

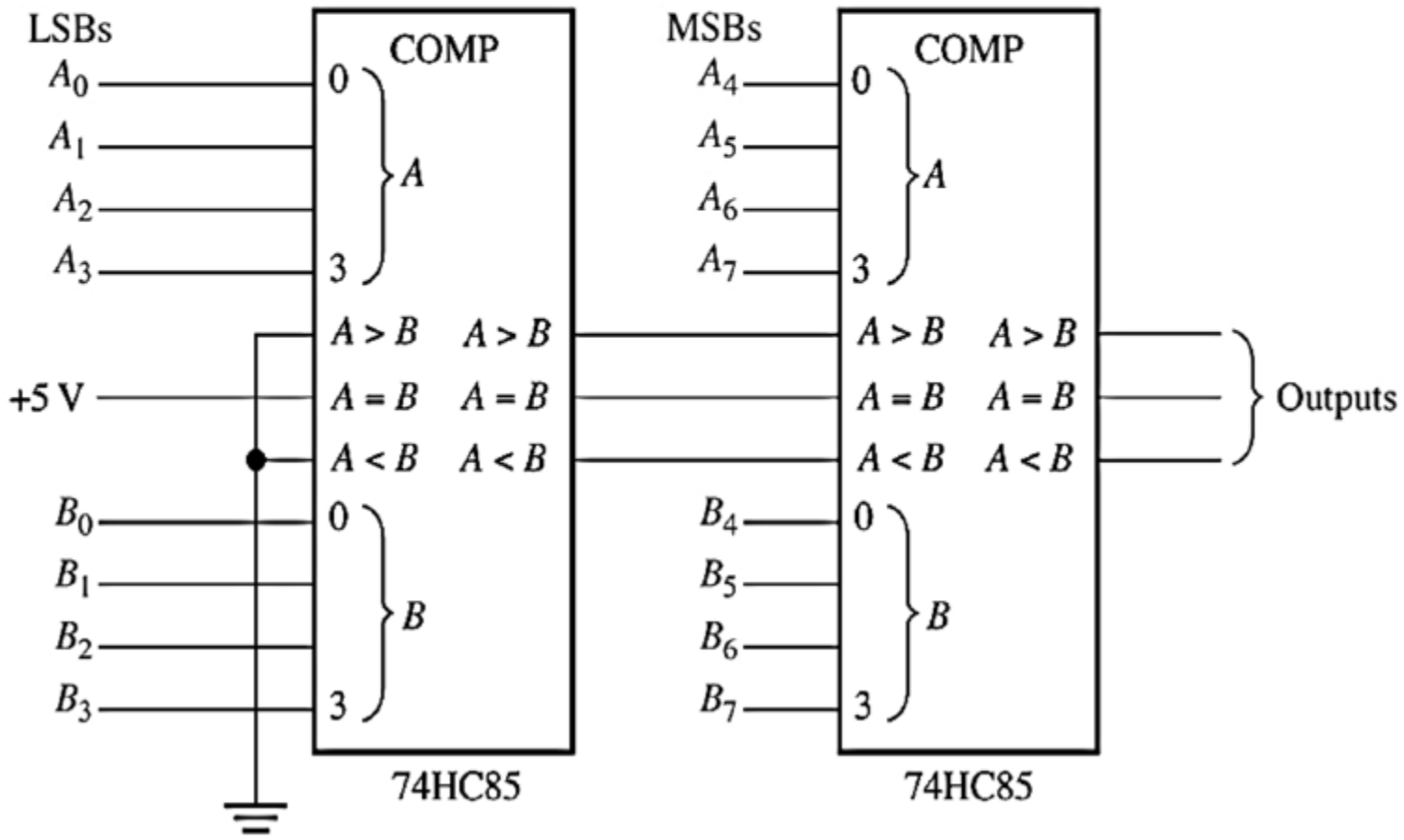
Step 2: Compare A_2 and B_2 □ Number A = B; compare next bits.

Step 3: Compare A_1 and B_1 □ Number A = B; compare next bits.

Step 4: Compare A_0 and B_0 □ Number A > B

A > B = 1, A < B = 0, A = B = 0
when A = 1011 and B = 1010

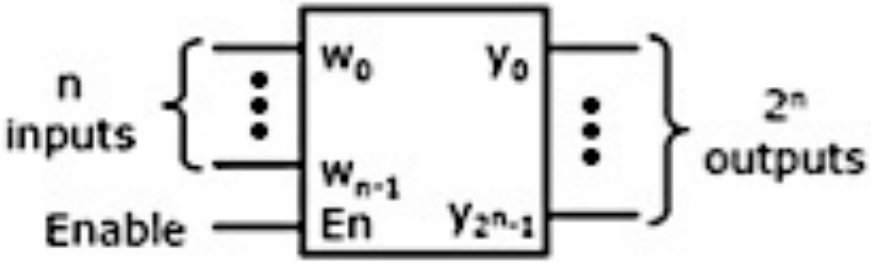
- For cascaded connection
  - Output of A>B, A=B, A< B must be connected to the corresponding input of the next stage
  - For lowest order comparator, A=B must be HIGH while A>B and A<B must be connected to LOW



Decoders

Decoders

- **Function**
  - detect the presence of a specified combination of bits (code) on its inputs
  - to indicate the presence of that code by a specified output level.
- Decoder has
  - *n* input lines to handle *n* bits and from one to 2^n output lines to indicate the presence of one or more *n*-bit combinations.
  - Output is 1 if its label matches input
  - Enable input pin (En) is used to enable or disable decoder outputs.



Types of Decoder

1) Binary decoder
   a) 3-Bits decoder (1 from 8) = 3-line to 8-line
   b) 4-Bit decoder (1 from 16) = 4-line to 16-line

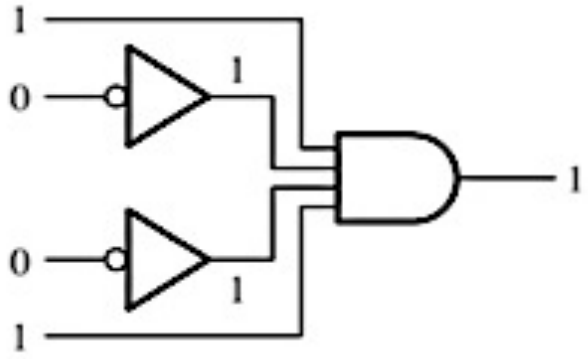
2) BCD to Decimal decoder

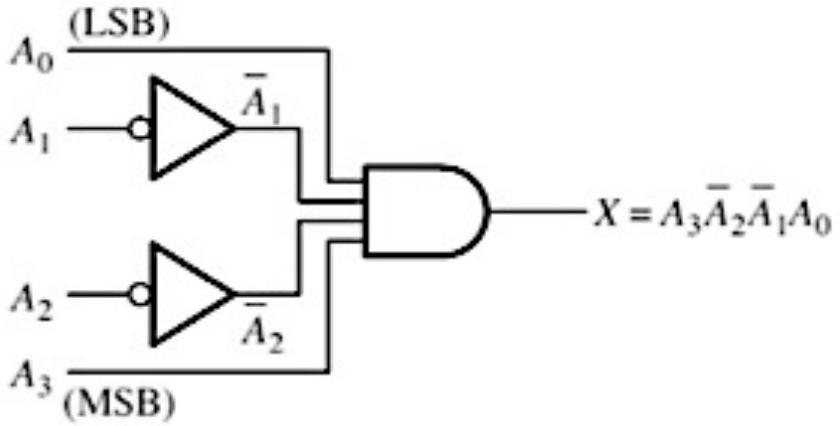
3) BCD to 7-segments

Decoder:

Binary Decoder

- A decoder that recognize a binary pattern at its input and activate the output whenever that pattern occurs
  - Use AND if we want the output to be HIGH
  - Use NAND if we want the output to be LOW
- Example : A decoder that produces a logic HIGH if 1001_2 occurs at the input

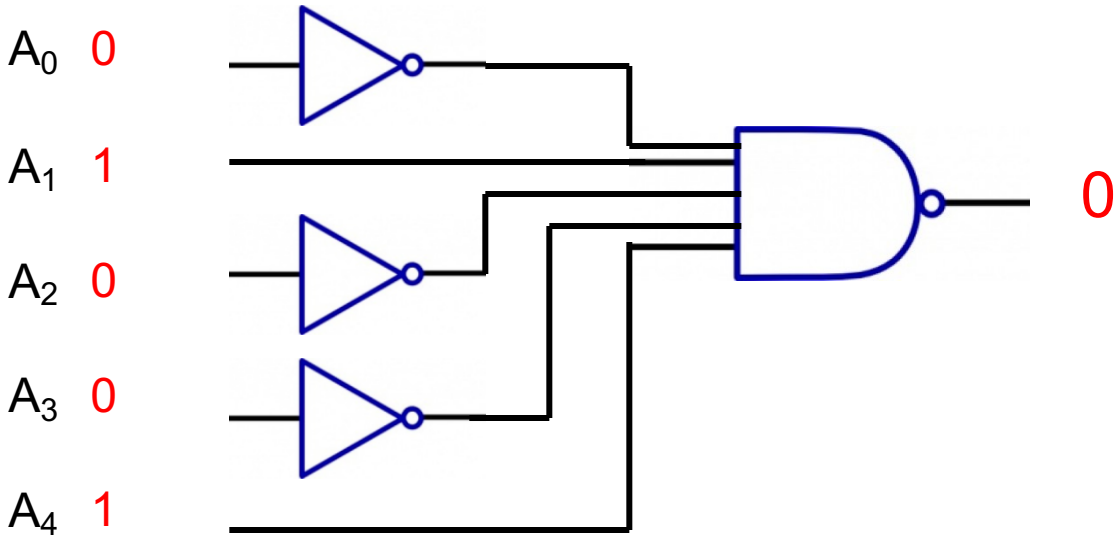




Extra

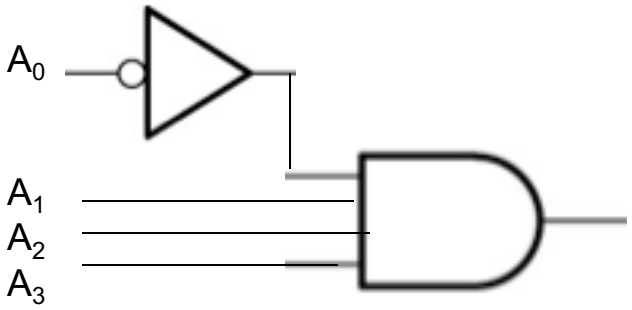
Exercise 6.4: Develop the logic required to detect the binary code 10010 and produce an active-LOW output.

Solution 6.4:

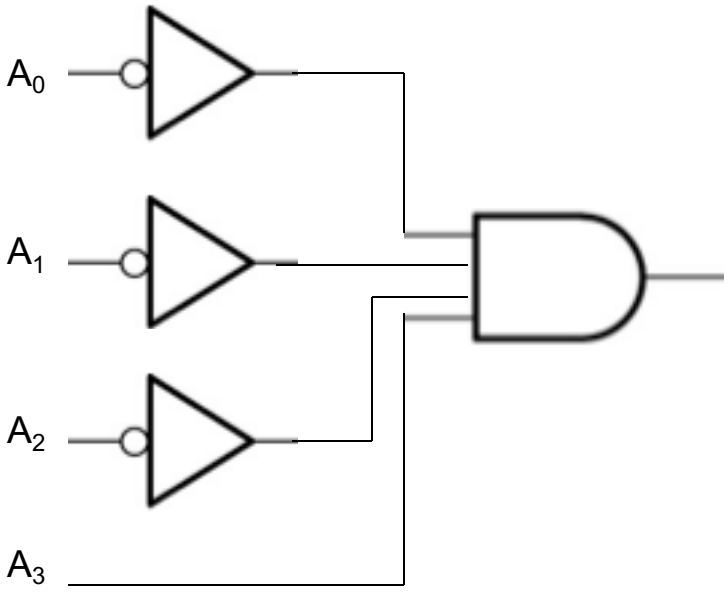


Extra

Exercise 6.5: When the output is active-HIGH for each of decoding gates in the Figure, what is the binary code appearing on the inputs? The MSB is A_3.



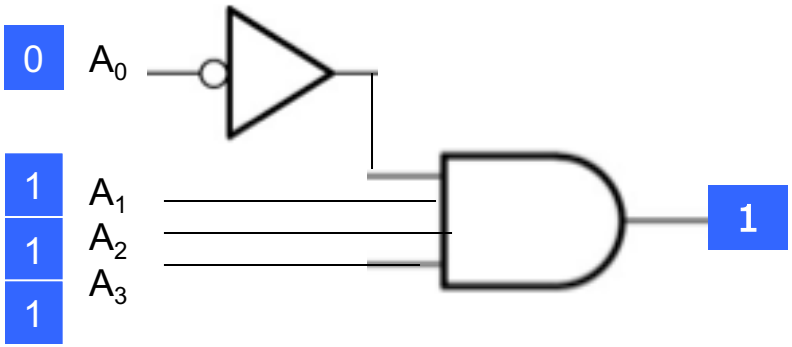
(a)

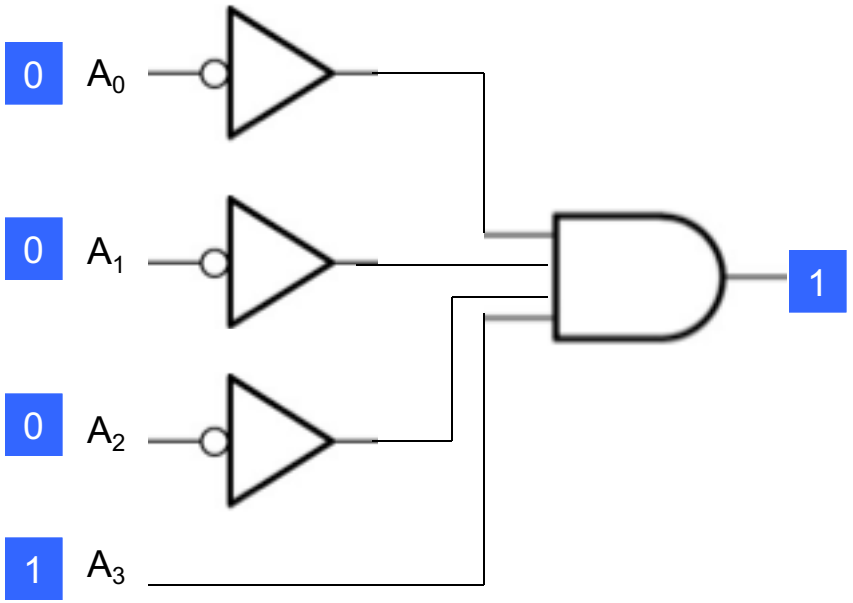


(b)

Extra

Solution 6.5:





(a) $A_3A_2A_1A_0 = 1110$

(b) $A_3A_2A_1A_0 = 1000$

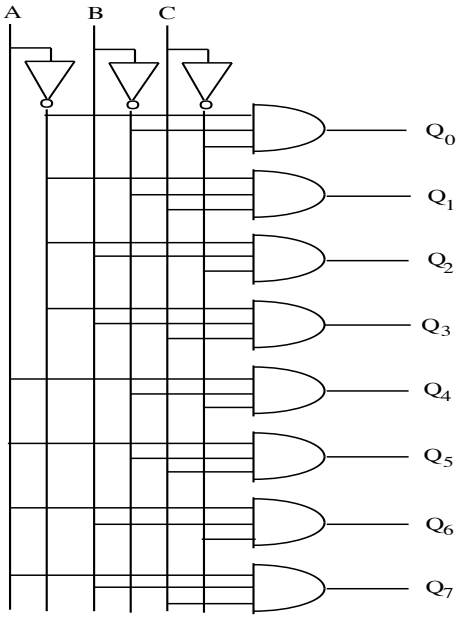
Binary Decoder: 3-Bit Decoder

- Sometimes it is good to have a general purpose decoder that can decode all the possibility of binary pattern at the input
  - Such a decoder has multiple output but its output will only be active if a unique binary pattern exist at the input

For this example:
Input active HIGH Output active HIGH

Decoding functions:	A	B	C	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7
$\overline{A}\overline{B}\overline{C}$	0	0	0	1	0	0	0	0	0	0	0
$\overline{A}\overline{B}C$	0	0	1	0	1	0	0	0	0	0	0
$\overline{A}B\overline{C}$	0	1	0	0	0	1	0	0	0	0	0
$\overline{A}BC$	0	1	1	0	0	0	1	0	0	0	0
$A\overline{B}\overline{C}$	1	0	0	0	0	0	0	1	0	0	0
$A\overline{B}C$	1	0	1	0	0	0	0	0	1	0	0
$AB\overline{C}$	1	1	0	0	0	0	0	0	0	1	0
ABC	1	1	1	0	0	0	0	0	0	0	1

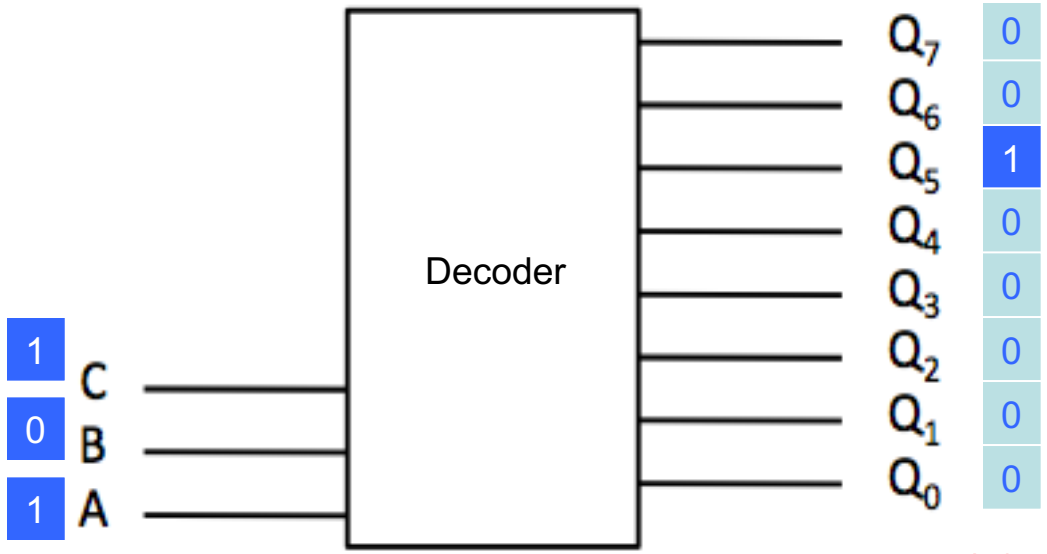
Truth Table for 3 bits decoder



Extra

AND is used because we want an active HIGH output.
The each input there will be only one HIGH output.

- The symbol is



$A\overline{B}C$

Active HIGH input

Example:

If the input = 101,

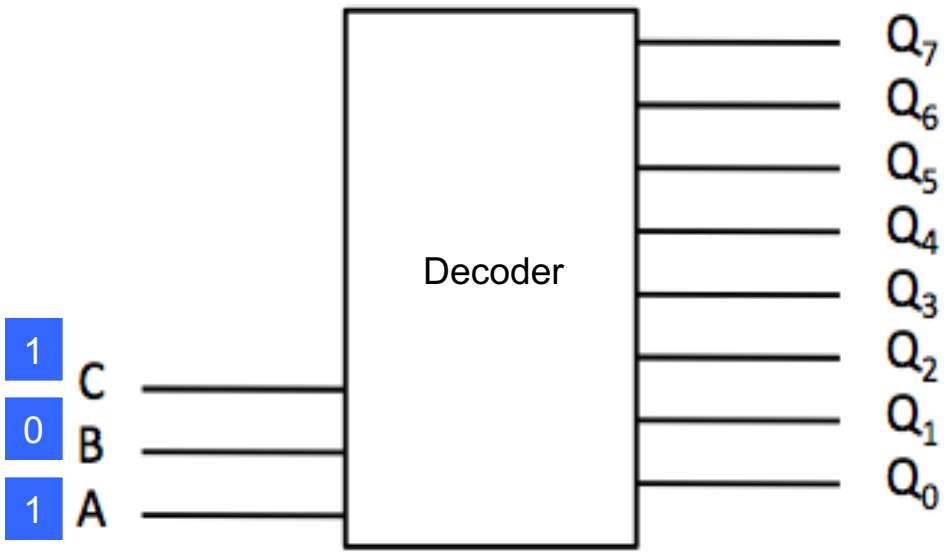
the output $Q_5 = 1$

Note:
Only one output will active with a unique binary pattern at input.

Extra

NAND is used because we want an active LOW output.
The each input there will be only one LOW output.

- The symbol is



$\overline{A}B\overline{C}$

Active LOW input

1 1 0 1 1 1 1 1

Active LOW output

Example:

If the input = 101,

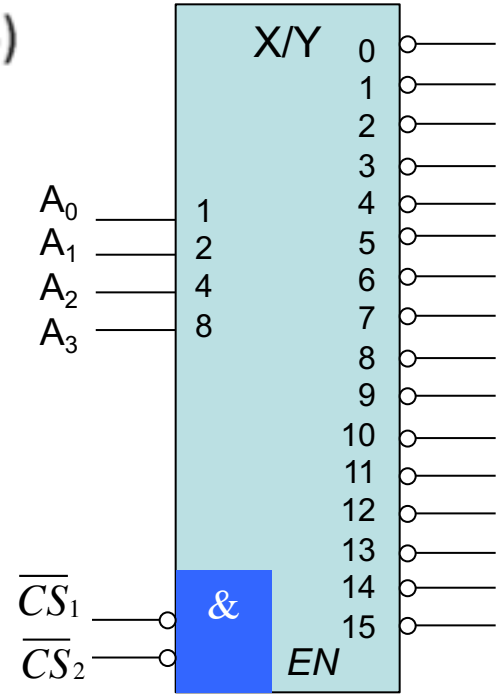
the output $Q_5 = 0$

Note:
Only one output will active with a unique binary pattern at input.

Decoder:
4-Bit Decoder

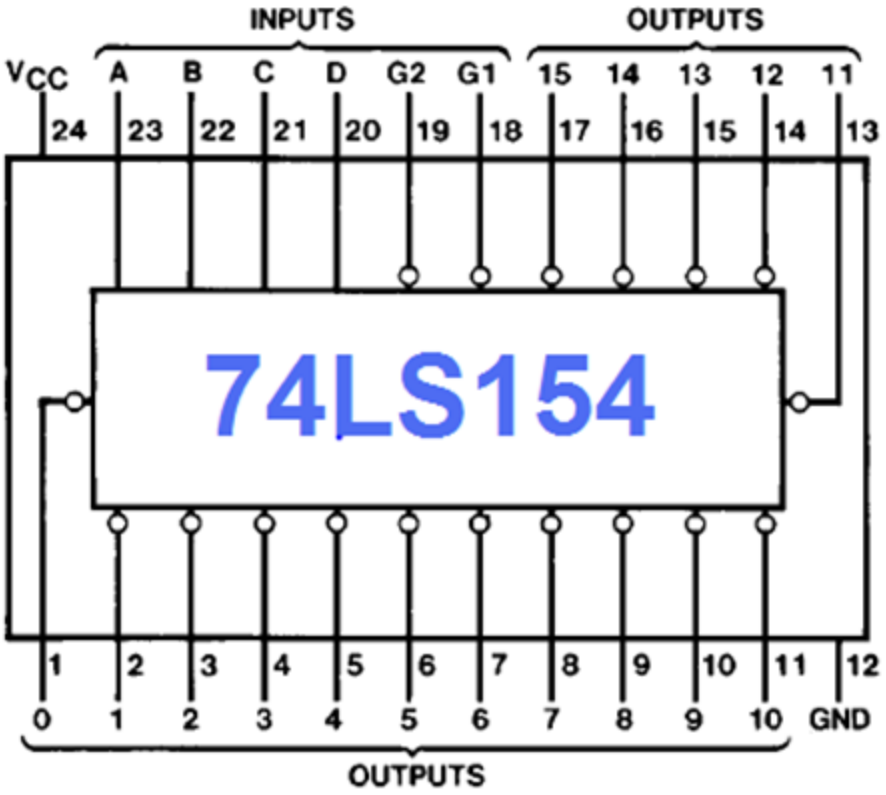
- Decoder can be called n to 2^n or 1 of 2^n for the same decoder
- The common MSI decoder chip
  - 2 to 4 (1 of 4), 3 to 8 (1 of 8) and 4 to 16 (1 of 16)

- 74HC154 is a 1 of 16 decoder
  - It has 4 active high input and 16 active low output
  - It also has 2 active low enable (EN) pin
    - This pin is used to select the IC if we use more than one chip
    - To use the IC, this two pin must be connected to logic LOW



Extra


VANXY
Fantasy Electronics Co.,Ltd
74HC154D
L0E5T4
Un61034B
FANTASY
Better Product, Better Service !


INPUTS
OUTPUTS
VCC A B C D G2 G1 15 14 13 12 11
24 23 22 21 20 19 18 17 16 15 14 13
74LS154
1 2 3 4 5 6 7 8 9 10 11 12
0 1 2 3 4 5 6 7 8 9 10 GND
OUTPUTS

The function table -- to descibe the function of the device

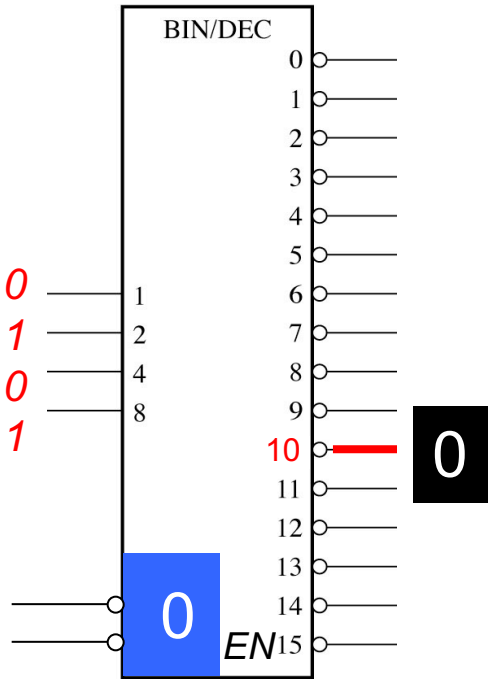
Note

1. H = HIGH voltage level
   L = LOW voltage level
   X = don't care

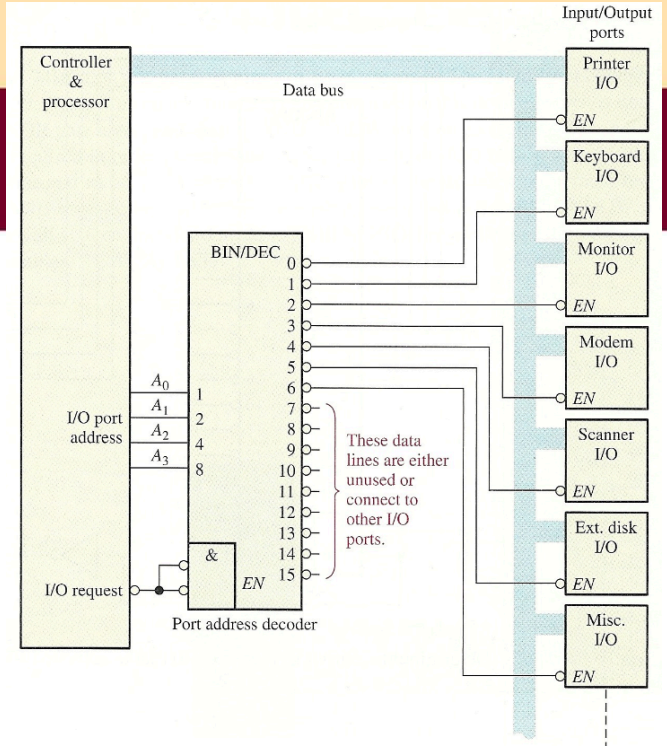
FUNCTION TABLE

		INPUTS				OUTPUTS														
$\overline{E}_0$	$\overline{E}_1$	A_0	A_1	A_2	A_3	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$	$\overline{Y}_4$	$\overline{Y}_5$	$\overline{Y}_6$	$\overline{Y}_7$	$\overline{Y}_8$	$\overline{Y}_9$	$\overline{Y}_{10}$	$\overline{Y}_{11}$	$\overline{Y}_{12}$	$\overline{Y}_{13}$	$\overline{Y}_{14}$	$\overline{Y}_{15}$
H	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	L	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	L	L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	H	L	L	L	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	H	H	L	L	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	L	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H
L	L	H	L	H	L	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H
L	L	L	H	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H
L	L	H	H	H	L	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H
L	L	L	L	L	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H
L	L	H	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H
L	L	L	H	L	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H
L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H
L	L	L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H
L	L	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H
L	L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L

Extra



A	B	C	D	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
0	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
1	0	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0



Example:

Address Decoder.

Decoder is use a lot to decode an address of a computer.

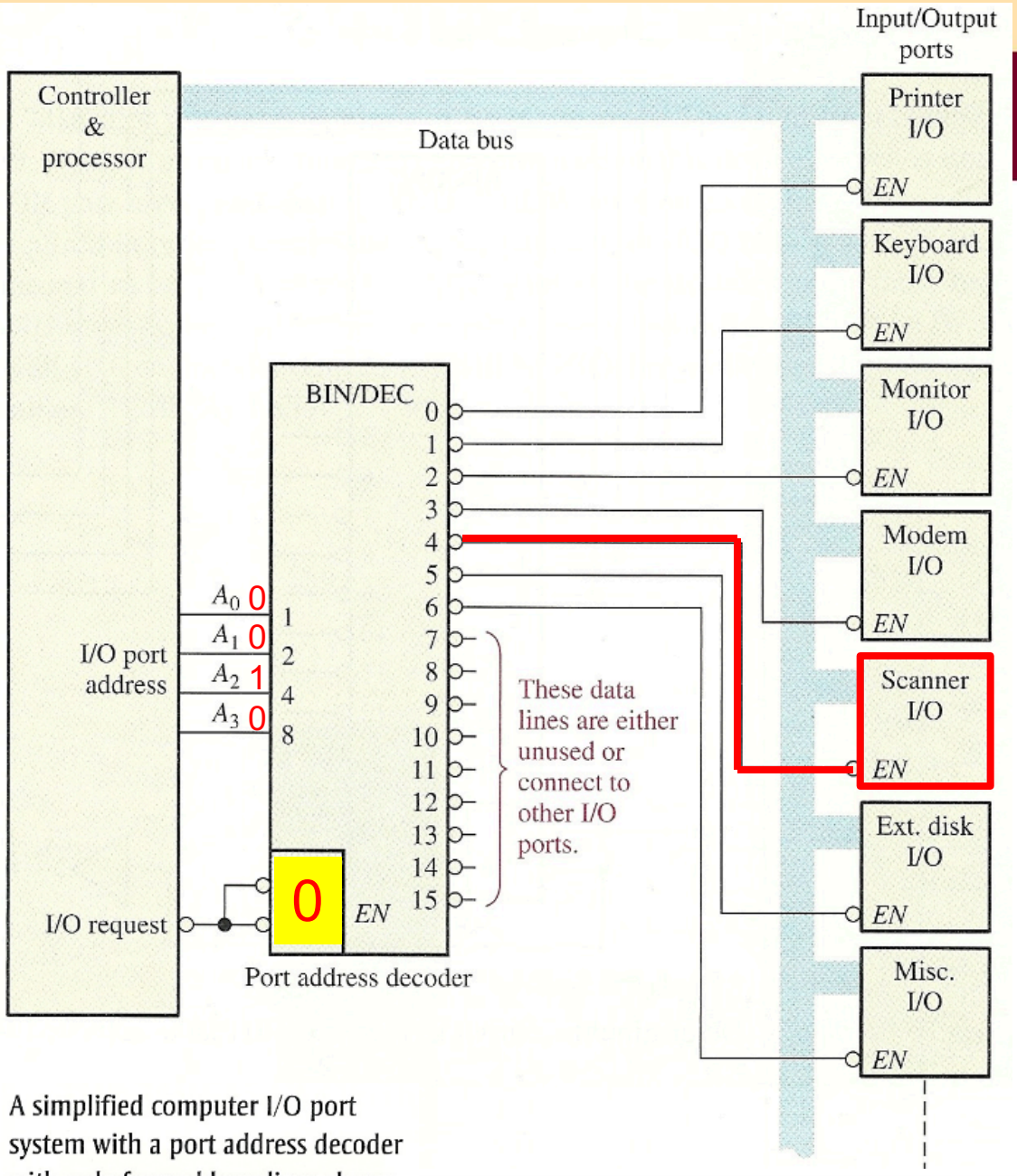
- In a computer system every devices is given an address (like an ID) to identify the devices
- Address must be unique for a device, some devices uses a range of addresses
- Decoder is used to decode the address
- Whenever a computer try to access the I/O device it will issue an I/O request by making that signal LOW
    - this will enable an active low enable pin of a decoder
- Next the address of an I/O device will be sent by the computer as $A_3A_2A_1A_0$
- This address will be decoded and activate one of the output (becomes LOW)
    - If $A_3A_2A_1A_0 = 0110$, then output 6 will be LOW, all other output are HIGH
    - The LOW output will enable an active LOW device at address 6

Extra

Example:

Port Address: 0100_2
I/O Request : LOW(0)

I/O Port: Scanner



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

Extra

Review . . .

Binary Coded Decimal (BCD)

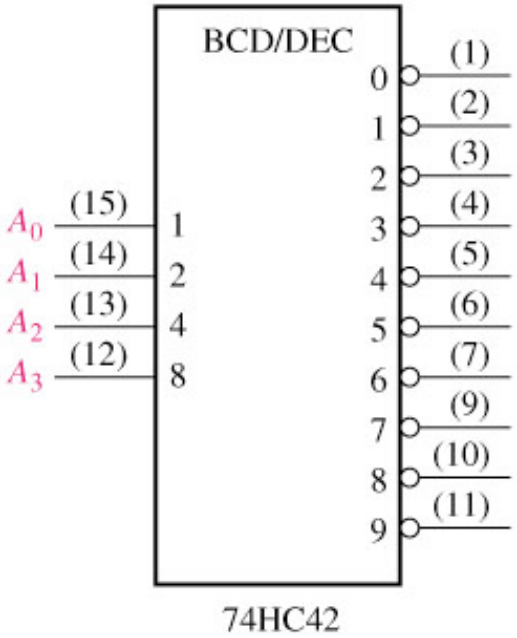
- BCD is a way to express each of the decimal digits with a binary code.
- There are only 10 code groups in the BCD system

0	1	2	3	4	5	6	7	8	9
0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

- Example: 43_{10} = **0100 0011**

Decoder: BCD to Decimal

- Convert from BCD to decimal
  - Frequently referred as 1 of 10 decoder
  - The same as binary decoder except only 10 decoding gates required
    - An active LOW output MSI Chip is 74HC42



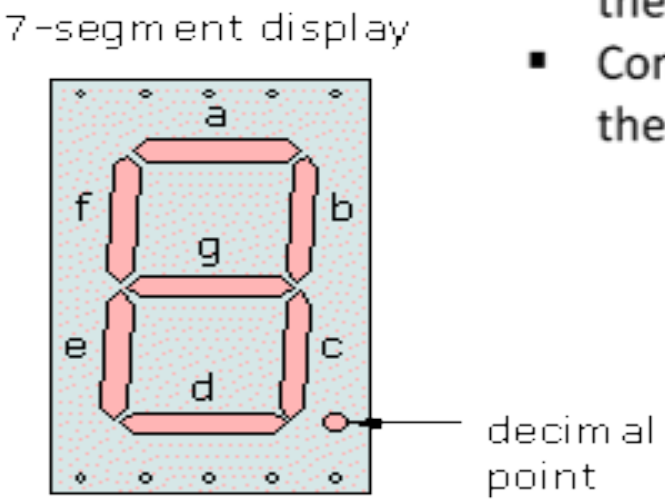
74HC42

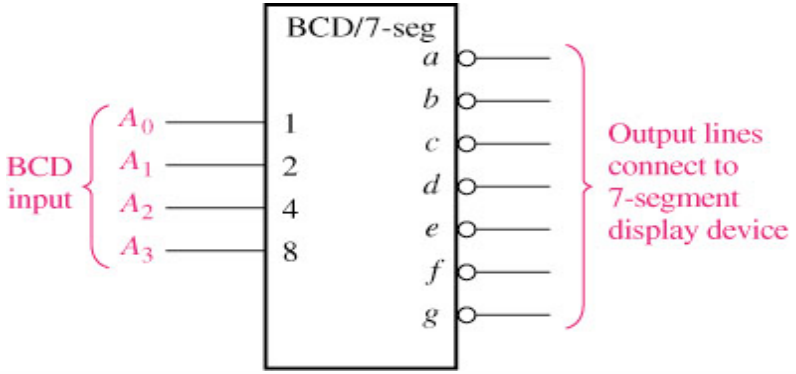
Decimal digit	A_3	A_2	A_1	A_0	Decoding Function
0	0	0	0	0	$\bar{A}_3\bar{A}_2\bar{A}_1\bar{A}_0$
1	0	0	0	1	$\bar{A}_3\bar{A}_2\bar{A}_1 A_0$
2	0	0	1	0	$\bar{A}_3\bar{A}_2 A_1\bar{A}_0$
3	0	0	1	1	$\bar{A}_3\bar{A}_2 A_1 A_0$
4	0	1	0	0	$\bar{A}_3 A_2\bar{A}_1\bar{A}_0$
5	0	1	0	1	$\bar{A}_3 A_2\bar{A}_1 A_0$
6	0	1	1	0	$\bar{A}_3 A_2 A_1\bar{A}_0$
7	0	1	1	1	$\bar{A}_3 A_2 A_1 A_0$
8	1	0	0	0	$A_3\bar{A}_2\bar{A}_1\bar{A}_0$
9	1	0	0	1	$A_3\bar{A}_2\bar{A}_1 A_0$

The decoding function , where each output has a unique function

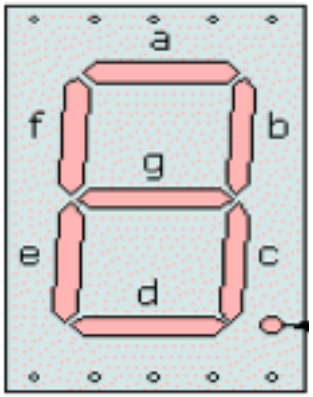
Decoder: BCD to 7-Segments

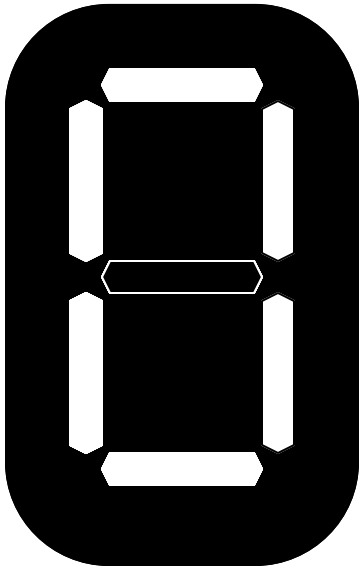
- Receive a BCD code at the input and lit the proper LED of the 7-segment display based on the BCD value
- The display shows the decimal numbers 0-9
- The individual segments making up a 7-segment display are identified by letters, from *a* to *g*
  - 2 type of 7-segment display
    - Common anode – 0 at it input will lit the segment therefore require a decoder with an active LOW output
    - Common cathode- 1 at it input will lit the segment therefore require a decoder with an active HIGH output





BCD inputs				Segment outputs							Display
D	C	B	A	a	b	c	d	e	f	g	
0	0	0	0	1	1	1	1	1	1	0	0
0	0	0	1	0	1	1	0	0	0	0	1
0	0	1	0	1	1	0	1	1	0	1	2
0	0	1	1	1	1	1	1	0	0	1	3
0	1	0	0	0	1	1	0	0	1	1	4
0	1	0	1	1	0	1	1	0	1	1	5
0	1	1	0	1	0	1	1	1	1	1	6
0	1	1	1	1	1	1	0	0	0	0	7
1	0	0	0	1	1	1	1	1	1	1	8
1	0	0	1	1	1	1	1	0	1	1	9

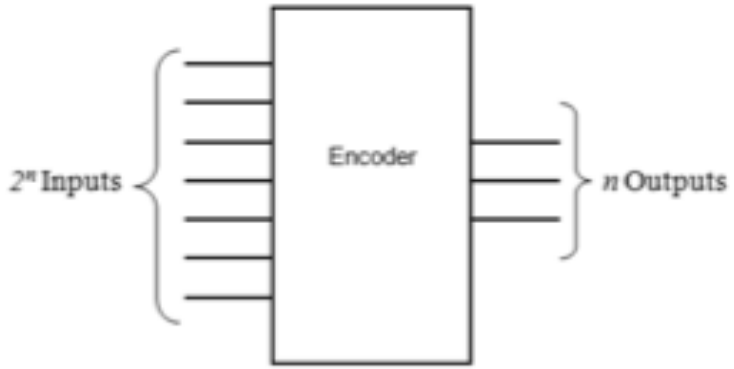




Encoders

Encoders

- Encoder does the opposite of the decoder
  - E.g: convert Decimal to BCD
- A ***binary encoder encodes information (data) from*** $2n$ *inputs into an n-bit code (output)*
  - Exactly one of the inputs should have a value of one
  - The outputs represent the binary number that identifies which input is equal to 1
- Encoders reduce the number of bits needed to represent given information

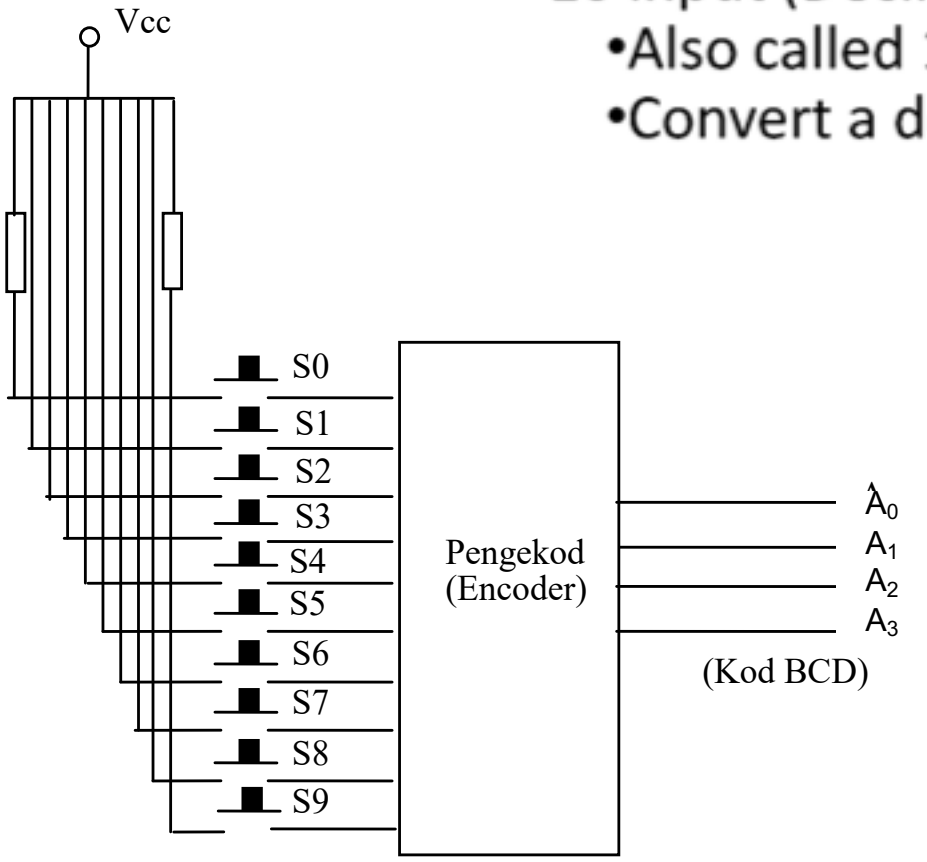


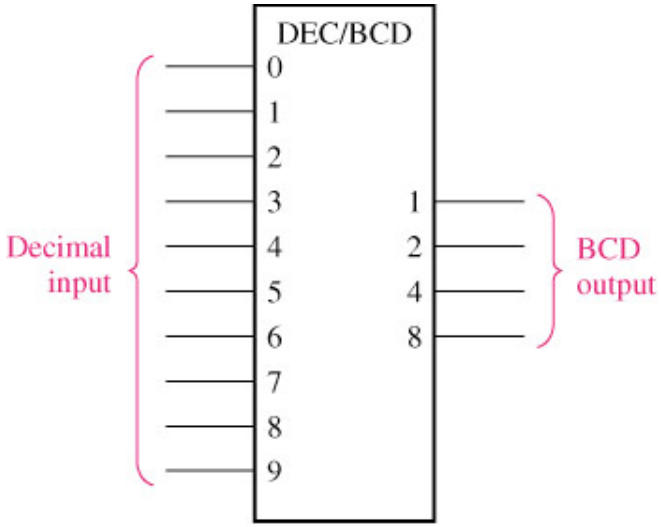
Type of Encoders:

- Decimal to BCD Encoder
- Priority Encoder

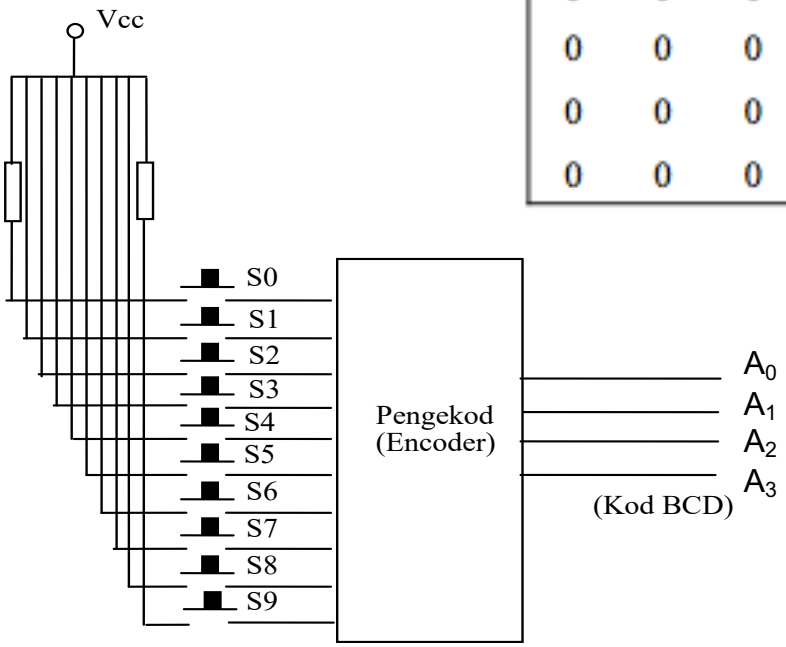
Encoder:
Decimal to BCD

- 10 input (Decimal) and 4 output (BCD)
  - Also called 10 line to 4 line encoder
  - Convert a decimal value to a BCD code





S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	A_3	A_2	A_1	A_0
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	0	0	0	1	1	0
0	0	0	0	0	0	0	1	0	0	0	1	1	1
0	0	0	0	0	0	0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	1	0	0	1



Logic symbols for a decimal to BCD encoder.

S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	A_3	A_2	A_1	A_0
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	0	0	0	1	1	0
0	0	0	0	0	0	0	1	0	0	0	1	1	1
0	0	0	0	0	0	0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	1	0	0	1

- From truth table
  - A_0 HIGH if switch 1,3,5,7,9 HIGH
  - A_1 HIGH if switch 2,3,6,7 HIGH
  - A_2 HIGH if switch 4,5,6,7 HIGH
  - A_3 HIGH if switch 8,9 HIGH
    - Therefore OR all the input to make the corresponding output HIGH

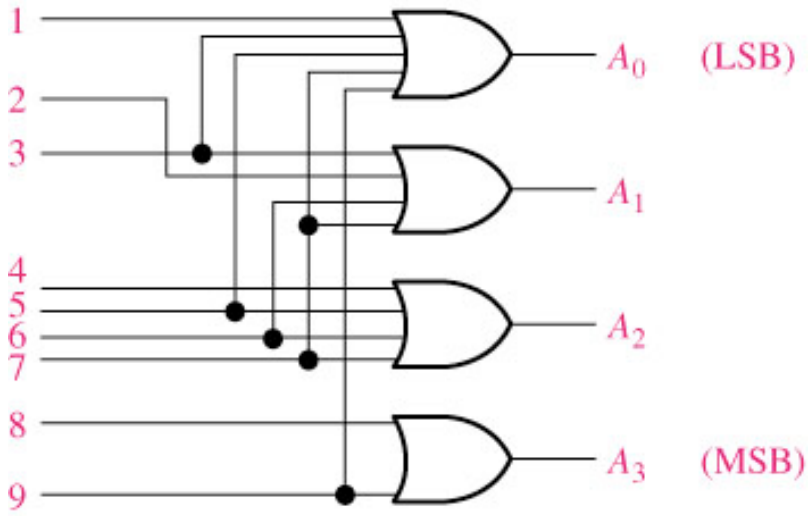


Figure: Basic logic diagram of a decimal-to-BCD encoder. A 0-digit input is not needed because the BCD outputs are all LOW when there are no HIGH inputs.

Extra

(Module: Page 194)

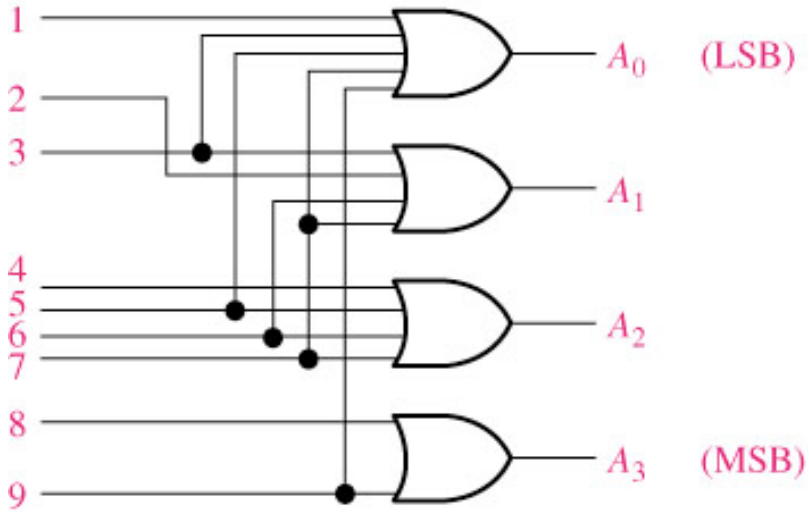
Exercise 6.6: Suppose, HIGH are applied to input 2 and 9 of the circuit

(a) What are the states of the output lines?

(b) Does this represent a valid BCD code?

(c) What is the restriction on the encoder logic?

Solution 6.6:



(a) A_3= 0, A_2= 0, A_1= 1, A_0= 0
A_3= 1, A_2= 0, A_1= 0, A_0= 1

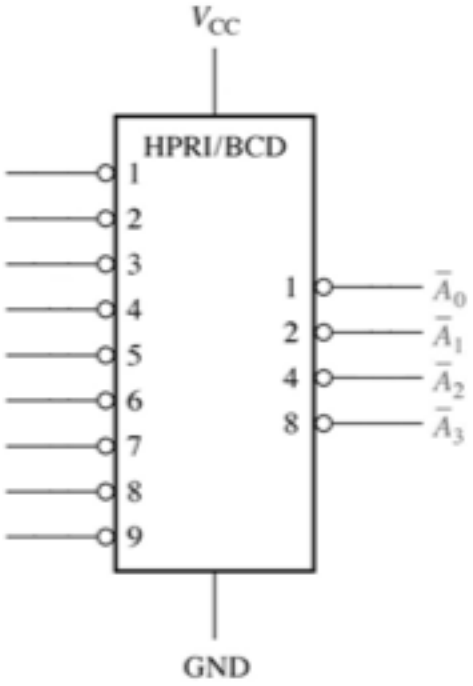
(b) YES. The valid BCD code is between 0-9

(c) Only one input can be HIGH; the rest must be LOW

Encoder: Priority Encoder

- IC 74HC147 perform the same encoding function but with additional feature
  - Produce BCD output corresponding to the higher decimal value at the input and ignore the lower value
    - HPRI – highest value input has priority

- Eg:
  - if pin 9 and 5 LOW at the same time, only pin 9 will be recognize because it has a higher priority
  - The output will be 0110 for 9 (because of active low at the output)



Multiplexers (MUX)

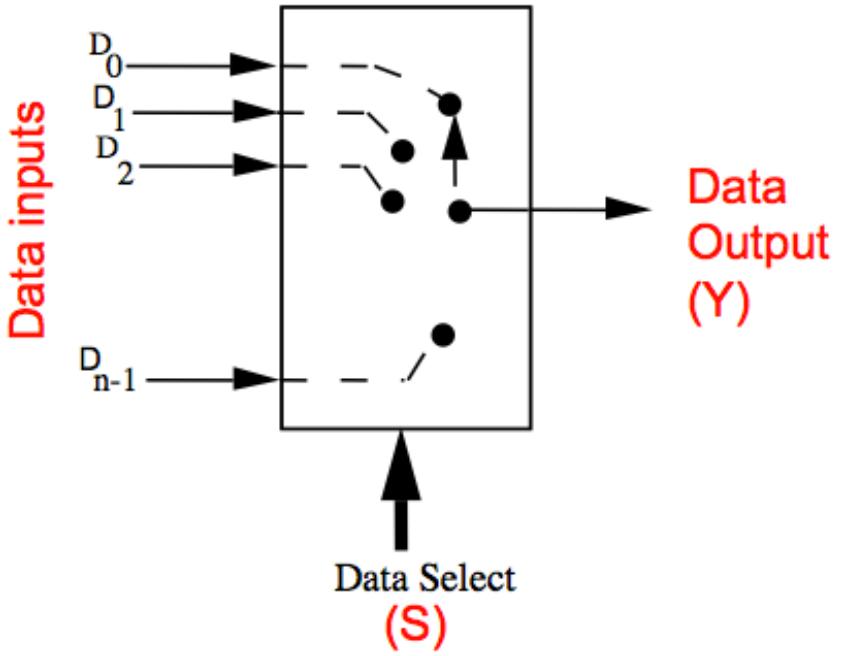
Multiplexers (DEMUX)

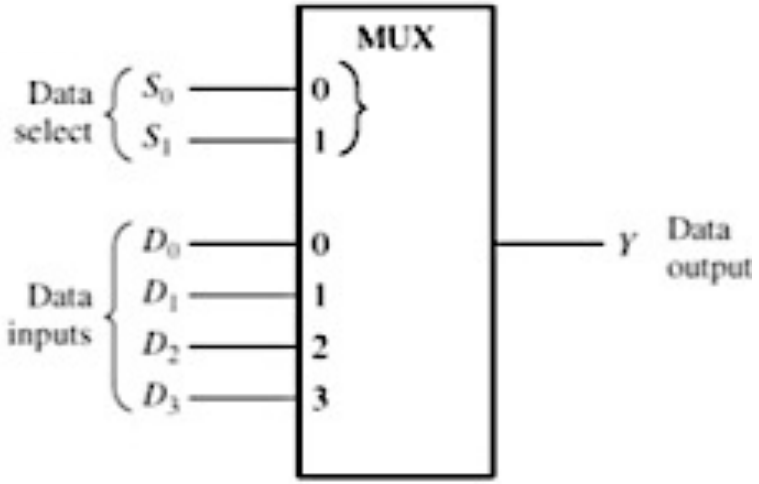
Multiplexers (MUX)
(Data Selector)

- A multiplexer (MUX) is a device that allows digital information from several sources to be routed onto a single line for transmission over that line to a common destination.
  - The basic multiplexer has several data-input lines and a single output line, ***Many to One***
  - It also has data-select inputs, which permit digital data on any one of the inputs to be switched to the output line.
    - Select line control which input routed to output

$$\text{number of select bits} = \frac{\log(\text{number of input})}{\log 2}$$

- MUX are also known as data selector
- DEMUX are also known as data distributor





$2^{1(\text{select bit})} = 2\ (\text{total input bit})$

$2^{2(\text{select bit})} = 4\ (\text{total input bit})$

Example: Multiplexer 1-of-2

The Design:

When S_0 =0, Y = D_0 and when S_0 = 1 , Y = D_1

A truth table is produced from the specification.

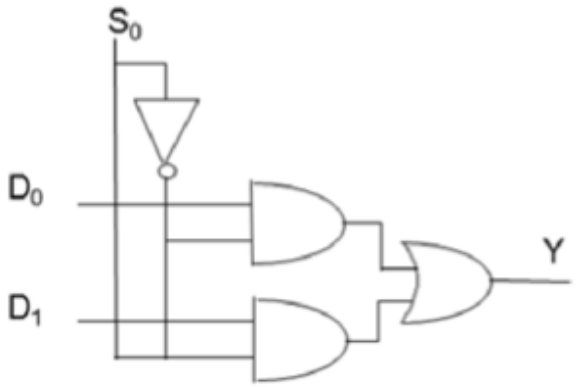
S_0	D_1	D_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

A more compact form can be produced.

S_0	Data Output
0	$Y = D_0$
1	$Y = D_1$

The circuit equation:

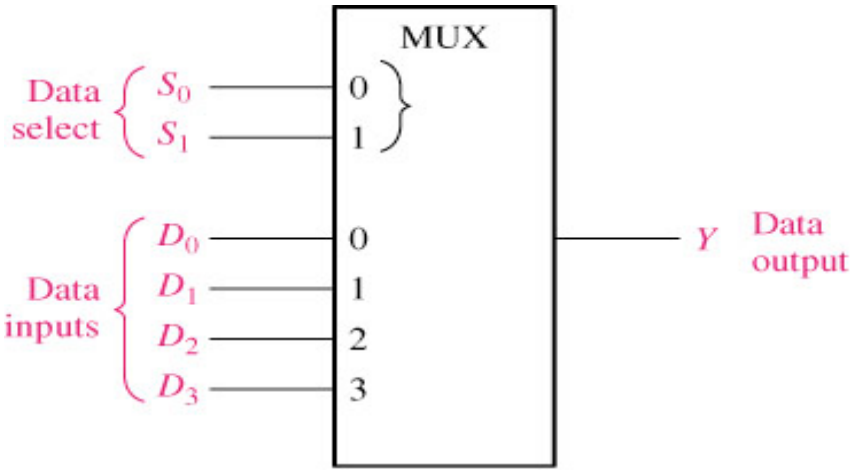
$$Y = \overline{S}_0 D_0 + S_0 D_1$$

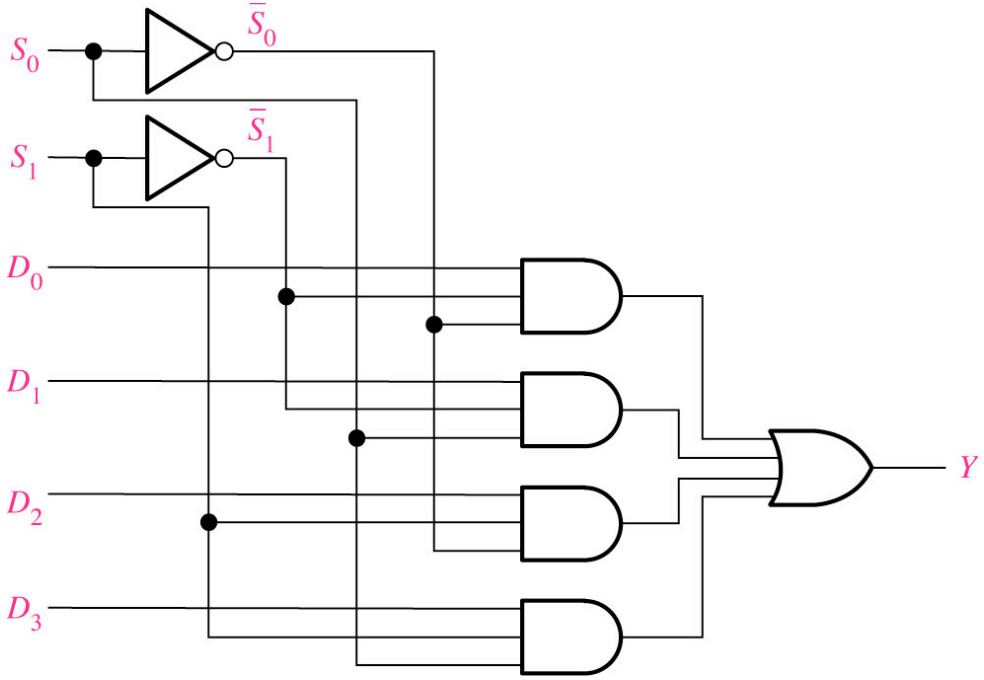


Data- select input		Data output
S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

Example: Multiplexer 1-of-4

- Able to route 4 inputs to 1 output
  - Requires 2 bits data selector





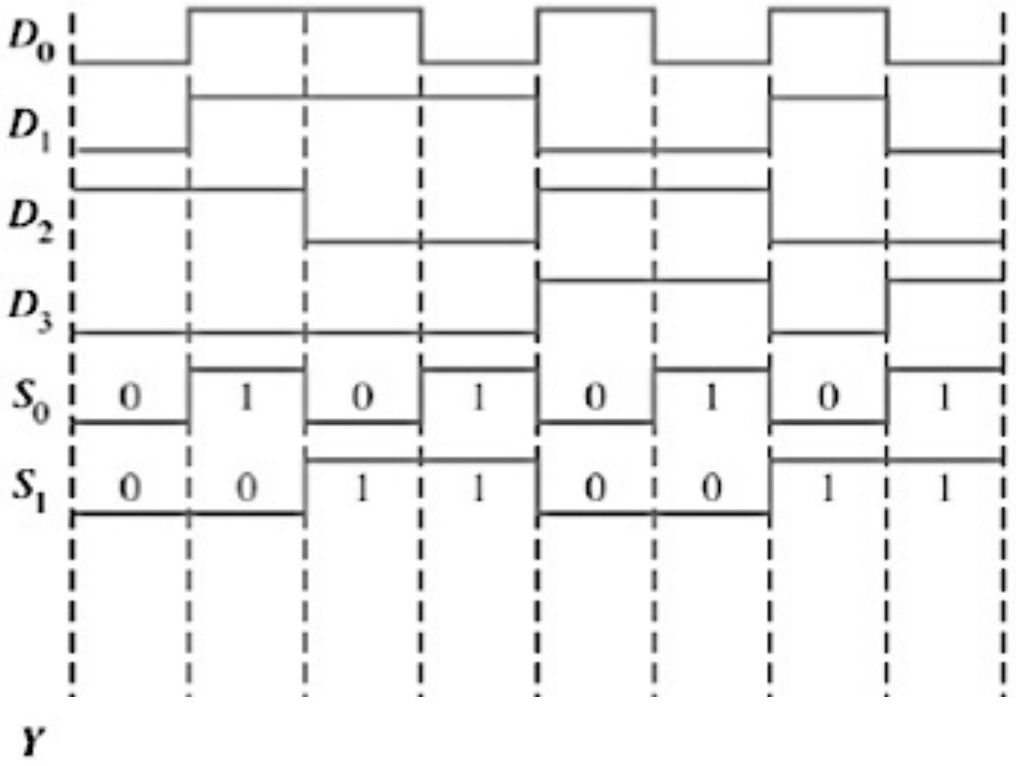
Timing Diagram to show the working of 1 of 4 Multiplexer

- The output will depend on the value of selector
- S_1 is the MSB and S_0 the LSB

Exercise 7:

The data-input and data-select waveforms in the Figure are applied to the multiplexer.

Determine the output waveform Y, in the relation to the inputs.



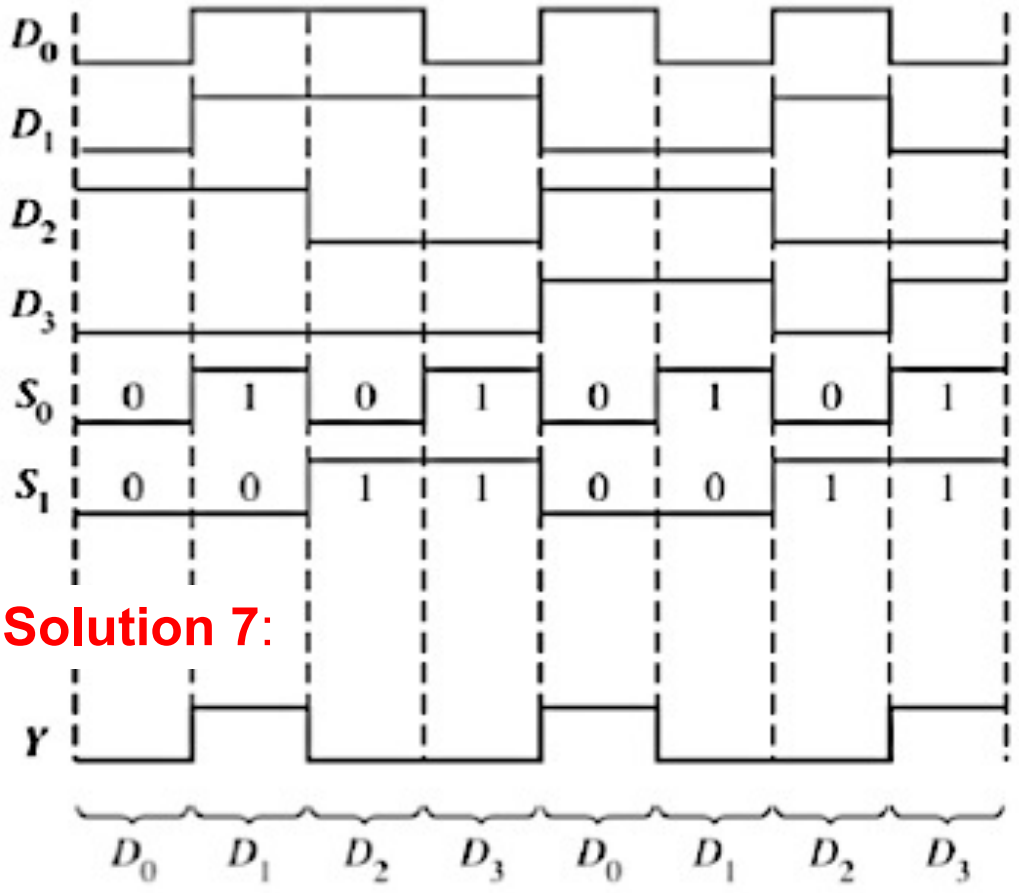
Timing Diagram to show the working of 1 of 4 Multiplexer

- The output will depend on the value of selector
- S_1 is the MSB and S_0 the LSB

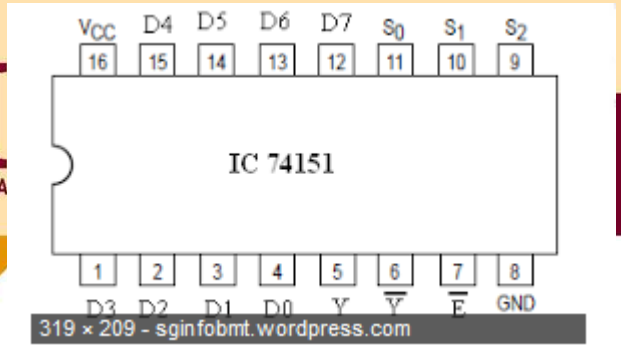
Exercise 7:

The data-input and data-select waveforms in the Figure are applied to the multiplexer.

Determine the output waveform Y, in the relation to the inputs.

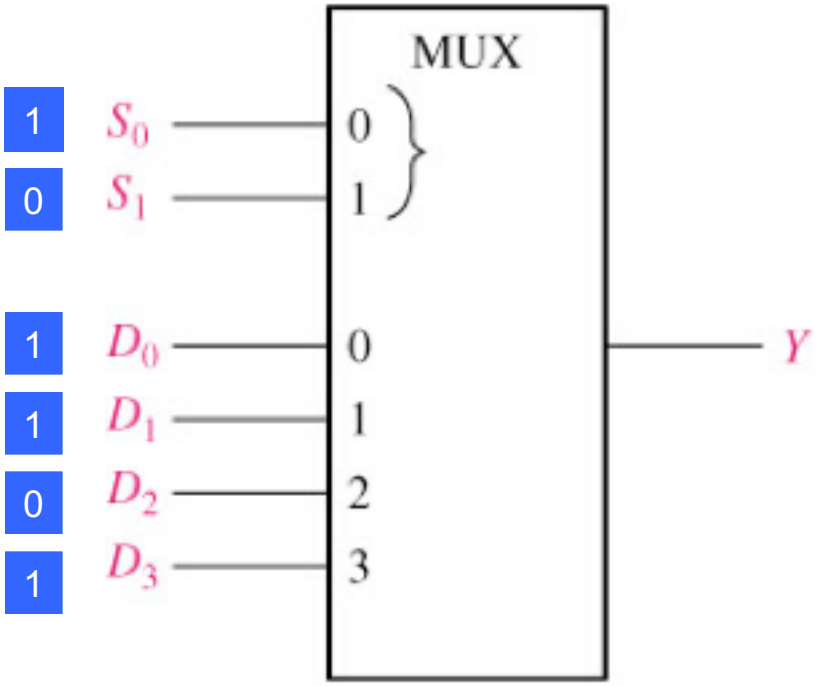


Solution 7:



Extra

Exercise 8: What is the output of Y if the binary values of S and D are given in the Figure.



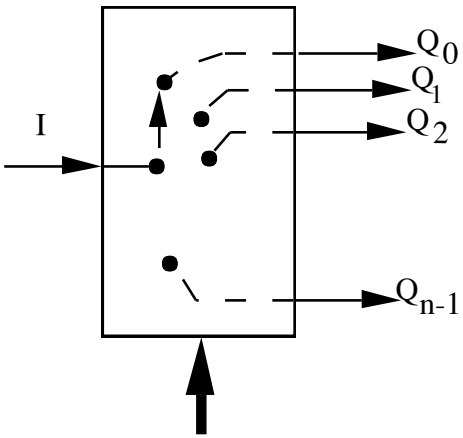
Solution 8:

Data- select input		Data output
S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

$Y = D_1 = 1$

Demultiplexers (DEMUX)
(Data Distributor)

- A DEMUX is basically reverse the multiplexing function.
- It takes digital information from one line and distributes it to a given number of output lines.
- For this reason, DEMUX is also known as a data distributor.
- Decoder can also be used as DEMUX.



If only I = 1

S_1 S_0	Q_3 Q_2 Q_1 Q_0
0 0	0 0 0 **1**
0 1	0 0 **1** 0
1 0	0 **1** 0 0
1 1	**1** 0 0 0

The Design

- Truth Table

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

$$D_3 = S_1 S_0 \, DataIn$$

$$D_2 = S_1 \overline{S_0} \, DataIn$$

$$D_1 = \overline{S_1} S_0 \, DataIn$$

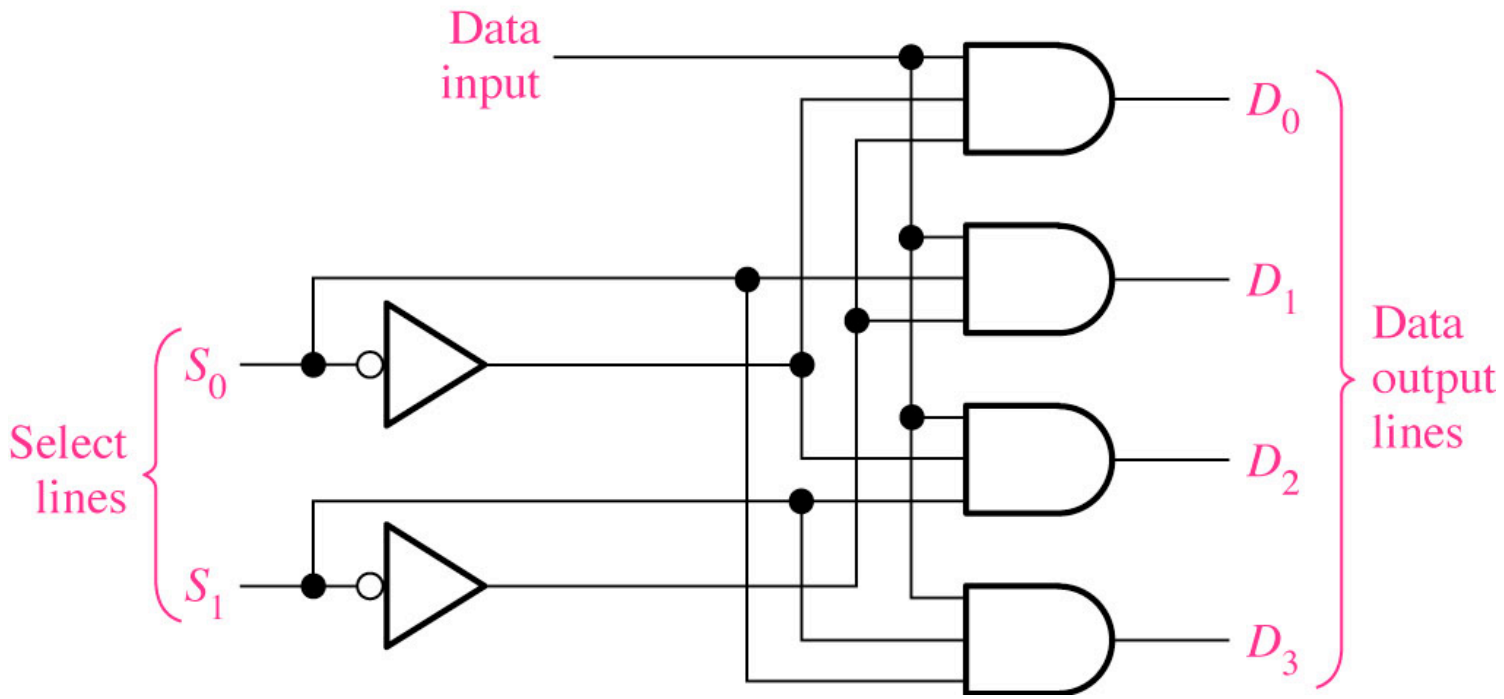
$$D_0 = \overline{S_1} \, \overline{S_0} \, DataIn$$

- A more compact Table

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

Extra

The data-input lines goes to all the AND gates.
The 2 data-select lines enable only 1 gate at a time.



Code Converter

Parity Generators / Checkers

Code Converter

- The purpose of ***code converter circuits is to*** convert from one type of code to another type of code
  - For example:
    - A 3-to-8 decoder converts from a binary number to a one-hot encoding at the output
    - A 8-to-3 encoder performs the opposite
- Many different types of code converter circuits can be constructed
  - One common example a a BCD-to7-segment decoder as explained in a decoder section

Parity Generators / Checkers

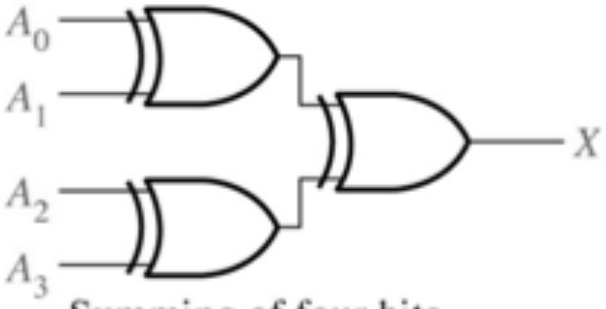
- Errors can occur as digital codes are being transferred from one point to another within a digital system.
- The errors take the form of undesired changes in the bits that make up the coded information; that is, a 1 can change to a 0, or a 0 to a 1.
- When an error occurs undetected, it can cause serious problem in a digital system.

Basic Parity Logic

- To determine whether no of 1's or 0's odd or even

- Use Modulo 2 summation
- Device that can do Modulo 2 is XOR
- If number of input bits
  - EVEN ($A_1A_0 = 00$ or $A_1A_0 = 11$) then X =0
  - ODD ($A_1A_0 = 01$ or $A_1A_0 = 10$) then X = 1

Summing of two bits



Summing of four bits

- **Parity Checker** – when this device is used as an parity checker, the number of input bits always be even, and when a parity error occurs, the Σ Even output goes LOW and the ∑ Odd output goes HIGH.

- **Parity Generator** – If this device is used as an EVEN parity generator, the parity bit is taken at the ∑ Odd output because this output is a 0 if there is an even number of input bits and it is a 1 if there is an odd number.

End of Module 6