

SEEE 1223

DIGITAL ELECTRONICS

CHAPTER 6: COMBINATIONAL MSI

DR. MOHD SAIFUL AZIMI BIN MAHMUD

P19a-04-03-30

School of Electrical Engineering

Faculty of Engineering

Universiti Teknologi Malaysia

019-7112948

azimi@utm.my

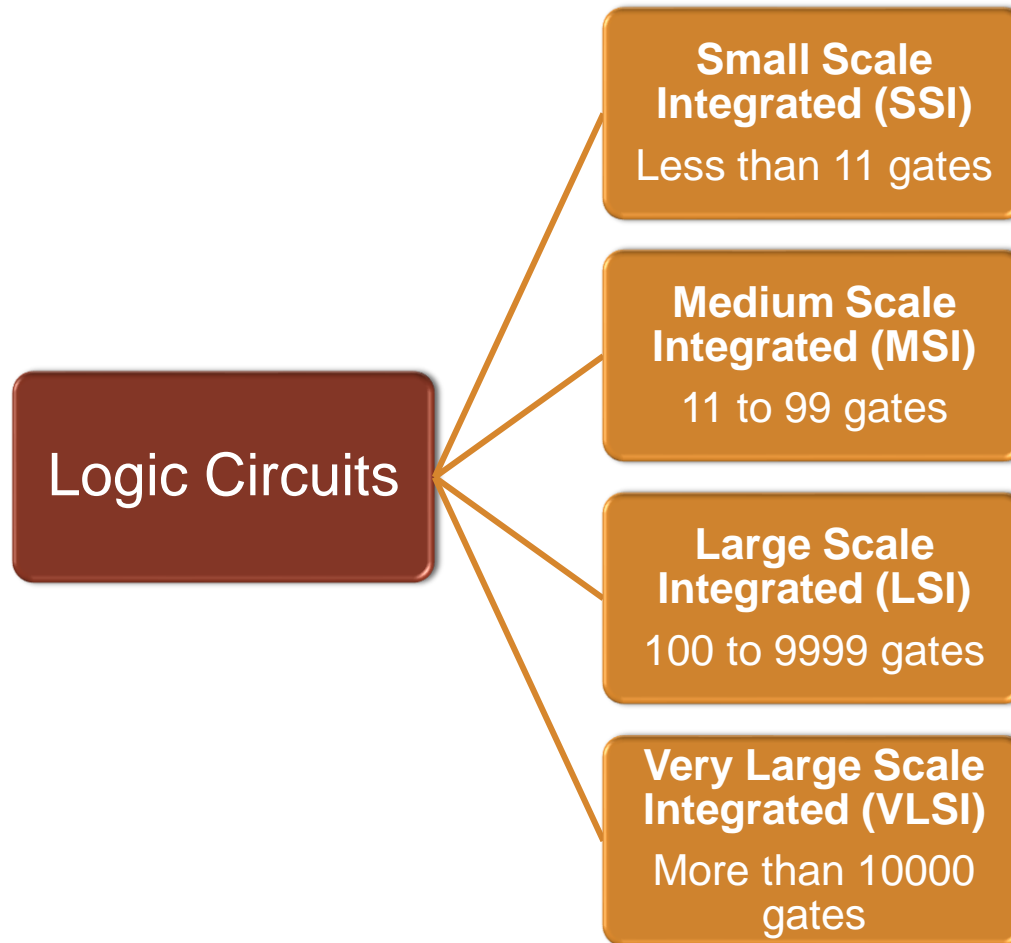


MSI CIRCUIT

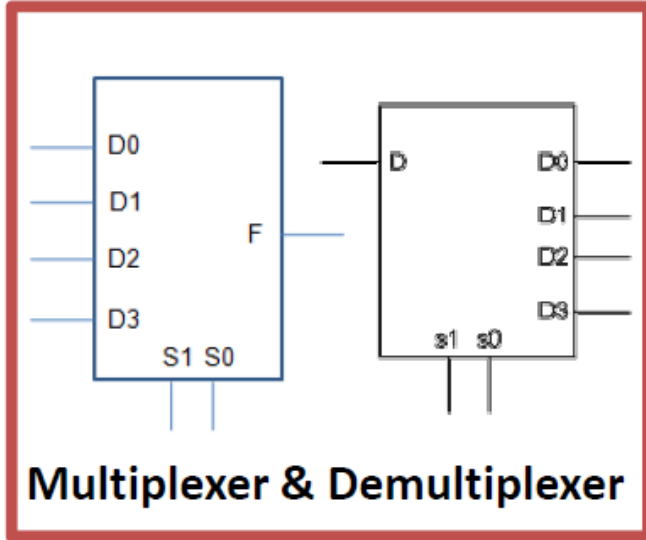
MSI CIRCUIT

INTRODUCTION

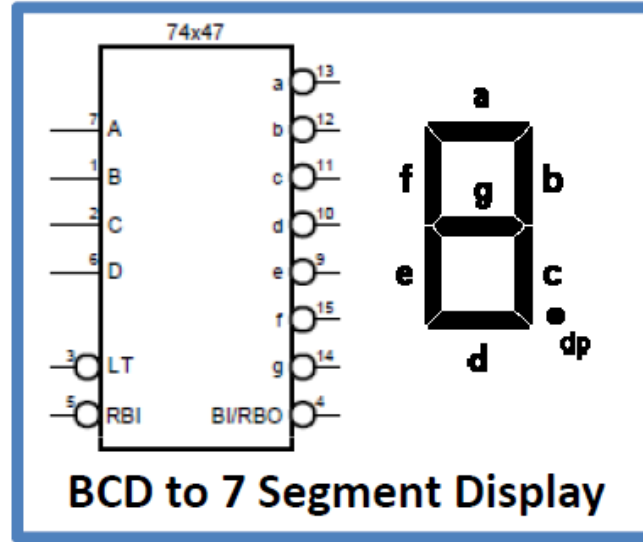
- **MSI (Medium scale integrated)** circuits are logic circuit that contains 11 to 99 logic gates in a circuit.



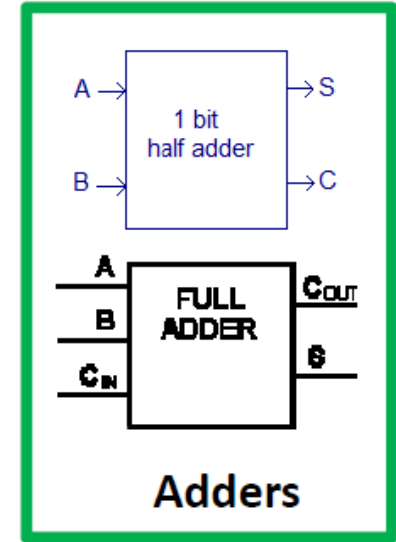
MSI CIRCUIT INTRODUCTION



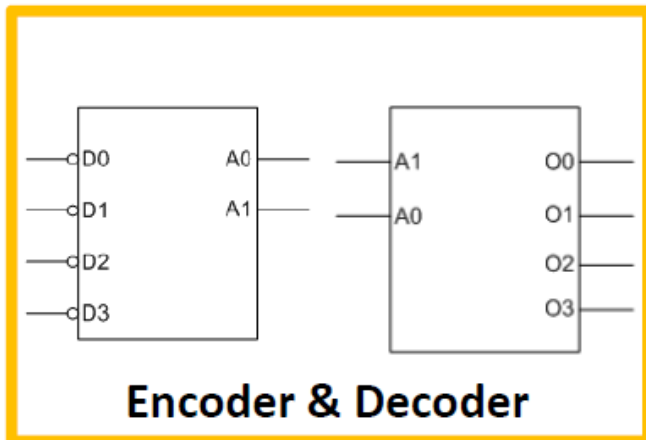
Multiplexer & Demultiplexer



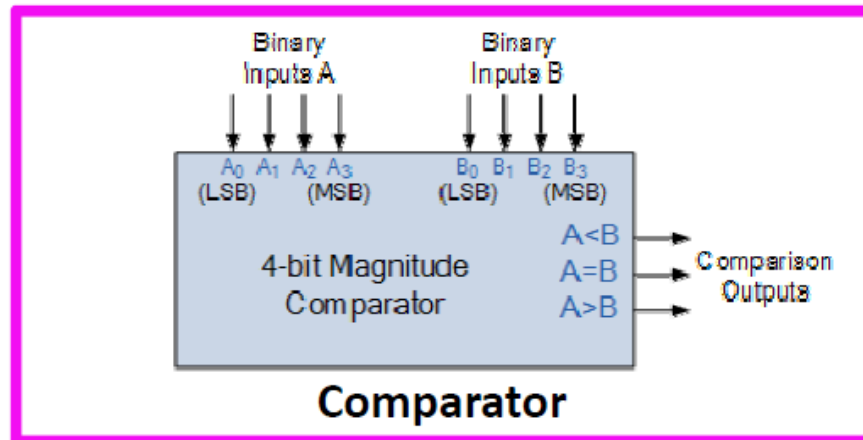
BCD to 7 Segment Display



Adders



Encoder & Decoder



Comparator

MULTIPLIER

MULTIPLEXER

INTRODUCTION

- **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.
- Basic Multiplexer has **2^n data-inputs, n data-selector inputs and one single output.**
- Multiplexers are also known as **data selectors.**
- Multiplexers usually written as **$(Y) \times 1$** or **$(Y): 1$** , where Y is the number of input data lines.

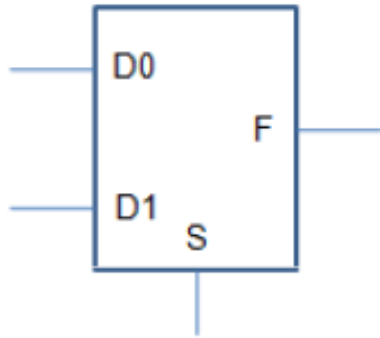
Example

A 4 input data line multiplexer is written as **4: 1 mux.**

MULTIPLEXER

2:1 MUX

- **2:1 Mux** consists of **2 inputs, 1 selector and 1 output.**



Symbol for 2:1 Mux

Function:

$$F = D0 \text{ when } S = 0$$

$$F = D1 \text{ when } S = 1$$

Function Table

<i>S</i>	<i>F</i>
0	<i>D0</i>
1	<i>D1</i>

- How to design a 2:1 mux using:
 - K-map? What are the input and output?
 - By inspection of its function?

MULTIPLEXER

2:1 MUX

- Design **2:1 Mux** using K-map.
- As we know, 2:1 mux consists of **2 inputs, 1 output and 1 selector**.
- Inputs = **$D0, D1, S$**
- Output = **F**

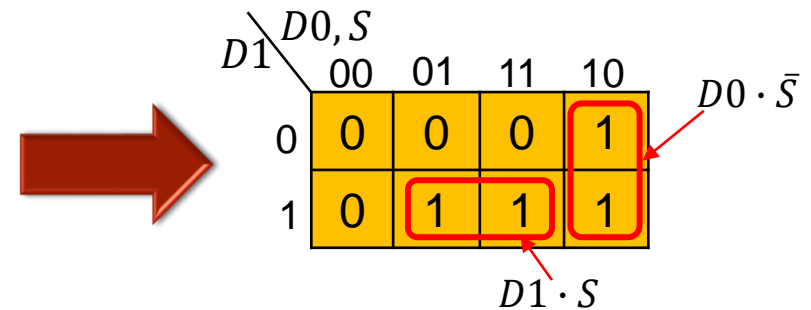
Truth Table

Inputs			Output
$D1$	$D0$	S	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	1	1
1	1	1	1

Function:

$F = D0$ when $S = 0$

$F = D1$ when $S = 1$



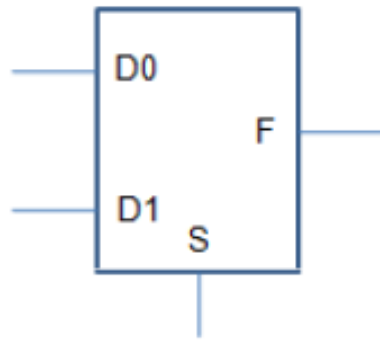
$$F = D0 \cdot \bar{S} + D1 \cdot S$$

Can you draw the logic circuit?

MULTIPLEXER

2:1 MUX

- Design **2:1 Mux** by inspection of its function



Symbol for 2:1 Mux

As we know, 2:1 mux function:

$$F = D0 \text{ when } S = 0 \quad \longrightarrow \quad F = D0 \cdot \bar{S}$$

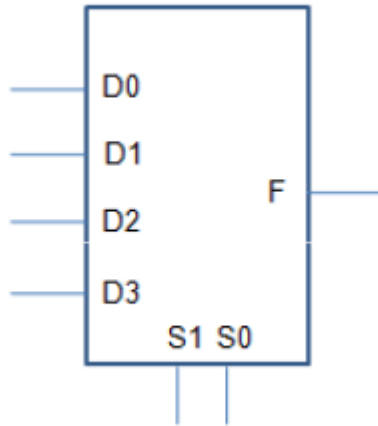
$$F = D1 \text{ when } S = 1 \quad \longrightarrow \quad F = D1 \cdot S$$

- Therefore, $F = D0 \cdot \bar{S} + D1 \cdot S$

MULTIPLEXER

4:1 MUX

- **4:1 Mux** consists of **4 inputs, 2 selectors and 1 output.**



Function:

$$F = D0 \text{ when } S1 = 0 \text{ and } S0 = 0$$

$$F = D1 \text{ when } S1 = 0 \text{ and } S0 = 1$$

$$F = D2 \text{ when } S1 = 1 \text{ and } S0 = 0$$

$$F = D3 \text{ when } S1 = 1 \text{ and } S0 = 1$$

Function Table

S1	S0	F
0	0	D0
0	1	D1
1	0	D2
1	1	D3

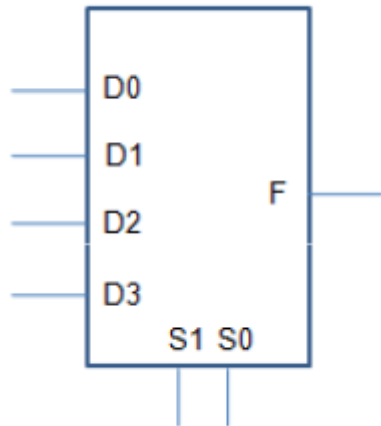
Symbol for 4:1 Mux

- How to design a 4:1 mux using:
 - K-map? What are the input and output?
 - By its function?

MULTIPLEXER

4:1 MUX

- Since 4:1 Mux has six inputs ($D3, D2, D1, D0, S1, S0$) and one output (F), therefore it is difficult/time consuming to use K-maps.
- Thus by looking at the functions of 4:1 mux:



Function:

$$F = D0 \text{ when } S1 = 0 \text{ and } S0 = 0 \rightarrow F = D0 \cdot \overline{S1} \cdot \overline{S0}$$

$$F = D1 \text{ when } S1 = 0 \text{ and } S0 = 1 \rightarrow F = D1 \cdot \overline{S1} \cdot S0$$

$$F = D2 \text{ when } S1 = 1 \text{ and } S0 = 0 \rightarrow F = D2 \cdot S1 \cdot \overline{S0}$$

$$F = D3 \text{ when } S1 = 1 \text{ and } S0 = 1 \rightarrow F = D3 \cdot S1 \cdot S0$$

Symbol for 4:1 Mux

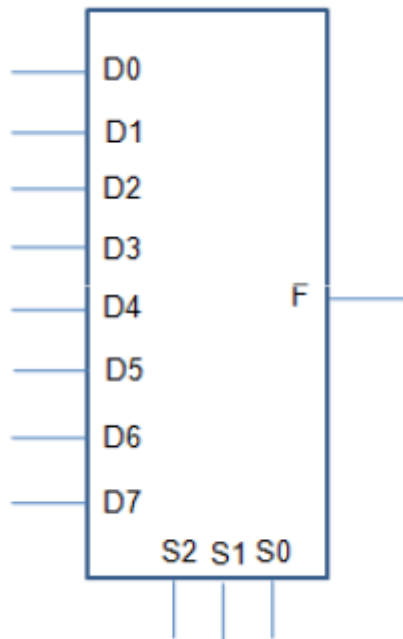
- Therefore, $F = D0 \cdot \overline{S1} \cdot \overline{S0} + D1 \cdot \overline{S1} \cdot S0 + D2 \cdot S1 \cdot \overline{S0} + D3 \cdot S1 \cdot S0$

Can you draw the logic circuit?

MULTIPLEXER

8:1 MUX

- **8:1 Mux** consists of **8 inputs, 3 selectors and 1 output.**



Function:

$F = D0$ when $S2 = 0$, $S1 = 0$ and $S0 = 0$

$F = D1$ when $S2 = 0$, $S1 = 0$ and $S0 = 1$

$F = D2$ when $S2 = 0$, $S1 = 1$ and $S0 = 0$

$F = D3$ when $S2 = 0$, $S1 = 1$ and $S0 = 1$

$F = D4$ when $S2 = 1$, $S1 = 0$ and $S0 = 0$

$F = D5$ when $S2 = 1$, $S1 = 0$ and $S0 = 1$

$F = D6$ when $S2 = 1$, $S1 = 1$ and $S0 = 0$

$F = D7$ when $S2 = 1$, $S1 = 1$ and $S0 = 1$

Function Table

S2	S1	S0	F
0	0	0	D0
0	0	1	D1
0	1	0	D2
0	1	1	D3
1	0	0	D4
1	0	1	D5
1	1	0	D6
1	1	1	D7

Symbol for 8:1 Mux

What is the logic expression for F ?

MULTIPLEXER

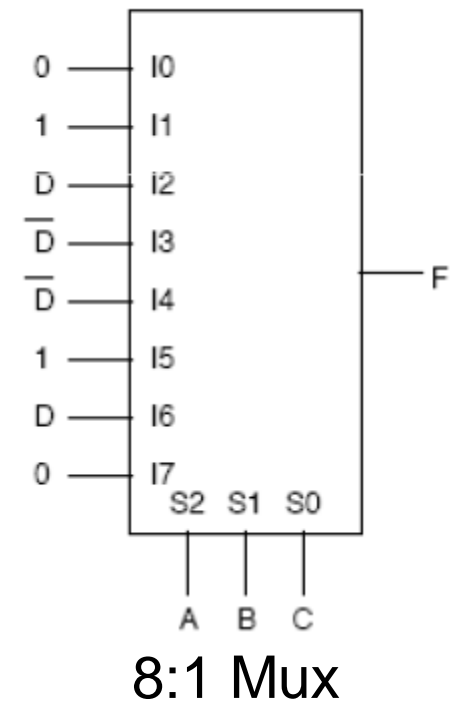
MULTIPLEXER APPLICATIONS

Example

Implement $F(A, B, C, D) = \sum m(2, 3, 5, 6, 8, 10, 11, 13)$ using an 8:1 Mux

Inputs				Output
A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0

Inputs				Output
A	B	C	D	F
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0



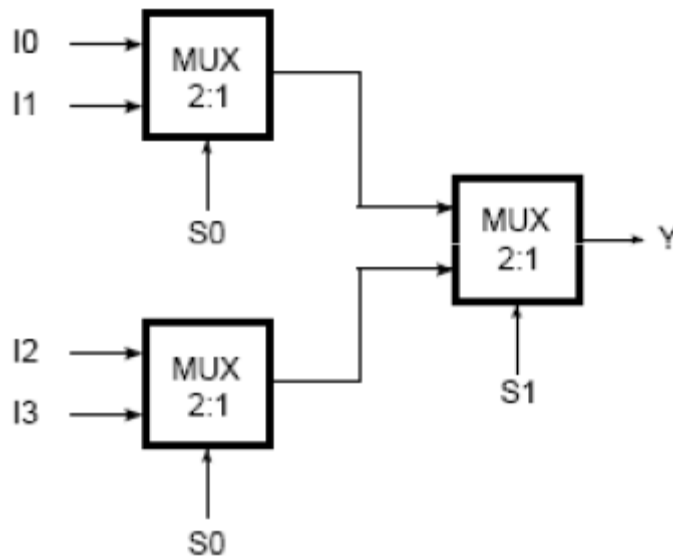
MULTIPLEXER

MULTIPLEXER EXPANSIONS

- A few multiplexers can be combined to built a bigger multiplexer.

Example

A 4:1 Mux can be built by combining **three** 2:1 Mux.

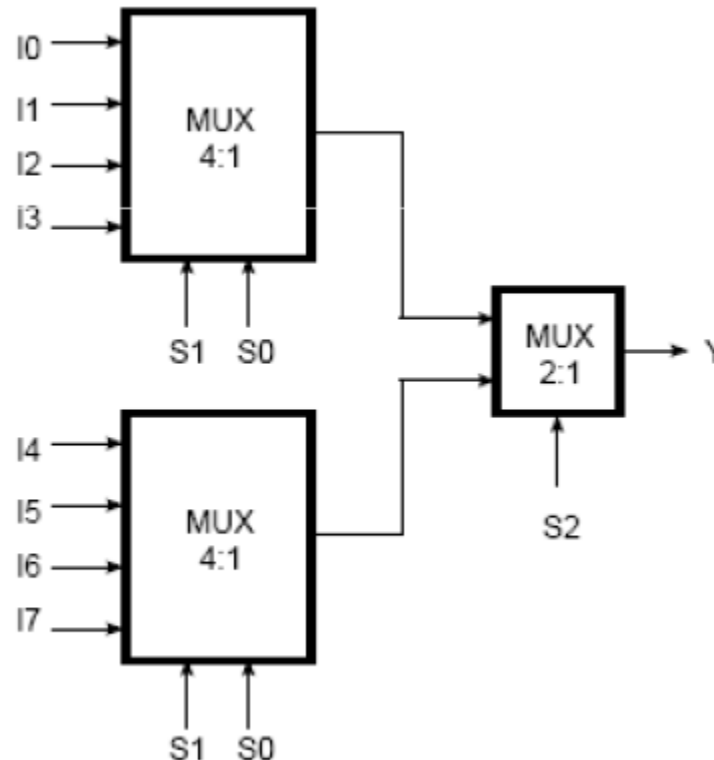


MULTIPLEXER

MULTIPLEXER EXPANSIONS

Example

A 8:1 Mux can be built by combining **two** 4:1 Mux and **one** 2:1 Mux.



MULTIPLEXER

MULTIPLEXER INTEGRATED CIRCUIT (IC)

- Mux (and other common logic blocks) can be bought as a packaged **integrated circuits (IC)**.
- Commonly used IC is **TTL** (Transistor-Transistor Logic) and **CMOS** (Complementary Metal-Oxide Semiconductor).

Example

An inverter IC in TTL is named 74LS04 (LS for Low Speed TTL). While inverter IC in CMOS is named 74HC04 (HC for High Speed CMOS).

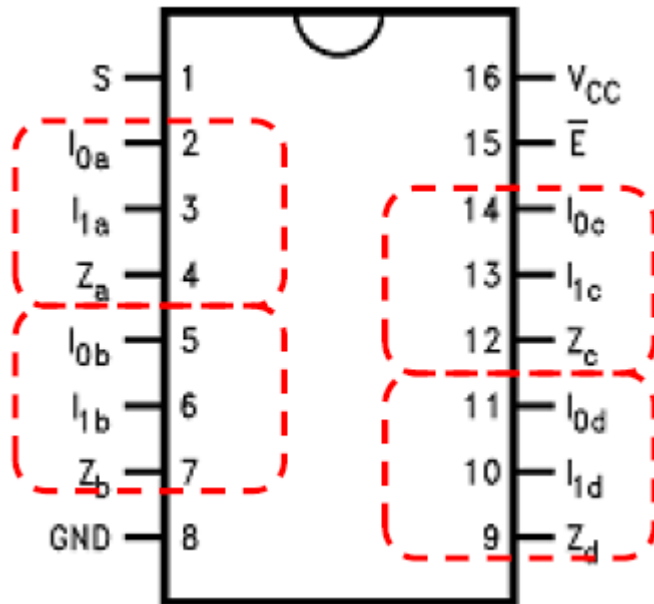
- 2:1 Mux IC: 74LS157/74HC157 (74x157)
- 4:1 Mux IC: 74LS153/74HC153 (74x153)
- 8:1 Mux IC: 74LS151/74HC151 (74x151)

MULTIPLEXER

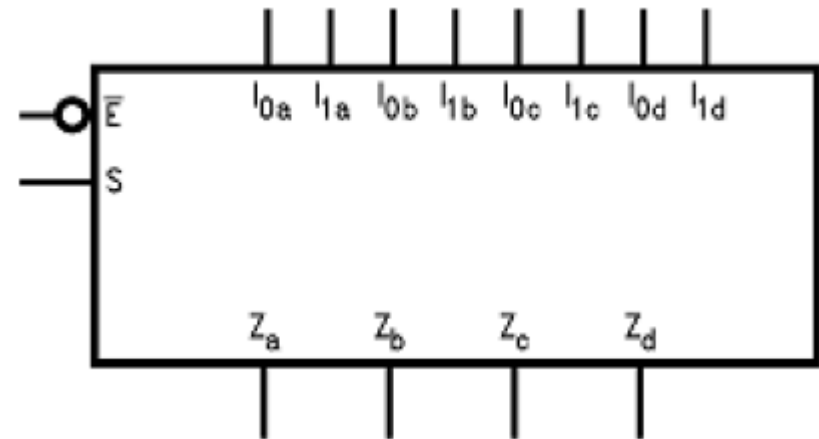
MULTIPLEXER IC: 74x157 (QUAD 2:1 MUX)

- **74x157** is a quad 2:1 Mux.
- Contains of **four** 2:1 Mux.
- Controlled by a **single common selector** input.
- It has one **active-low** enable input.

Connection Diagram



Logic Symbols



When $S = 0$ and $\bar{E} = 0$; $Z_a = I_{0a}$, $Z_b = I_{0b}$, $Z_c = I_{0c}$, $Z_d = I_{0d}$

MULTIPLEXER

MULTIPLEXER IC: 74x157 (QUAD 2:1 MUX)

Truth Table

Inputs				Output
\bar{E}	S	I_0	I_1	Z
H	X	X	X	L
L	H	X	L	L
L	H	X	H	H
L	L	L	X	L
L	L	H	X	H

H = HIGH Voltage Level

L = LOW Voltage Level

X = Immaterial (Irrelevant)

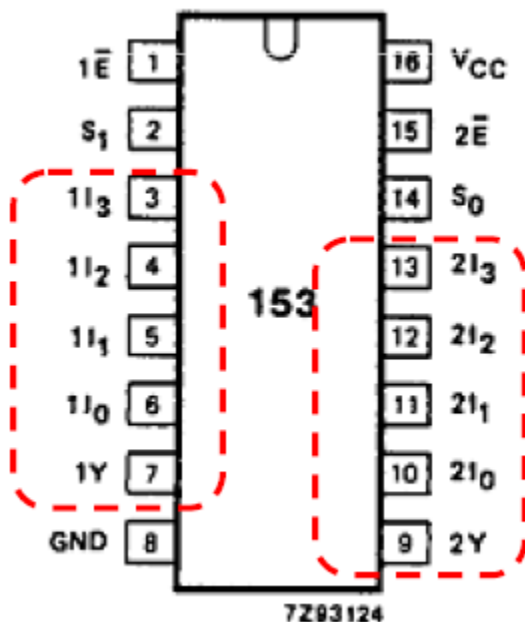
Output Z selects I_0 or I_1 depending on select S (with $E = 0$)

MULTIPLEXER

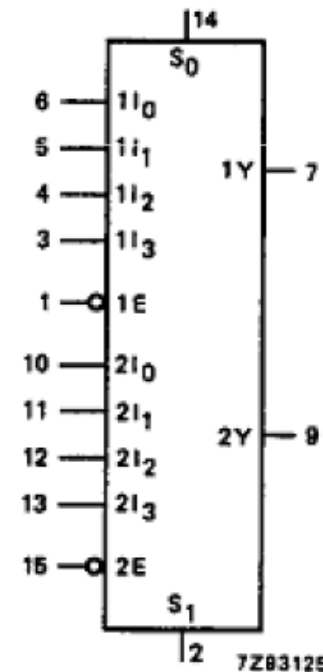
MULTIPLEXER IC: 74x153 (DUAL 4:1 MUX)

- **74x153** is a dual 4:1 Mux.
- Contains of **two** 4:1 Mux.
- Controlled by a **two common selector** input.
- It has two **active-low** enable input.

Connection Diagram



Logic Symbols



MULTIPLEXER

MULTIPLEXER IC: 74x153 (DUAL 4:1 MUX)

Function Table

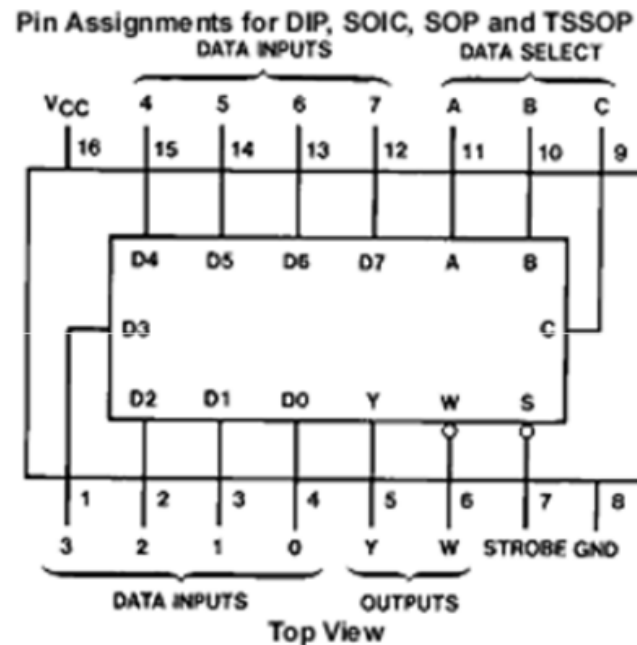
SELECT INPUTS		DATA INPUTS				OUTPUT ENABLE	OUTPUT
S_0	S_1	nl_0	nl_1	nl_2	nl_3	$n\bar{E}$	nY
X	X	X	X	X	X	H	L
L	L	L	X	X	X	L	L
L	L	H	X	X	X	L	H
H	L	X	L	X	X	L	L
H	L	X	H	X	X	L	H
L	H	X	X	L	X	L	L
L	H	X	X	H	X	L	H
H	H	X	X	X	L	L	L
H	H	X	X	X	H	L	H

Output nY selects nl_0 , nl_1 , nl_2 or nl_3 depending on S_1 and S_0 (with $n\bar{E} = 0$).

MULTIPLEXER

MULTIPLEXER IC: 74x151 (8:1 MUX)

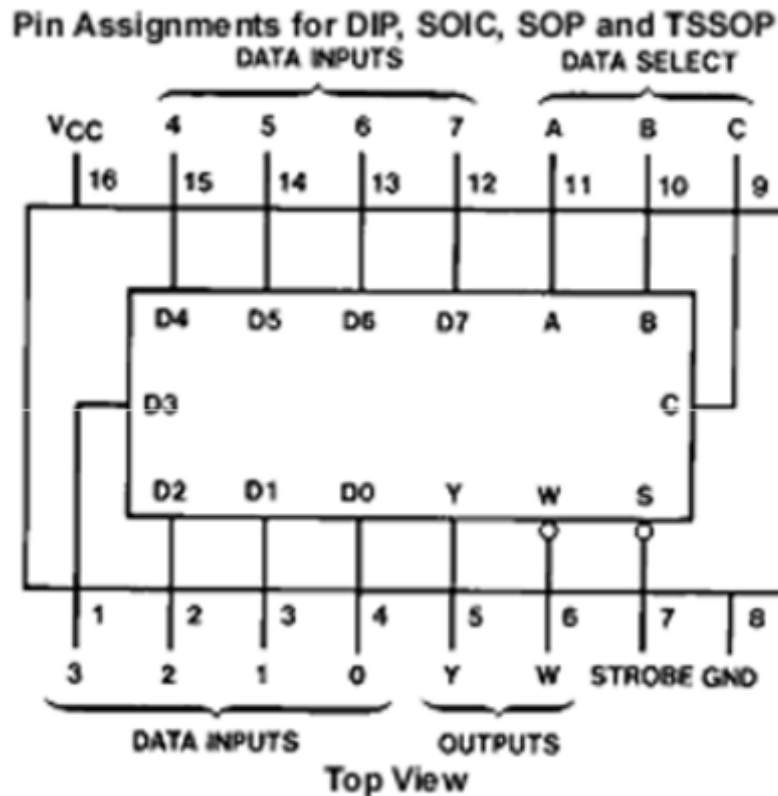
- **74x151** is a 8:1 Mux.
- Contains of **one** 8:1 Mux.
- It has two outputs
 1. Active High
 2. Active Low
- It has one **active-low** enable input.



MULTIPLEXER

MULTIPLEXER IC: 74x151 (8:1 MUX)

Connection Diagram



Truth Table

Inputs				Outputs	
Select			Strobe	Y	W
C	B	A	S		
X	X	X	H	L	H
L	L	L	L	D0	$\overline{D0}$
L	L	H	L	D1	$\overline{D1}$
L	H	L	L	D2	$\overline{D2}$
L	H	H	L	D3	$\overline{D3}$
H	L	L	L	D4	$\overline{D4}$
H	L	H	L	D5	$\overline{D5}$
H	H	L	L	D6	$\overline{D6}$
H	H	H	L	D7	$\overline{D7}$

MULTIPLEXER

REVIEWS

- How to design a 3:1 Mux or a 7:1 Mux?
 - 3:1 Mux is structured as 4:1 Mux
 - 7:1 Mux is structured as 8:1 Mux
- How many select bits is needed for 16:1 Mux?
 - 4 select inputs (S3, S2, S1, S0)
- How many inputs does a 32:1 Mux have?
 - 5 select bits and 32 input data lines (37 inputs)

DEMULTIPLEXER

DEMULTIPLEXER

INTRODUCTION

- **Demultiplexer (Demux)** perform in the inverse of the mux function.
- It takes data from one line and distribute to given number and of output lines.
- Basic Demultiplexer has **one input, n data-selector inputs and 2^n output.**
- Demultiplexer usually written as **$1x(Y)$** or **$1:(Y)$** , where Y is the number of output data lines.

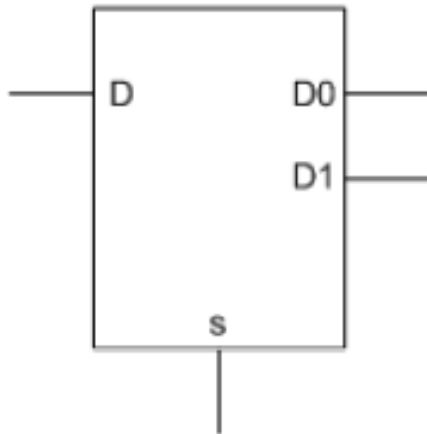
Example

A 4 output data line demultiplexer is written as **$1:4$ demux.**

DEMULTIPLEXER

1:2 DEMUX

- **1:2 Demux** consists of **1 input, 1 selector and 2 outputs.**



Symbol for 1:2 Demux

Function:

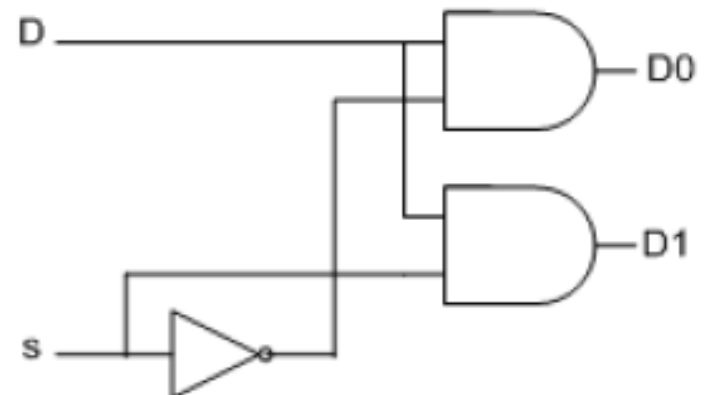
$D0 = D, D1 = 0$ when $S = 0$ $\longrightarrow D0 = D \cdot \bar{S}$

$D0 = 0, D1 = D$ when $S = 1$ $\longrightarrow D1 = D \cdot S$

Function Table

S	$D1$	$D0$
0	0	D
1	D	0

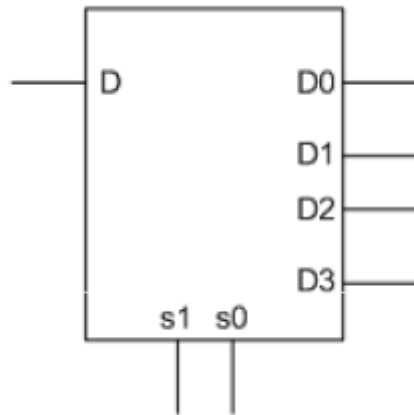
Logic circuit



DEMULTIPLEXER

1:4 DEMUX

- **1:4 Demux** consists of **1 input, 2 selectors and 4 outputs.**



Symbol for 1:4 Demux

Function:

$$D0 = D \text{ when } S1 = 0 \text{ and } S0 = 0 \rightarrow D0 = D \cdot \overline{S1} \cdot \overline{S0}$$

$$D1 = D \text{ when } S1 = 0 \text{ and } S0 = 1 \rightarrow D1 = D \cdot \overline{S1} \cdot S0$$

$$D2 = D \text{ when } S1 = 1 \text{ and } S0 = 0 \rightarrow D2 = D \cdot S1 \cdot \overline{S0}$$

$$D3 = D \text{ when } S1 = 1 \text{ and } S0 = 1 \rightarrow D3 = D \cdot S1 \cdot S0$$

Function Table

$S1$	$S0$	$D3$	$D2$	$D1$	$D0$
0	0	0	0	0	D
0	1	0	0	D	0
1	0	0	D	0	0
1	1	D	0	0	0



DECODER

DECODER

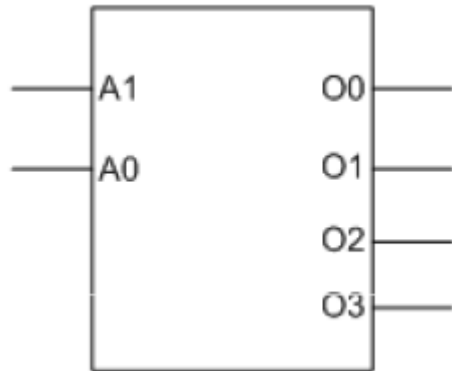
INTRODUCTION

- **Decoder** used to detect the presence of specified combination of bits (code) on its inputs and indicates the presence of that code by a specific output level.
- It has **n inputs and 2^n outputs (n -to- 2^n)**.
- Decoder can be designed as 1-to-2, 2-to-4, 3-to-8, 4-to-16 and etc.
- If **enable inputs** is presents, it must be asserted to enable decoder function.

DECODER

2-to-4 DECODER

- **2-to-4 Decoder** consists of **2 inputs and 4 outputs**.



Symbol for **Active High** 2-to-4 Decoder

Function:

$$O_3 O_2 O_1 O_0 = 0001 \text{ when } A_1 A_0 = 00 \rightarrow O_0 = \overline{A_1} \cdot \overline{A_0}$$

$$O_3 O_2 O_1 O_0 = 0010 \text{ when } A_1 A_0 = 01 \rightarrow O_1 = \overline{A_1} \cdot A_0$$

$$O_3 O_2 O_1 O_0 = 0100 \text{ when } A_1 A_0 = 10 \rightarrow O_2 = A_1 \cdot \overline{A_0}$$

$$O_3 O_2 O_1 O_0 = 1000 \text{ when } A_1 A_0 = 11 \rightarrow O_3 = A_1 \cdot A_0$$

Function Table

Inputs		Outputs			
A1	A0	O ₃	O ₂	O ₁	O 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

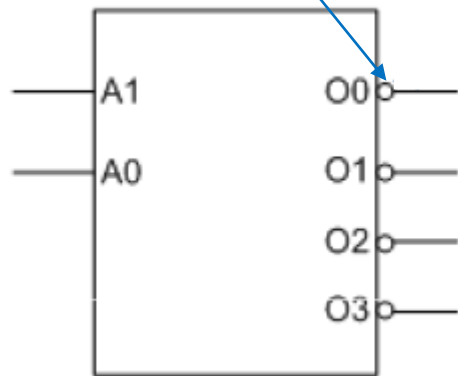
DECODER

2-to-4 DECODER

- Typically, decoders are designed as **Active Low**.

Bubble at output denotes

active low output



Function:

$$O_3 O_2 O_1 O_0 = 1110 \text{ when } A_1 A_0 = 00 \rightarrow O_0 = \overline{\overline{A_1} \cdot \overline{A_0}}$$

$$O_3 O_2 O_1 O_0 = 1101 \text{ when } A_1 A_0 = 01 \rightarrow O_1 = \overline{\overline{A_1} \cdot A_0}$$

$$O_3 O_2 O_1 O_0 = 1011 \text{ when } A_1 A_0 = 10 \rightarrow O_2 = \overline{A_1 \cdot \overline{A_0}}$$

$$O_3 O_2 O_1 O_0 = 0111 \text{ when } A_1 A_0 = 11 \rightarrow O_3 = \overline{A_1 \cdot A_0}$$

Symbol for **Active Low**
2-to-4 Decoder

Function Table

Inputs		Outputs			
A1	A0	O ₃	O ₂	O ₁	O ₀
0	0	1	1	1	0
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	1	1	1

DECODER

2-to-4 DECODER

- **2-to-4 Decoder with Active Low Enable.**

Function:

If ($E = 1$),

$O_3 O_2 O_1 O_0 = 1111, A_1 A_0 = xx$

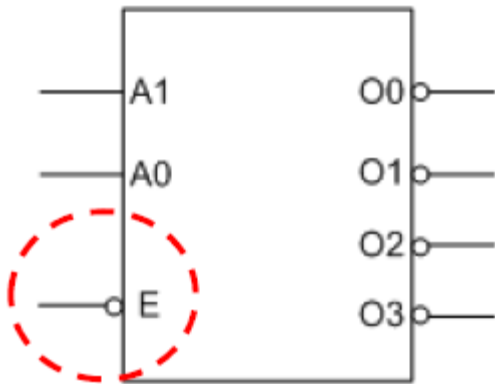
If ($E = 0$),

$O_3 O_2 O_1 O_0 = 1110$ when $A_1 A_0 = 00 \rightarrow O_0 = \overline{\overline{E} \cdot \overline{A_1} \cdot \overline{A_0}}$

$O_3 O_2 O_1 O_0 = 1101$ when $A_1 A_0 = 01 \rightarrow O_1 = \overline{\overline{E} \cdot \overline{A_1} \cdot A_0}$

$O_3 O_2 O_1 O_0 = 1011$ when $A_1 A_0 = 10 \rightarrow O_2 = \overline{\overline{E} \cdot A_1 \cdot \overline{A_0}}$

$O_3 O_2 O_1 O_0 = 0111$ when $A_1 A_0 = 11 \rightarrow O_3 = \overline{\overline{E} \cdot A_1 \cdot A_0}$



Symbol for **Active Low**

2-to-4 Decoder with
Active Low Enable

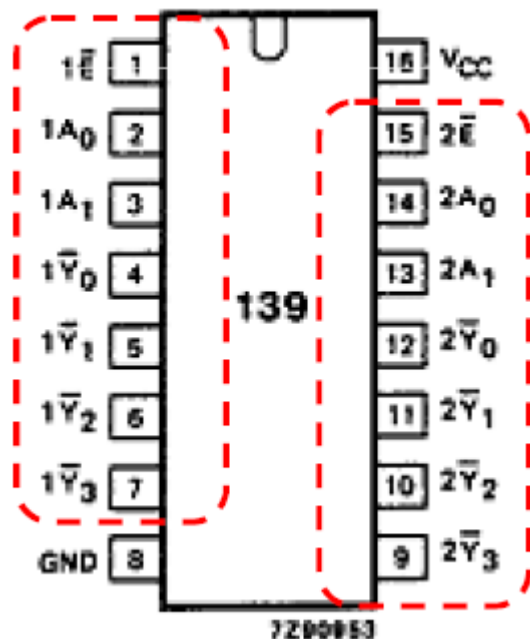
Function Table

E	Inputs		Outputs			
	A1	A0	O_3	O_2	O_1	O_0
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

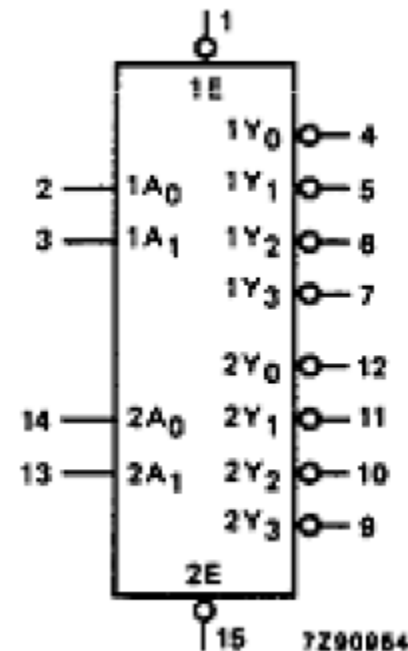
DECODER

DECODER IC: 74x139 (2-to-4 DECODER)

- **74x139** contains **two 2-to-4 Decoders**.
- To use either decoder, it must be enabled by inputting low signal at the enable input.
- When **enable = High**, **all output = High**.



Pin Configuration



Logic Symbol

DECODER

DECODER IC: 74x139 (2-to-4 DECODER)

Function Table

Inputs			Outputs			
$n\bar{E}$	nA_0	nA_1	$n\bar{Y}_0$	$n\bar{Y}_1$	$n\bar{Y}_2$	$n\bar{Y}_3$
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	H	L	H	L	H	H
L	L	H	H	H	L	H
L	H	H	H	H	H	L

H = HIGH Voltage Level

L = LOW Voltage Level

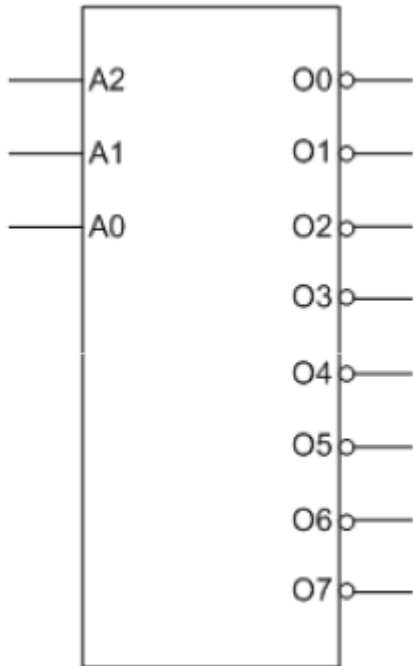
X = Don't care

Outputs depends on inputs A with $E = 0$

DECODER

3-to-8 Decoder

- **3-to-8 Decoder** consists of **3 inputs and 8 outputs**.



Function:

$$O_7 O_6 O_5 O_4 O_3 O_2 O_1 O_0 = 1111\ 1110 \text{ when } A_2 A_1 A_0 = 000$$

$$O_7 O_6 O_5 O_4 O_3 O_2 O_1 O_0 = 1111\ 1101 \text{ when } A_2 A_1 A_0 = 001$$

$$O_7 O_6 O_5 O_4 O_3 O_2 O_1 O_0 = 1111\ 1011 \text{ when } A_2 A_1 A_0 = 010$$

$$O_7 O_6 O_5 O_4 O_3 O_2 O_1 O_0 = 1111\ 0111 \text{ when } A_2 A_1 A_0 = 011$$

$$O_7 O_6 O_5 O_4 O_3 O_2 O_1 O_0 = 1110\ 1111 \text{ when } A_2 A_1 A_0 = 100$$

$$O_7 O_6 O_5 O_4 O_3 O_2 O_1 O_0 = 1101\ 1111 \text{ when } A_2 A_1 A_0 = 101$$

$$O_7 O_6 O_5 O_4 O_3 O_2 O_1 O_0 = 1011\ 1111 \text{ when } A_2 A_1 A_0 = 110$$

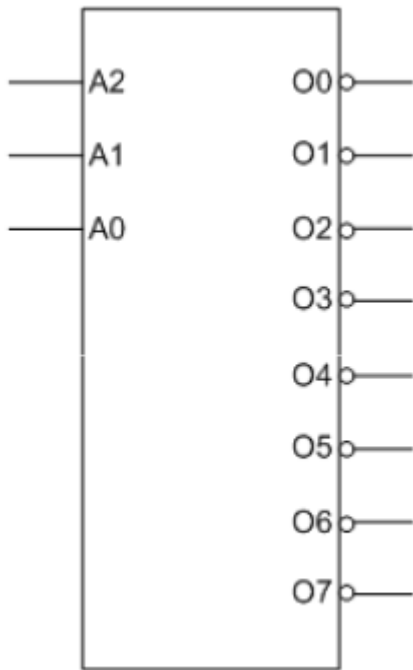
$$O_7 O_6 O_5 O_4 O_3 O_2 O_1 O_0 = 0111\ 1111 \text{ when } A_2 A_1 A_0 = 111$$

Symbol for **Active Low**
3-to-8 Decoder

DECODER

3-to-8 DECODER

- **3-to-8 Decoder** consists of **3 inputs and 8 outputs**.



Symbol for **Active Low**
3-to-8 Decoder

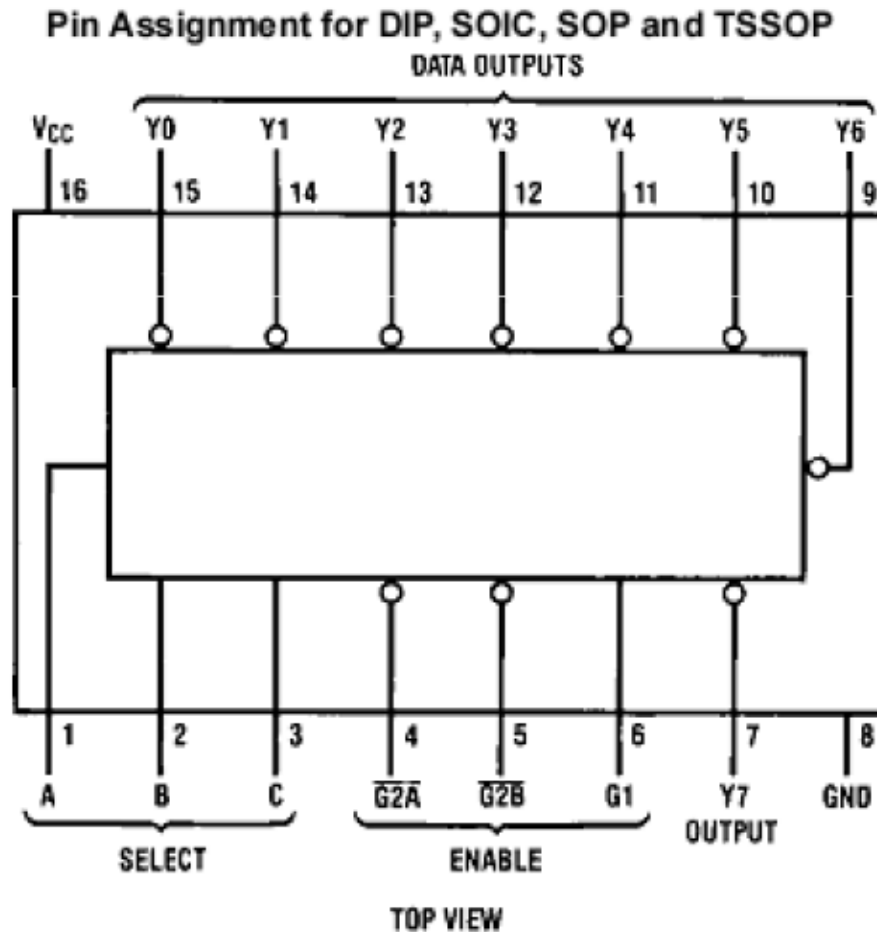
Function Table

Inputs			Outputs							
A2	A1	A0	O7	O6	O5	O4	O3	O2	O1	O0
0	0	0	1	1	1	1	1	1	1	0
0	0	1	1	1	1	1	1	1	0	1
0	1	0	1	1	1	1	1	0	1	1
0	1	1	1	1	1	1	0	1	1	1
1	0	0	1	1	1	0	1	1	1	1
1	0	1	1	1	0	1	1	1	1	1
1	1	0	1	0	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1

DECODER

DECODER IC: 74x138 (3-to-8 DECODER)

- **74x138** contains **one 3-to-8 Decoder**. (a popular device)
- It has **3 inputs, 3 enables and 8 outputs**.



DECODER

DECODER IC: 74x138 (3-to-8 DECODER)

Function Table

Inputs					Outputs							
Enable		Select										
$G1$	$\overline{G2}$	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

H = HIGH Voltage Level, L= LOW Voltage Level, X = Don't care

$$\overline{G2} = \overline{G2A} + \overline{G2B}$$

DECODER

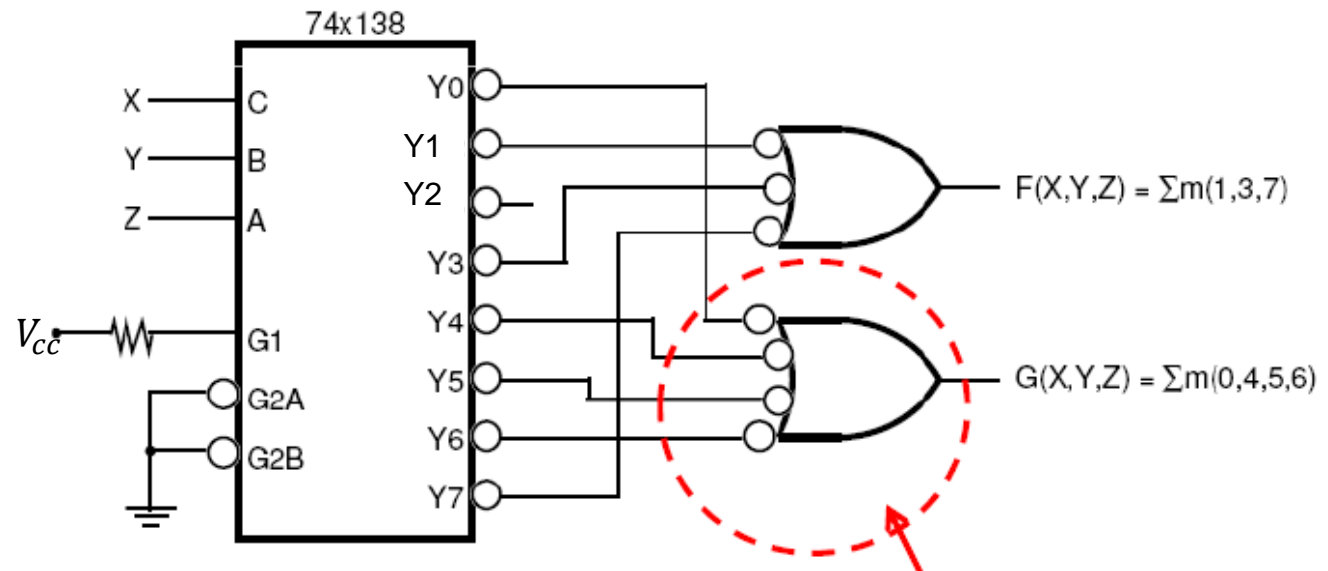
DECODER APPLICATIONS

Example

Show how the 3-to-8 Decoder and basic gate can implement the logic function $F(X, Y, Z) = \sum m(1, 3, 7)$ and $G(X, Y, Z) = \sum m(0, 4, 5, 6)$

Truth Table

Inputs			Output	
X	Y	Z	F	G
0	0	0	0	1
0	0	1	1	0
0	1	0	0	0
0	1	1	1	0
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	0



Also represents as
AND gate

BOOLEAN FUNCTION USING MULTIPLIER & DECODER

BOOLEAN FUNCTION USING MULTIPLEXER & DECODER

Example

Given $F(A, B, C) = \sum m(1, 2, 4, 5)$. Implement using;

- a) 8:1 Mux
- b) 4:1 Mux
- c) 2:1 Mux
- d) 3-to-8 Active Low Decoder

BOOLEAN FUNCTION USING MULTIPLEXER & DECODER

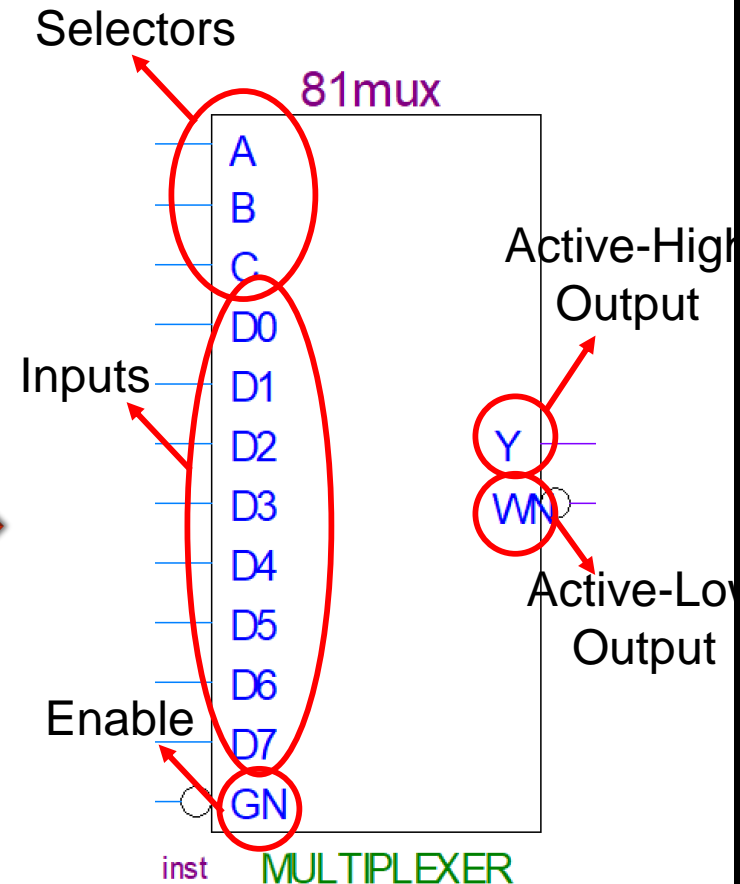
Solution

Using 8:1 Mux

$$F(A, B, C) = \sum m(1, 2, 4, 5)$$

Truth Table

Inputs			Output
A	B	C	F
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	



BOOLEAN FUNCTION USING MULTIPLEXER & DECODER

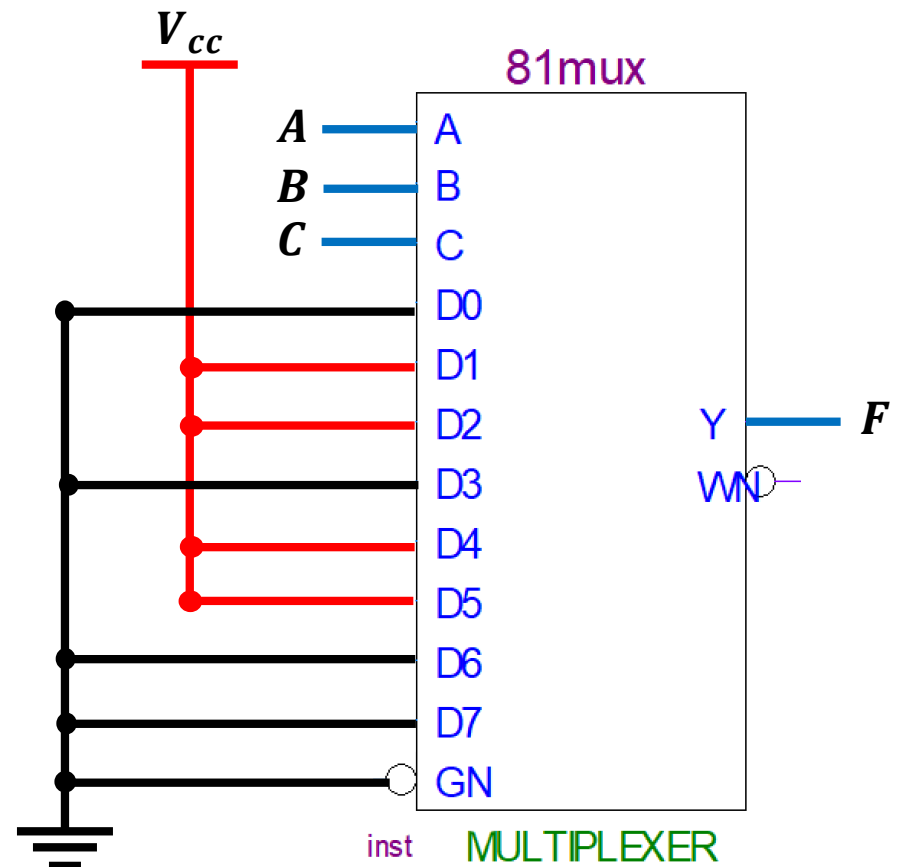
Solution

Using 8:1 Mux

$$F(A, B, C) = \sum m(1, 2, 4, 5)$$

Truth Table

Inputs			Output
A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0



BOOLEAN FUNCTION USING MULTIPLEXER & DECODER

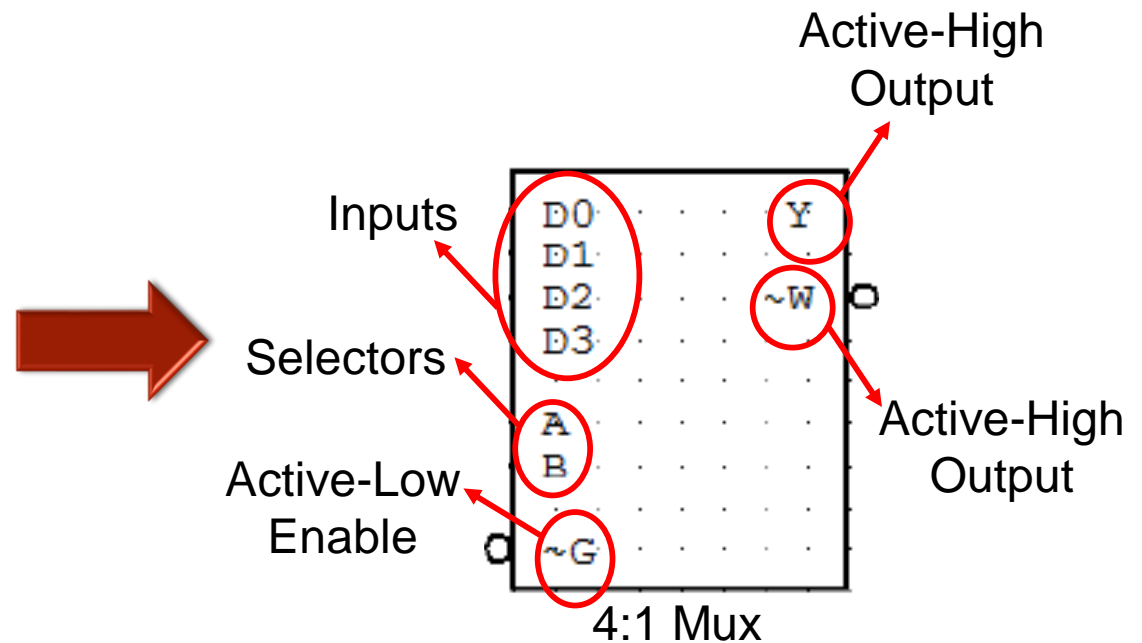
Solution

Using 4:1 Mux

$$F(A, B, C) = \sum m(1, 2, 4, 5)$$

Truth Table

Inputs			Output	
A	B	C	F	
0	0	0	0	$F = C$
0	0	1	1	
0	1	0	1	$F = \bar{C}$
0	1	1	0	
1	0	0	1	$F = 1$
1	0	1	1	
1	1	0	0	$F = 0$
1	1	1	0	



BOOLEAN FUNCTION USING MULTIPLEXER & DECODER

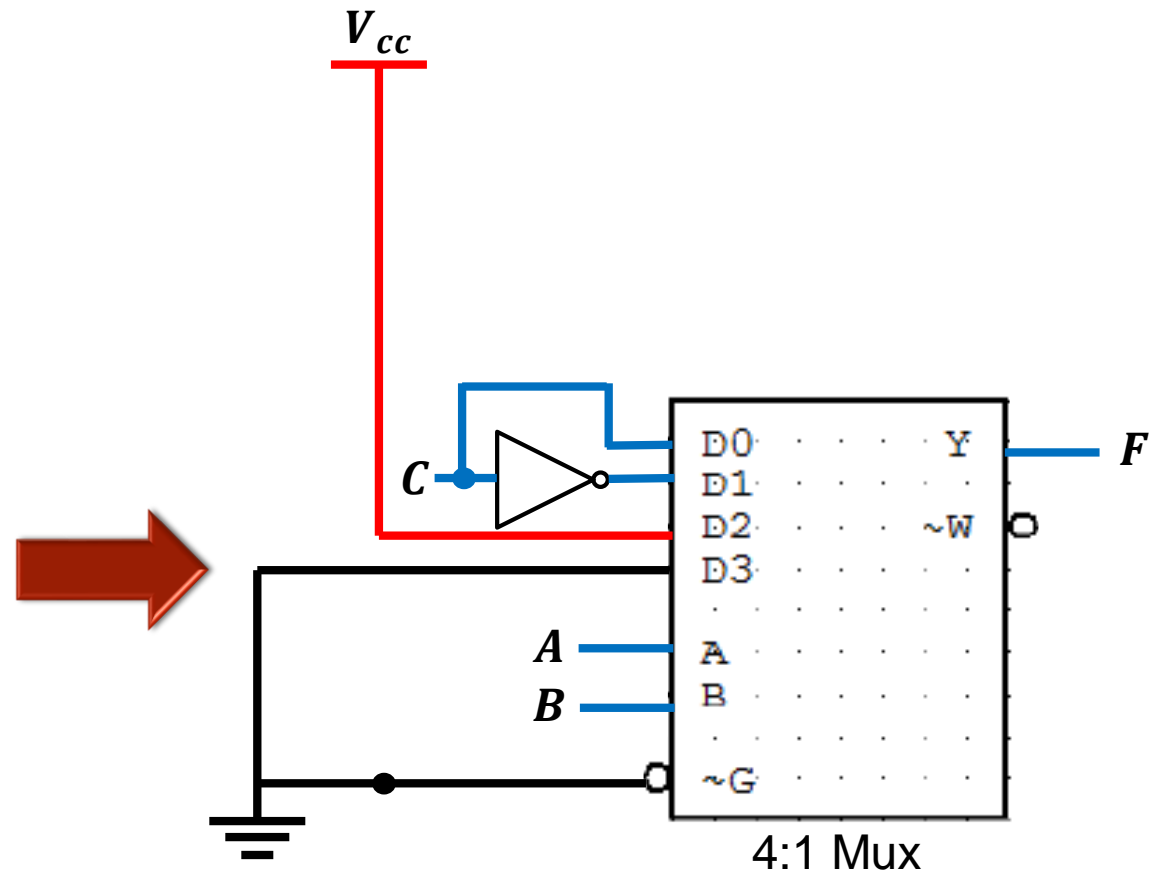
Solution

Using 4:1 Mux

$$F(A, B, C) = \sum m(1, 2, 4, 5)$$

Truth Table

Inputs			Output
A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0



BOOLEAN FUNCTION USING MULTIPLEXER & DECODER

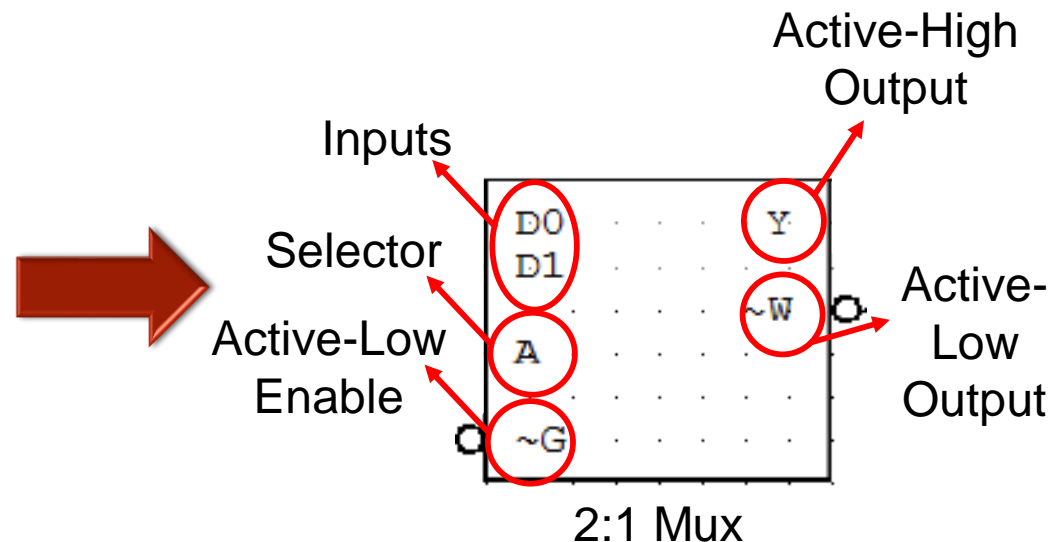
Solution

Using 2:1 Mux

$$F(A, B, C) = \sum m(1, 2, 4, 5)$$

Truth Table

Inputs			Output	
A	B	C	F	
0	0	0	0	$F = \bar{B}C + B\bar{C}$
0	0	1	1	
0	1	0	1	
0	1	1	0	
1	0	0	1	$F = \bar{B}$
1	0	1	1	
1	1	0	0	
1	1	1	0	



BOOLEAN FUNCTION USING MULTIPLEXER & DECODER

Solution

Using 2:1 Mux

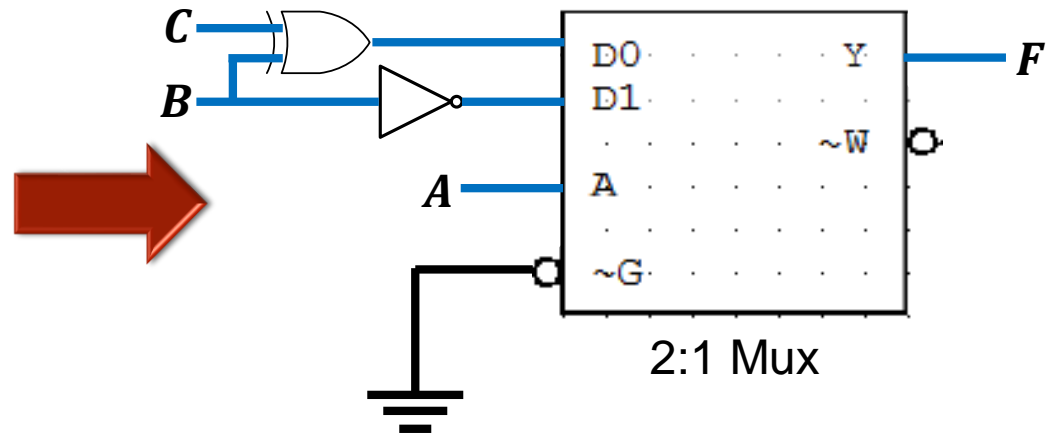
$$F(A, B, C) = \sum m(1, 2, 4, 5)$$

Truth Table

Inputs			Output
A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

$$F = \bar{B}C + B\bar{C}$$

$$F = \bar{B}$$

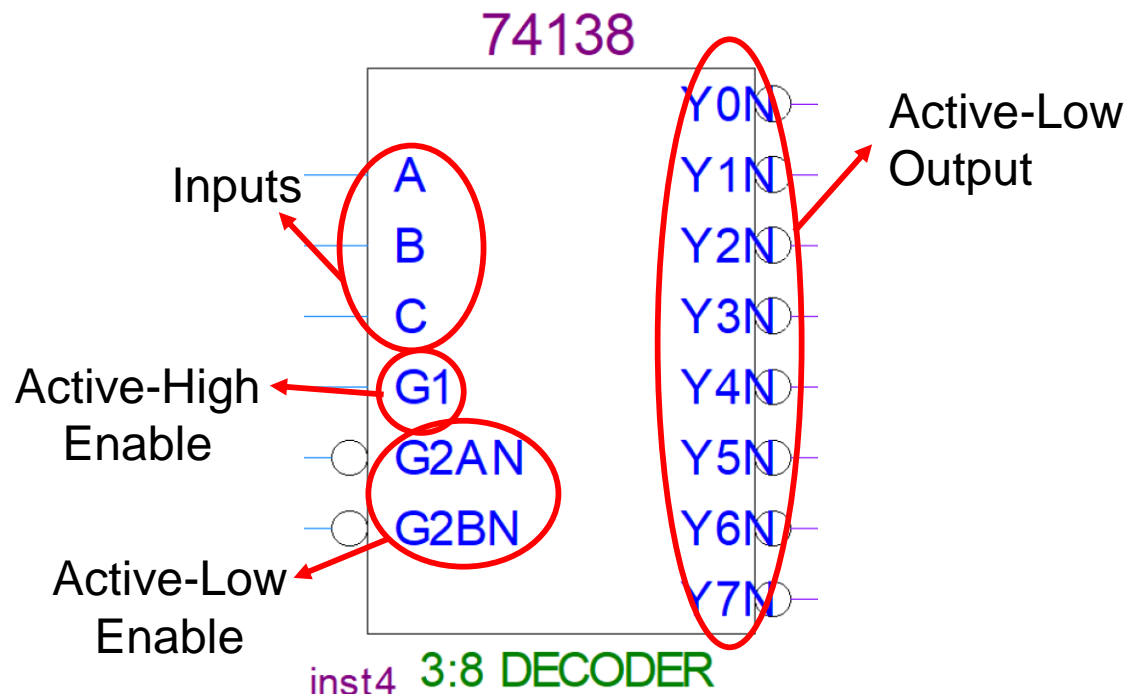


BOOLEAN FUNCTION USING MULTIPLEXER & DECODER

Solution

Using 3-to-8 Decoder

$$F(A, B, C) = \sum m(1, 2, 4, 5)$$

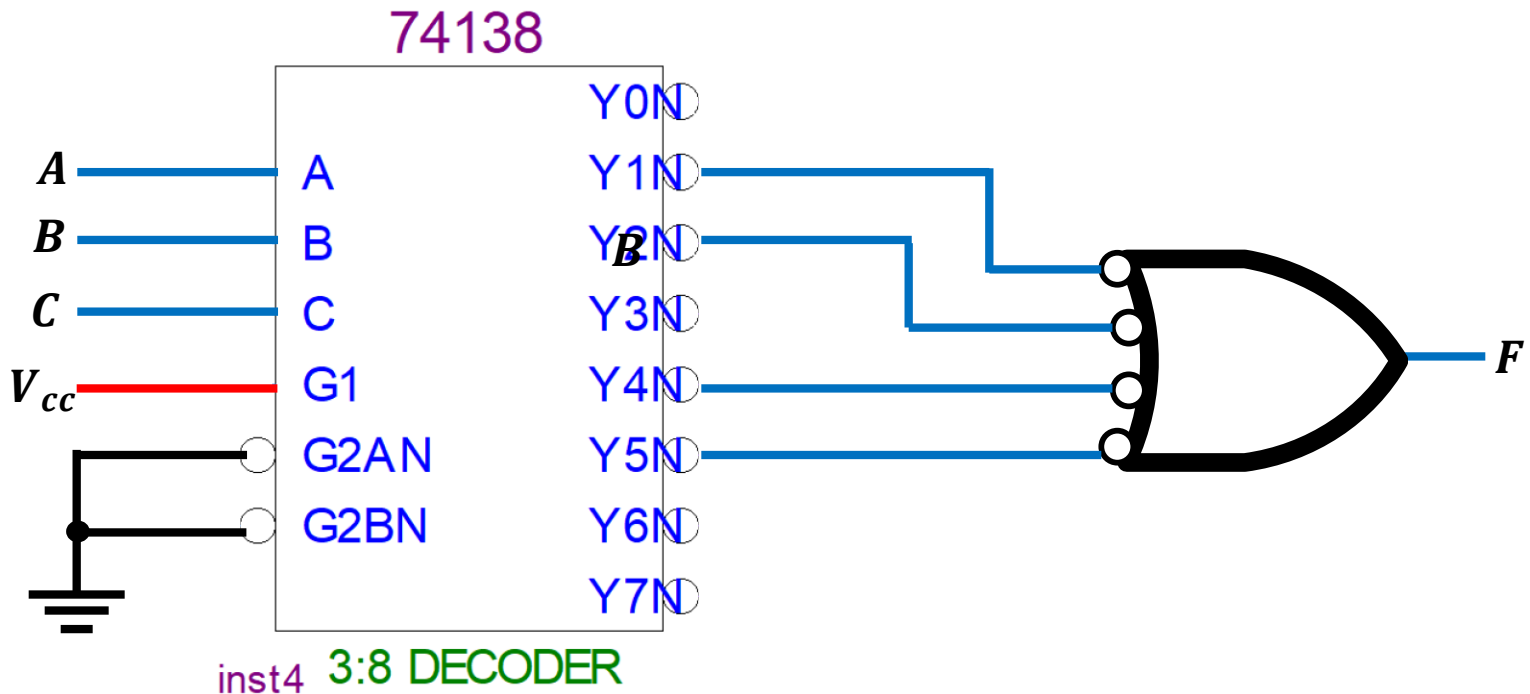


BOOLEAN FUNCTION USING MULTIPLEXER & DECODER

Solution

Using 3-to-8 Decoder

$$F(A, B, C) = \sum m(1, 2, 4, 5)$$



BOOLEAN FUNCTION USING MULTIPLEXER & DECODER

ASSESSMENT 1

Given Boolean expression $A \cdot \bar{C} + \bar{A} \cdot \bar{B} \cdot \bar{C} + \bar{A} \cdot \bar{B} \cdot C + \bar{A} \cdot (B \oplus C)$,
implement using;

- a) 8:1 Mux
- b) 4:1 Mux
- c) 2:1 Mux
- d) 3-to-8 Active Low Decoder

BOOLEAN FUNCTION USING MULTIPLEXER & DECODER

ASSESSMENT 2

Given Boolean expression $f(X, Y, Z) = \prod(M_0, M_1, M_2, M_4)$,
implement using;

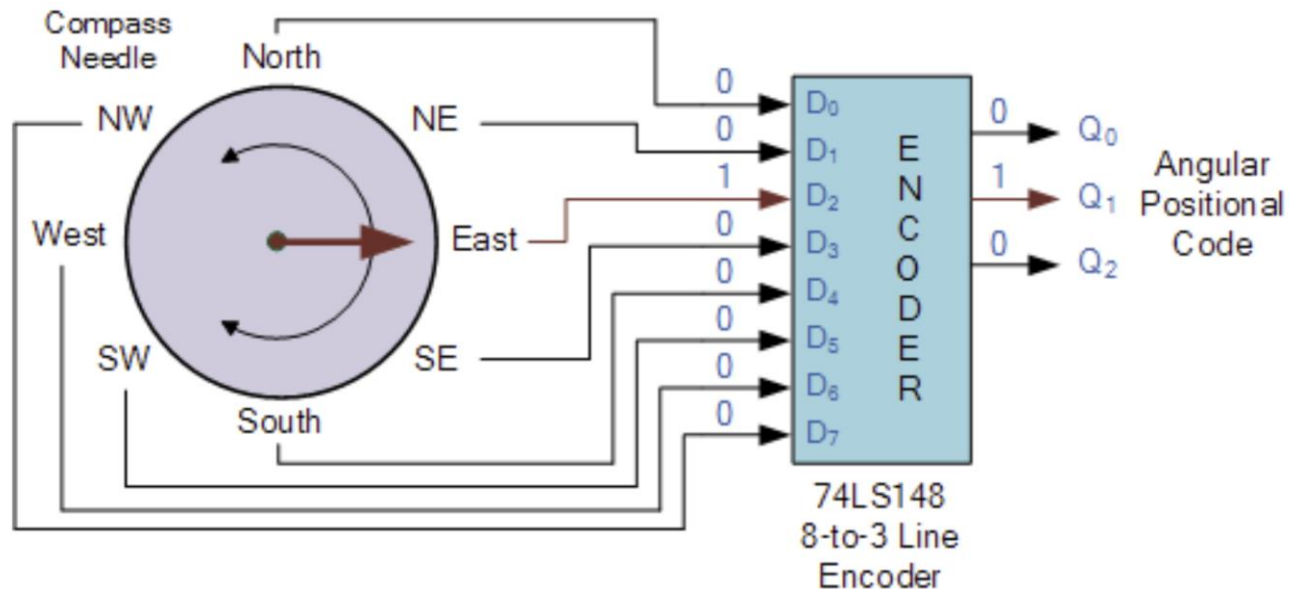
- a) 4:1 Mux
- b) 2:1 Mux
- c) 3-to-8 Active Low Decoder
- d) 2-to-4 Active Low Decoder

ENCODER

ENCODER

INTRODUCTION

- **Encoder** performs reverse function of decoder.
- **Encoder** used to compress the input into a code that contains the same information in fewer bits.
- It has **2^n inputs and n output. (2^n - to - n).**
- Only **one input** is allowed to be active at any one time.

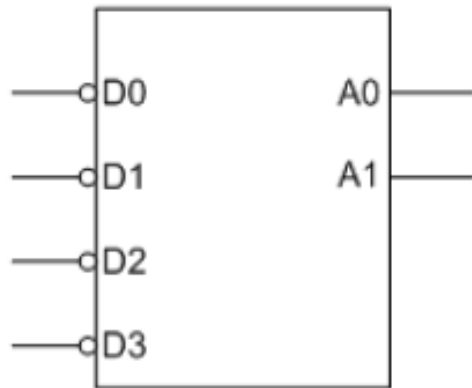


Encoder function for Compass (https://www.electronics-tutorials.ws/combination/comb_4.html)

ENCODER

4-to-2 ENCODER

- **4-to-2 Encoder** consists of **4 inputs and 2 outputs**.



Function:

$$A_1A_0 = 00 \text{ when } D_3D_2D_1D_0 = 1110$$

$$A_1A_0 = 01 \text{ when } D_3D_2D_1D_0 = 1101$$

$$A_1A_0 = 10 \text{ when } D_3D_2D_1D_0 = 1011$$

$$A_1A_0 = 11 \text{ when } D_3D_2D_1D_0 = 0111$$

Which implies:

$$A_1 = D_3\overline{D_2}D_1D_0 + \overline{D_3}D_2D_1D_0$$

$$A_0 = D_3D_2\overline{D_1}D_0 + \overline{D_3}D_2D_1D_0$$

4-to-2 **Active Low**
Encoder

**What happens if more than 1 input is '0' ($D_1 = 0$ and $D_2 = 0$)?

$$A_1A_0 = 11 \text{ **ERROR**}$$

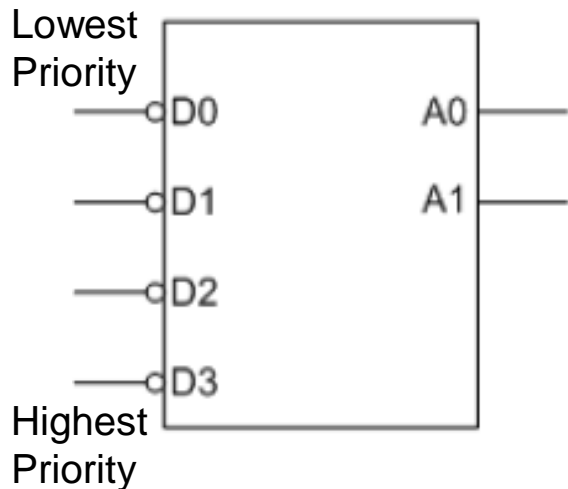
We need a **Priority Encoder**

Function Table	Inputs				Outputs	
	D_3	D_2	D_1	D_0	A_1	A_0
	1	1	1	0	0	0
	1	1	0	1	0	1
	1	0	1	1	1	0
	0	1	1	1	1	1

ENCODER

PRIORITY ENCODER

- Priority Encoder:** Outputs depends on largest active input.



Function:

$$A_1A_0 = 00 \text{ when } D_3D_2D_1D_0 = 1110$$

$$A_1A_0 = 01 \text{ when } D_3D_2D_1D_0 = 110X$$

$$A_1A_0 = 10 \text{ when } D_3D_2D_1D_0 = 10XX$$

$$A_1A_0 = 11 \text{ when } D_3D_2D_1D_0 = 0XXX$$

Which implies:

$$A_1 = D_3\overline{D_2}D_1D_0 + \overline{D_3}D_2D_1D_0$$

$$A_0 = D_3D_2\overline{D_1}D_0 + \overline{D_3}D_2D_1D_0$$

4-to-2 **Active Low** Priority Encoder

**What happens if more than 1 input is '0' ($D_0 = 0$ and $D_1 = 0$)?

Output $A_1A_0 = 01$

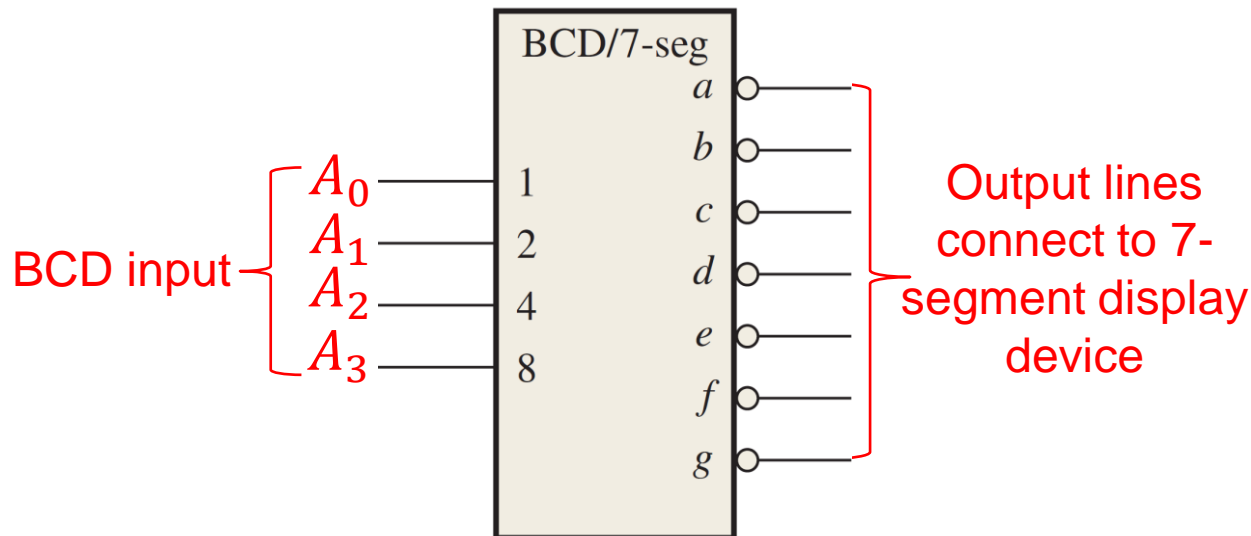
Function Table	Inputs				Outputs	
	D_3	D_2	D_1	D_0	A_1	A_0
	1	1	1	0	0	0
	1	1	0	X	0	1
	1	0	X	X	1	0
	0	X	X	X	1	1

BCD TO 7 SEGMENT DISPLAY DECODER

BCD TO 7 SEGMENT DECODER

INTRODUCTION

- **BCD to 7 segment decoder** accept **BCD codes** on it inputs, and provides outputs to drive **7-segment display** to produce decimal read out.

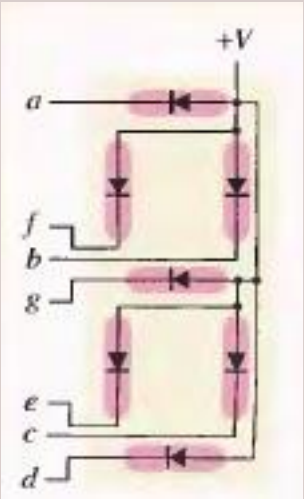
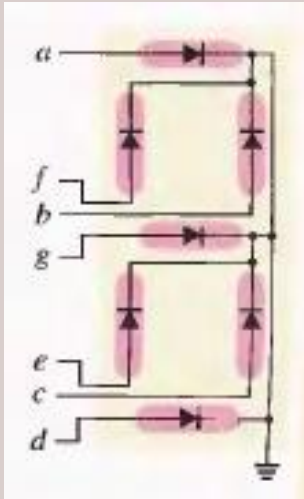


Logic symbol for BCD to 7-segment decoder with **Active-Low** output

BCD TO 7 SEGMENT DECODER

INTRODUCTION

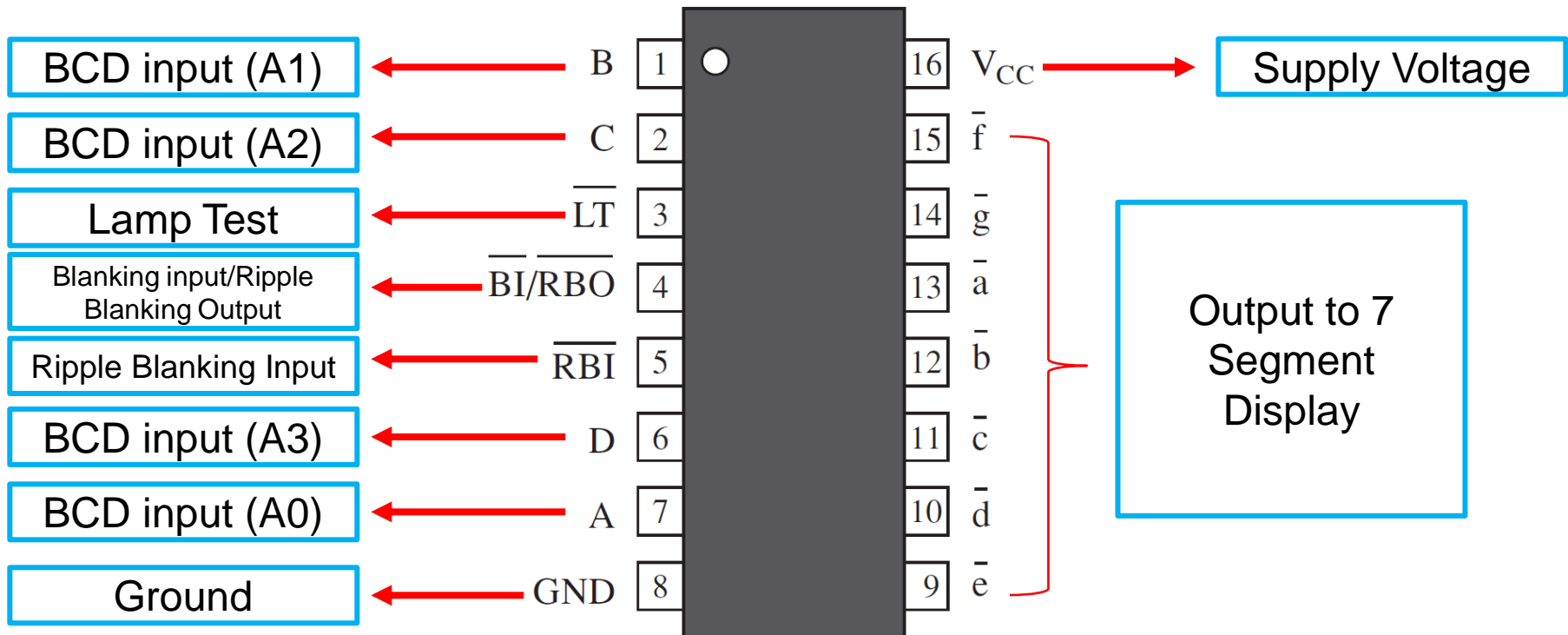
- Two type of 7-segment display;
 1. **Common Anode.**
 2. **Common Cathode.**

Common Anode	Common Cathode
	
<p>LED ON when LOW input is applied</p>	<p>LED ON when HIGH input is applied</p>

BCD TO 7 SEGMENT DECODER

INTRODUCTION: 74x47 IC

- The **74x47** is example of IC device that decodes a BCD input and drives the 7-segment display.
- The **74x47** is a **common anode** displays.

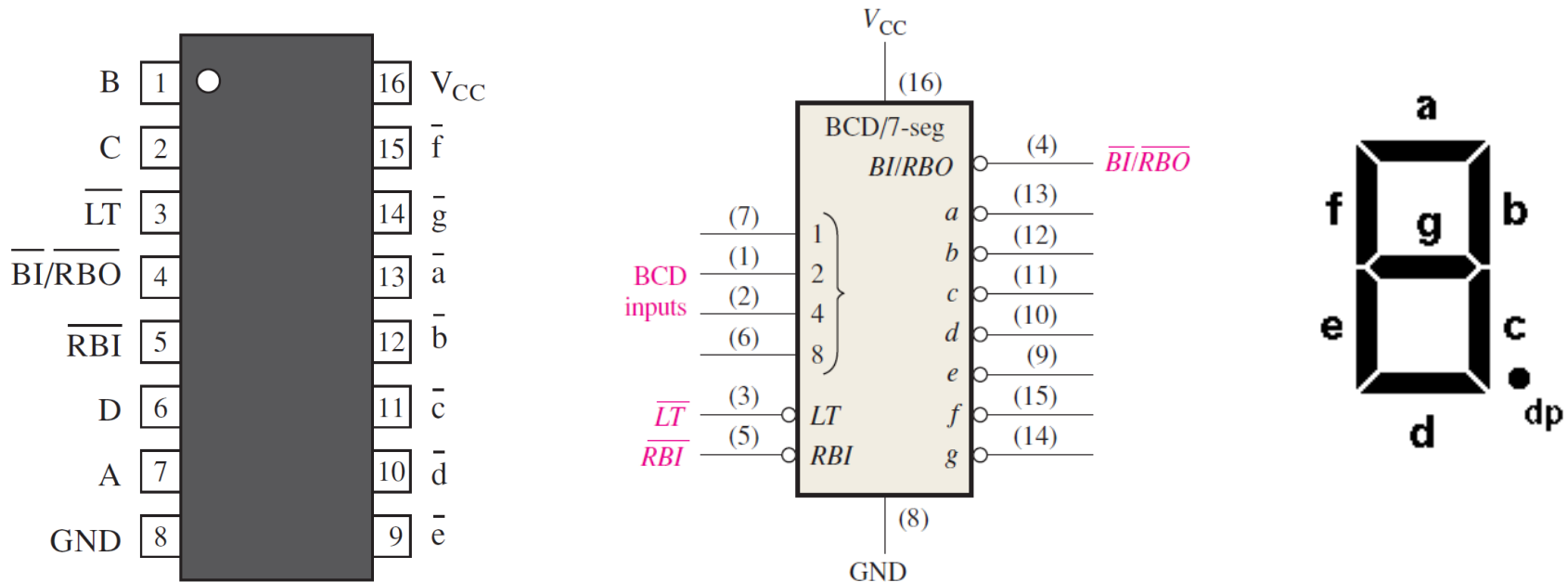


Connection Diagram

BCD TO 7 SEGMENT DECODER

INTRODUCTION: 74x47 IC

- The **74x47** is example of IC device that decodes a BCD input and drives the 7-segment display.
- The **74x47** is a **common anode** displays.



Connection Diagram

Logic Symbol

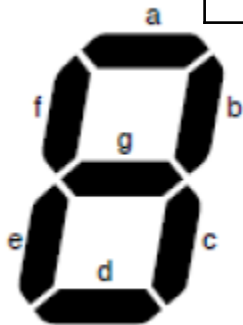
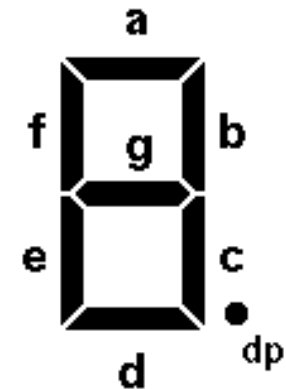
- Function: Converts 4-bit BCD (A3, A2, A1, A0) to 7-segment LED (a, b, c, d, e, f, g)

BCD TO 7 SEGMENT DECODER

INTRODUCTION: 74x47 IC

Truth Table

Decimal Value	BCD Code	7- Segment Display Code						
		a	b	c	d	e	f	g
0	0000	0	0	0	0	0	0	1
1	0001	1	0	0	1	1	1	1
2	0010	0	0	1	0	0	1	0
3	0011	0	0	0	0	1	1	0
4	0100	1	0	0	1	1	0	0
5	0101	0	1	0	0	1	0	0
6	0110	0	1	0	0	0	0	0
7	0111	0	0	0	1	1	1	1
8	1000	0	0	0	0	0	0	0
9	1001	0	0	0	1	1	0	0



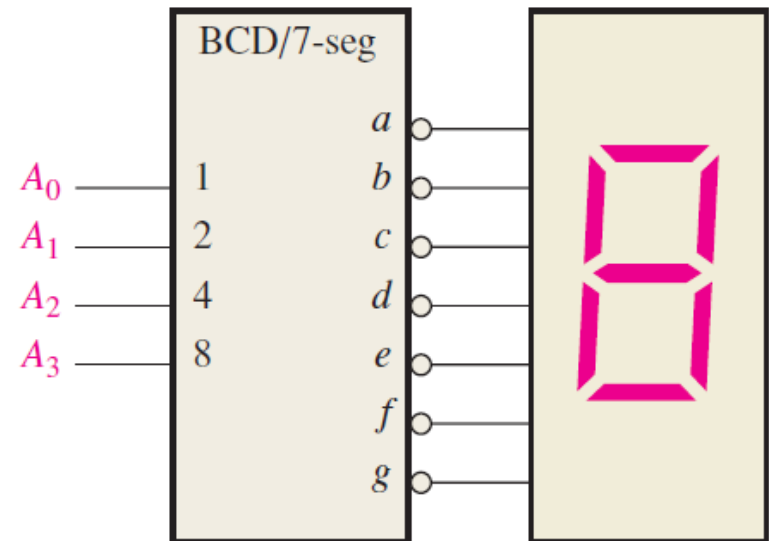
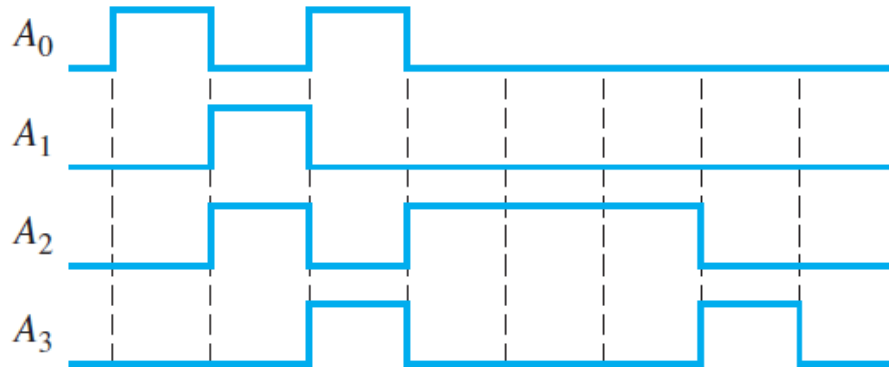
0 1234
56789

Segments turn on and off to display different numbers

BCD TO 7 SEGMENT DECODER

EXAMPLE

A 7-segment decoder drives the display as figure below. If waveforms are applied as indicated, determine the sequence of digits that appears on display.

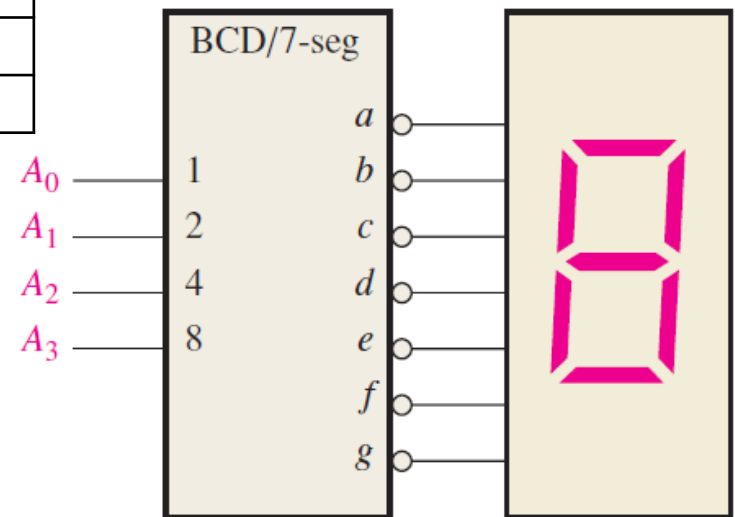
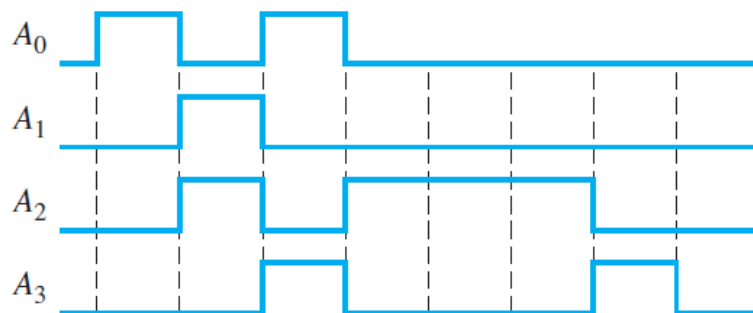


BCD TO 7 SEGMENT DECODER

EXAMPLE

Decimal Value	BCD Code	7- Segment Display Code						
		a	b	c	d	e	f	g
0	0000	0	0	0	0	0	0	1
1	0001	1	0	0	1	1	1	1
2	0010	0	0	1	0	0	1	0
3	0011	0	0	0	0	1	1	0
4	0100	1	0	0	1	1	0	0
5	0101	0	1	0	0	1	0	0
6	0110	0	1	0	0	0	0	0
7	0111	0	0	0	1	1	1	1
8	1000	0	0	0	0	0	0	0
9	1001	0	0	0	1	1	0	0

Answer: 1 → 6 → 9 → 4 → 4 → 4
→ 8 → 0



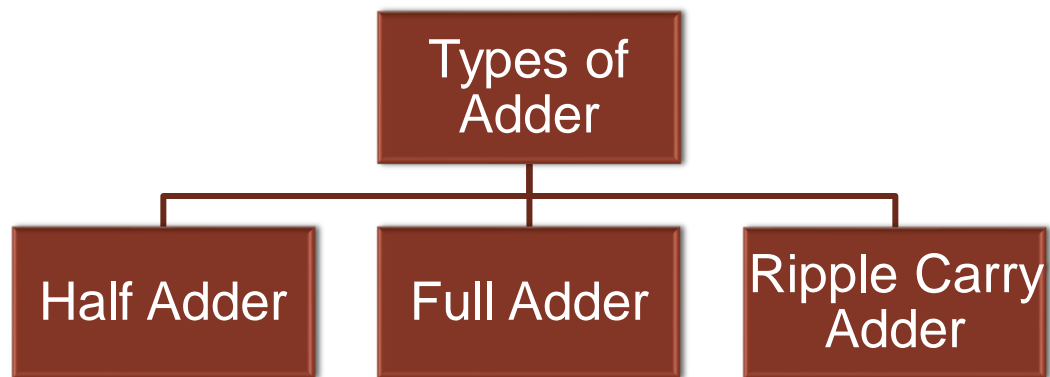


ADDERS

ADDERS & COMPARATOR

ADDERS: INTRODUCTION

- **Adders** combine two operand arithmetically using binary addition rules.



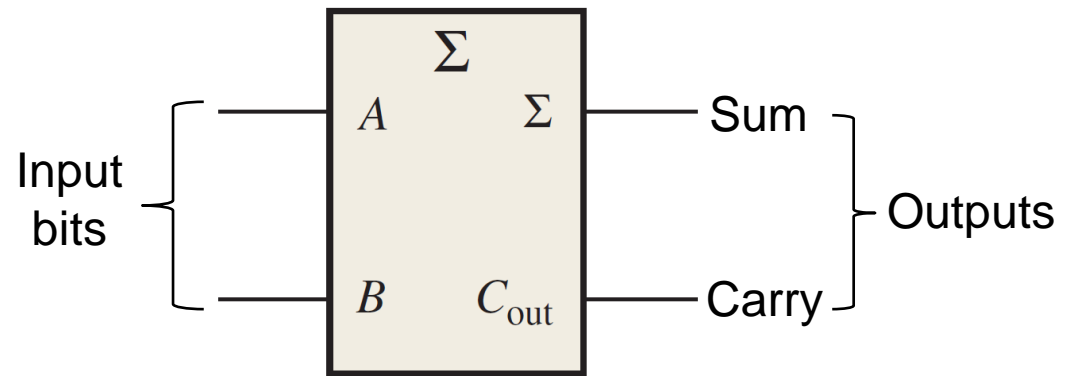
ADDERS & COMPARATOR

ADDERS: HALF ADDER

- **Half Adder** accepts **two binary** digits on its inputs and produce two binary digits on its outputs, **sum bit** and **carry bit**.

Truth Table for Half Adder

Inputs		Outputs	
A	B	C_{out}	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

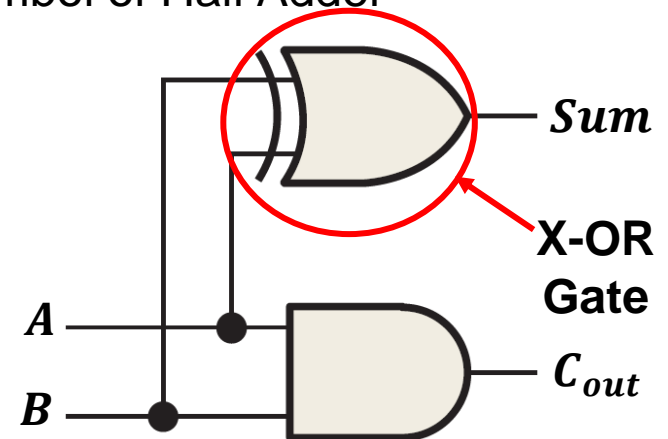


Logic Symbol of Half Adder

- From the operation of half adder, we can derived that:

$$C_{out} = AB$$

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



ADDERS & COMPARATOR

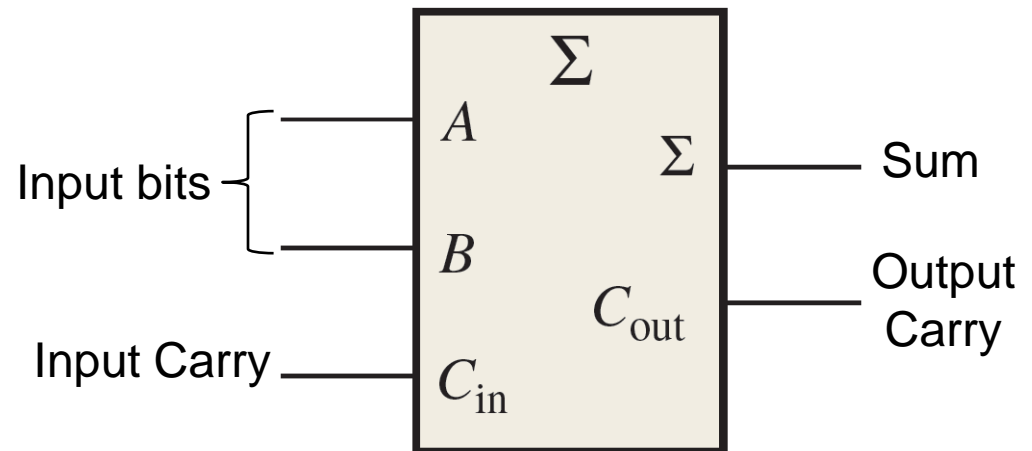
ADDERS: FULL ADDER

- **Full Adder** accepts **two inputs bits & input carry** and generate **sum output & output carry**.

Truth Table for Full Adder

Inputs			Outputs	
A	B	C_{in}	C_{out}	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Logic Symbol of Full Adder



ADDERS & COMPARATOR

ADDERS: FULL ADDER

- From the operation of Full Adder, we can derive Boolean equation of C_{out} and Sum using K-map.

		C_{out}			
		00	01	11	10
AB	C_{in}	00	01	11	10
	0	0	0	1	0
1	0	1	1	1	

$C_{in}B$ (points to 1 in row 1, col 2)
 $C_{in}A$ (points to 1 in row 1, col 4)
 AB (points to 1 in row 2, col 3)

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

		Sum			
		00	01	11	10
BC _{in}	A	00	01	11	10
	0	0	1	0	1
1	1	1	0	1	0

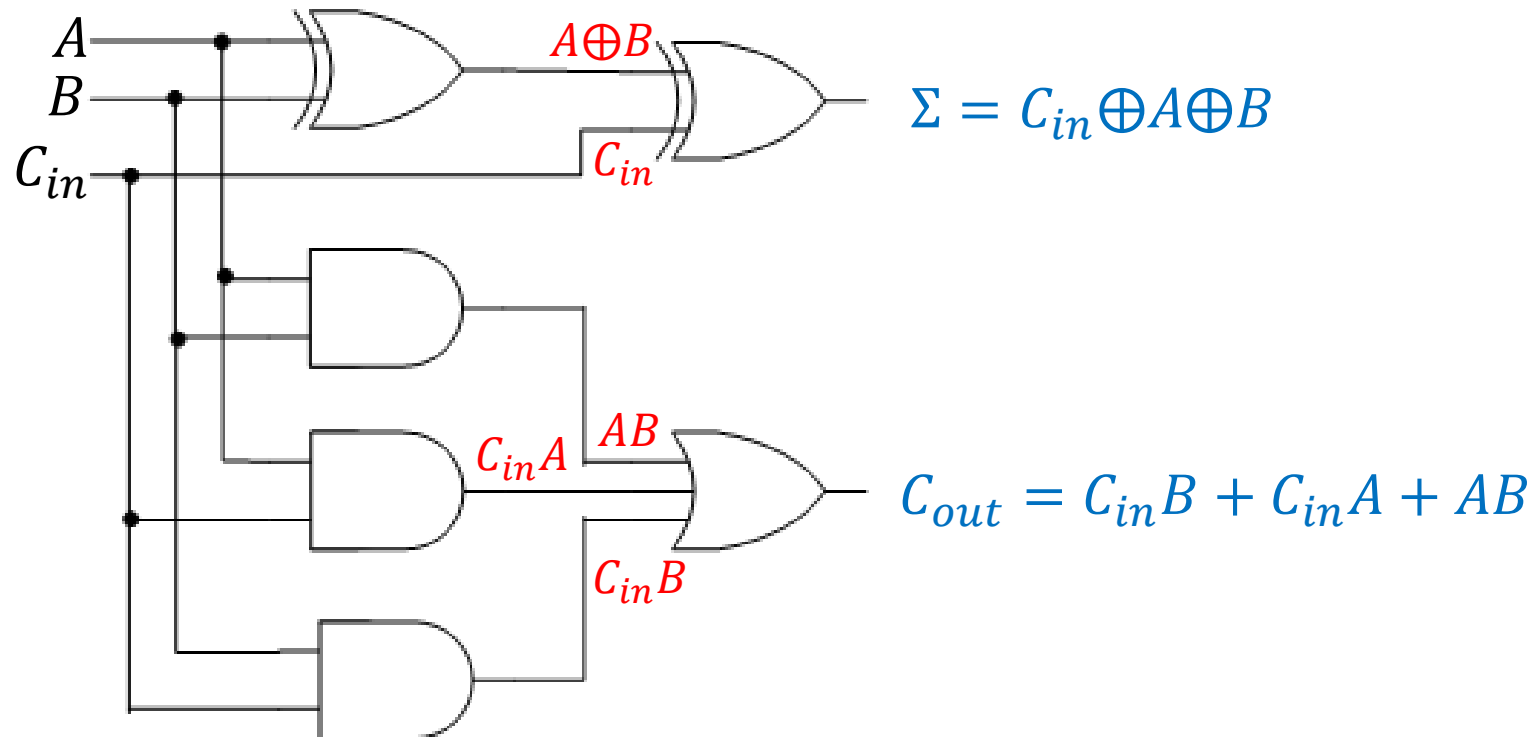
$C_{in}A$ (points to 1 in row 1, col 1)
 $C_{in}\bar{A}\bar{B}$ (points to 1 in row 2, col 2)
 $C_{in}AB$ (points to 1 in row 2, col 4)
 $\bar{C}_{in}\bar{A}B$ (points to 1 in row 1, col 4)

$$\begin{aligned}
 \Sigma &= C_{in}\bar{A}\bar{B} + C_{in}AB + \bar{C}_{in}\bar{A}B + \bar{C}_{in}A\bar{B} \\
 &= C_{in}(\bar{A}\bar{B} + AB) + \bar{C}_{in}(\bar{A}B + A\bar{B}) \\
 &= C_{in}(\overline{\bar{A}B + AB}) + \bar{C}_{in}(\bar{A}B + A\bar{B}) \\
 &= C_{in}\oplus A\oplus B
 \end{aligned}$$

ADDERS & COMPARATOR

ADDERS: FULL ADDER

- Logic circuit of **Full Adder**:

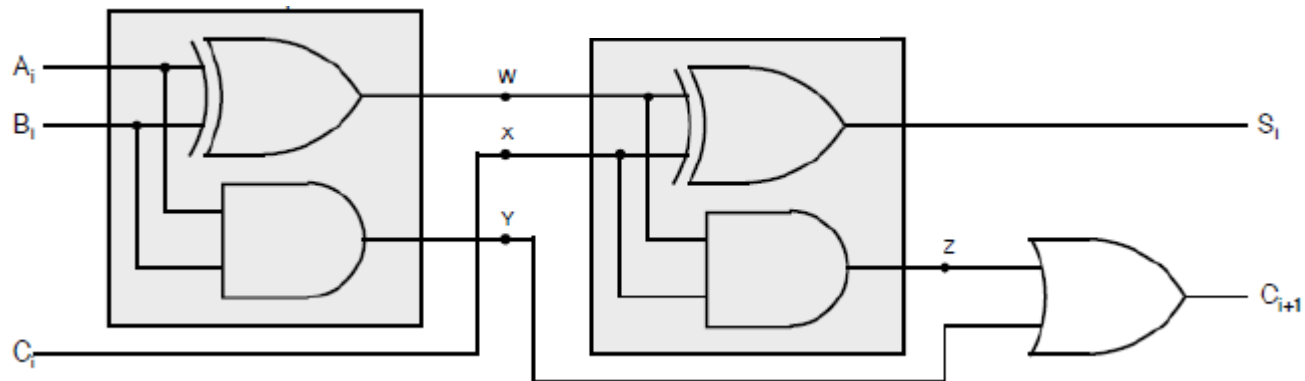
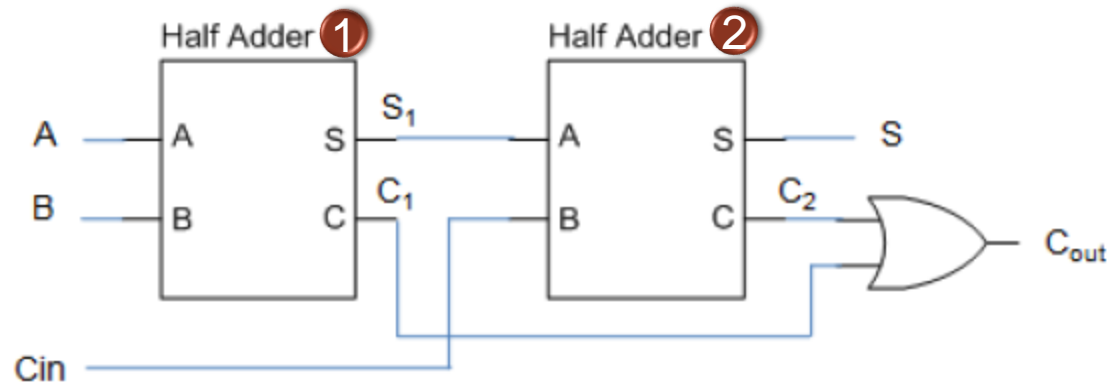


Logic Circuit of Full Adder

ADDERS & COMPARATOR

ADDERS: FULL ADDER

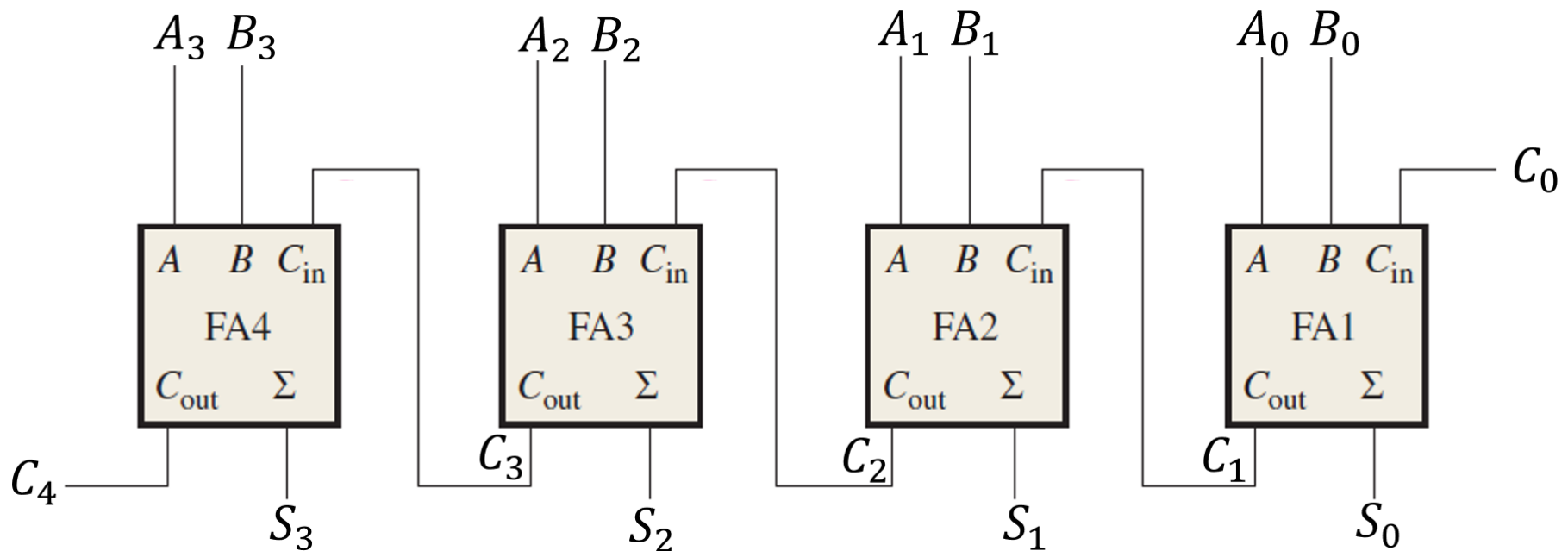
- How to design Full Adder using Half Adder?



ADDERS & COMPARATOR

ADDERS: RIPPLE CARRY ADDER

- **Ripple Carry Adder** is used to add multiple bit binary numbers.
- The **carry-out output** from a state is connected to the **carry-in input** of the next state.
- To design **4-bit ripple carry adder**, we need **4 full adders**.
- Input = $A_3A_2A_1A_0$, $B_3B_2B_1B_0$ and C_0 (C_0 initially set to 0).
- Output = $S_3S_2S_1S_0$ and C_4 .

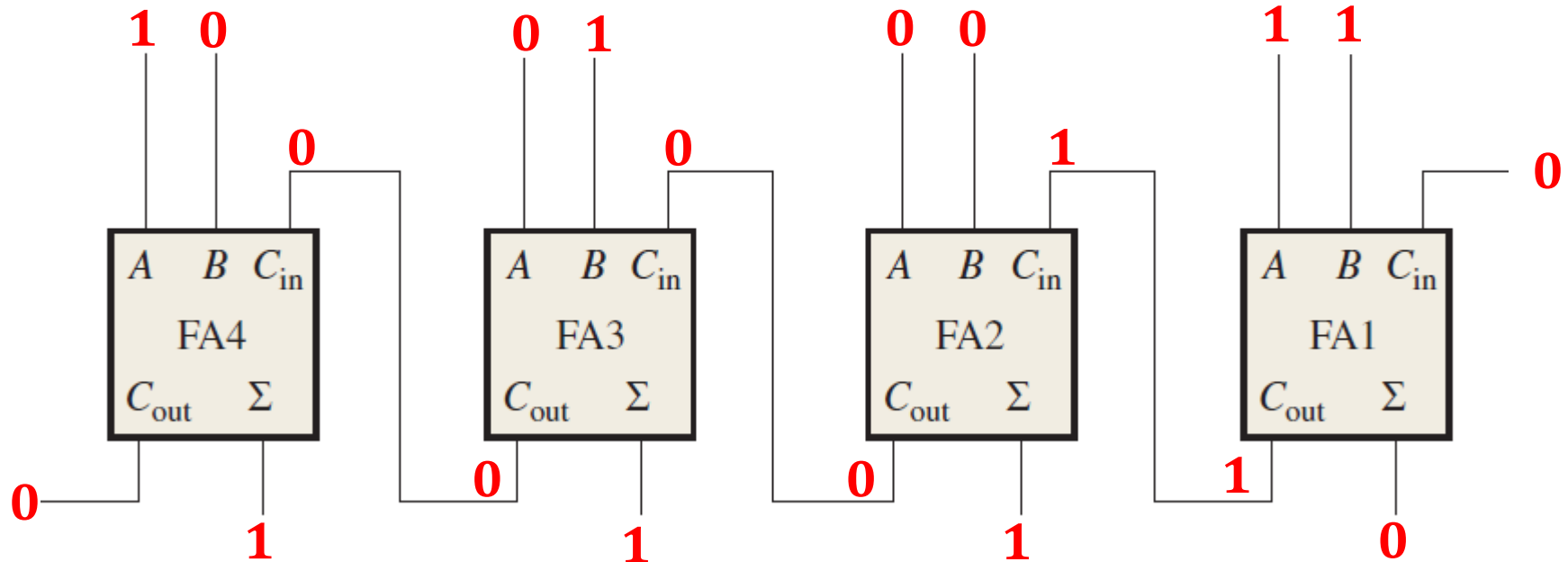


ADDERS & COMPARATOR

ADDERS: RIPPLE CARRY ADDER

Example 1

1001 + 0101



1 0 0 1 + 0 1 0 1

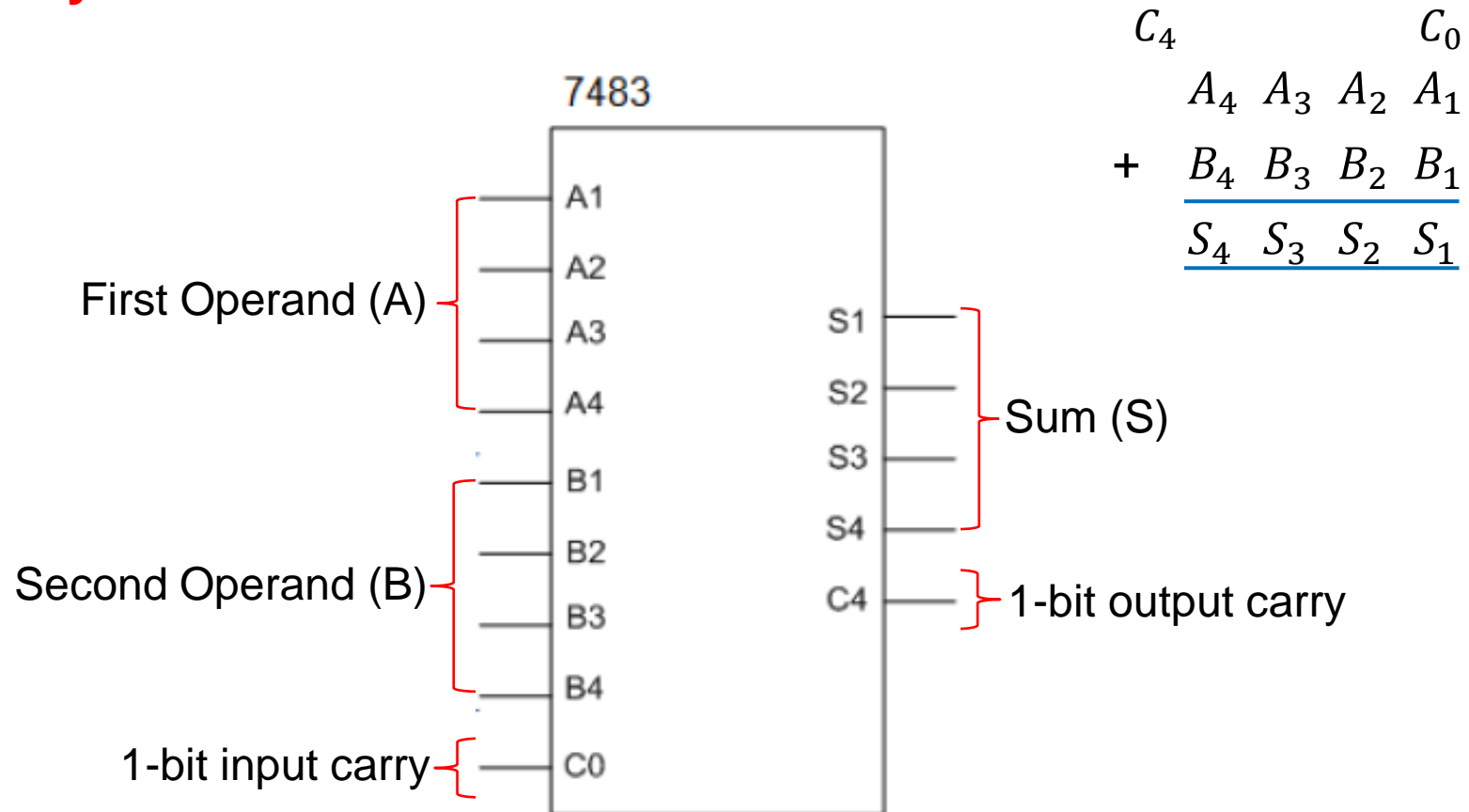
$A_3 A_2 A_1 A_0$ $B_3 B_2 B_1 B_0$

ANS = 1110

ADDERS & COMPARATOR

ADDERS: RIPPLE CARRY ADDER

- The **74x83** is an example of IC device for faster **4-bit ripple carry adder**.

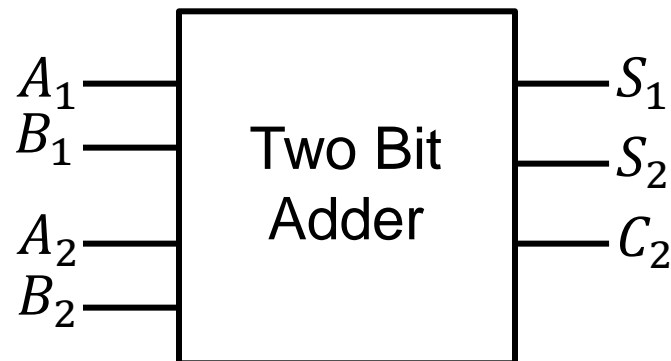


Logic Symbol of 74x83 Ripple Carry Adder

ADDERS & COMPARATOR

Example 2

Figure below shows the block diagram of a two bits adder A_2A_1 and B_2B_1 addition. The result should be in three bits binary number $C_2S_2S_1$. Obtain the truth table for output C_2 , S_2 and S_1 .



$$\begin{array}{r} C_1 \\ A_2 \ A_1 \\ + \ B_2 \ B_1 \\ \hline C_2 \ S_2 \ S_1 \end{array}$$

ADDERS & COMPARATOR

Example 2

Figure below shows the block diagram of a two bits adder A_2A_1 and B_2B_1 addition. The result should be in three bits binary number $C_2S_2S_1$. Obtain the truth table for output C_2 , S_2 and S_1 .

Truth Table for 2 bits Binary Addition

Input				Output		
A2	A1	B2	B1	C2	S2	S1
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	0	1	1	0	1	1
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0

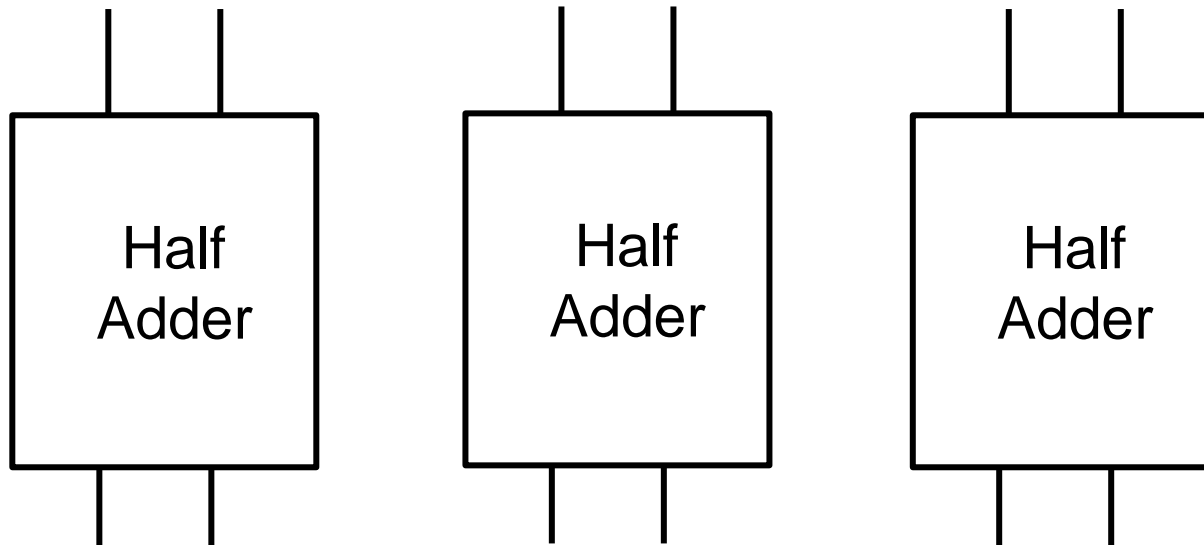
Input				Output		
A2	A1	B2	B1	C2	S2	S1
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	1	1	0

ADDERS & COMPARATOR

Example 2(cont.)

Design the adder using three half adder and logic gate by completing figure as follows:

A_1 B_1 A_2 B_2

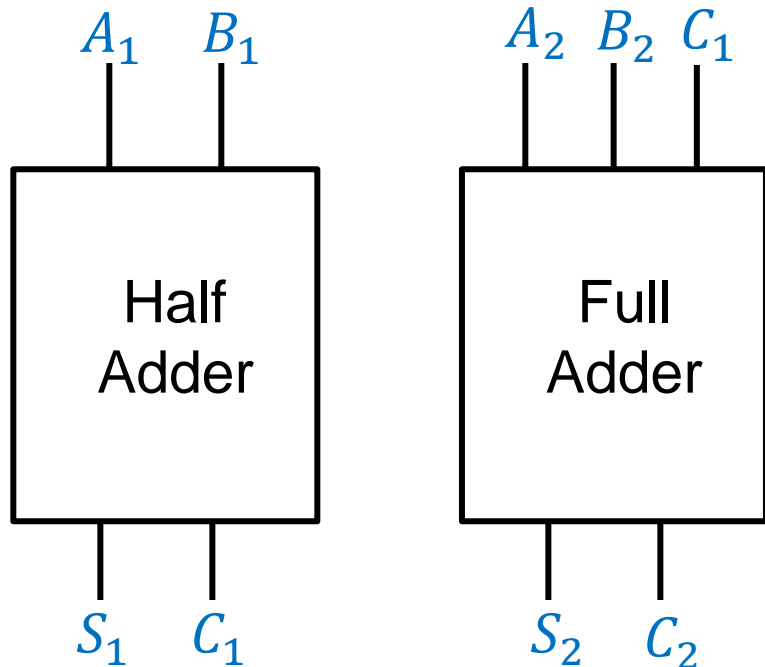


— C_2
— S_2
— S_1

ADDERS & COMPARATOR

Example 2(cont.)

Design the adder using three half adder and logic gate by completing figure as follows:

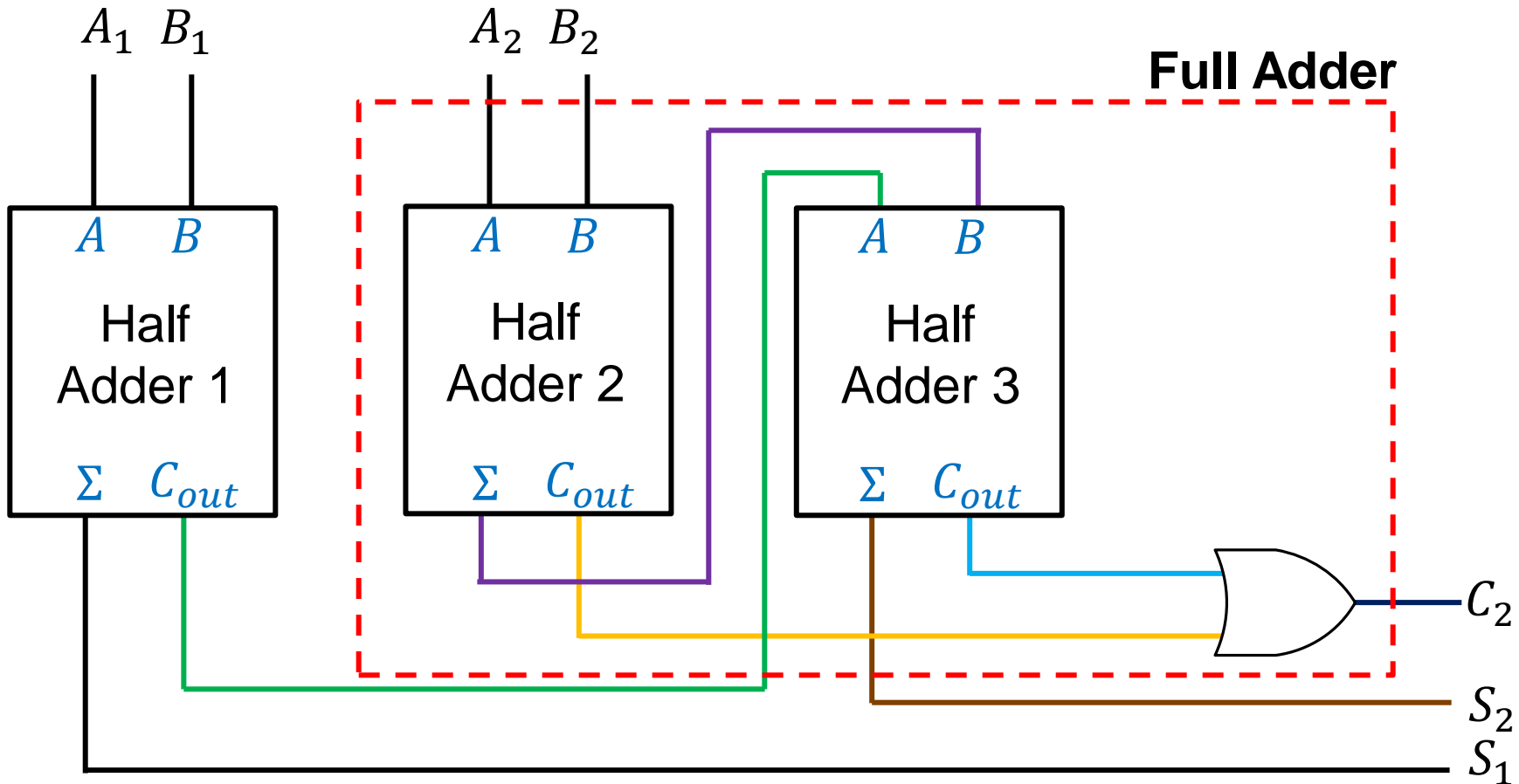


$$\begin{array}{r} C_1 \\ A_2 \ A_1 \\ + \ B_2 \ B_1 \\ \hline C_2 \ S_2 \ S_1 \end{array}$$

ADDERS & COMPARATOR

Example 2(cont.)

Design the adder using three half adder and logic gate by completing figure as follows:



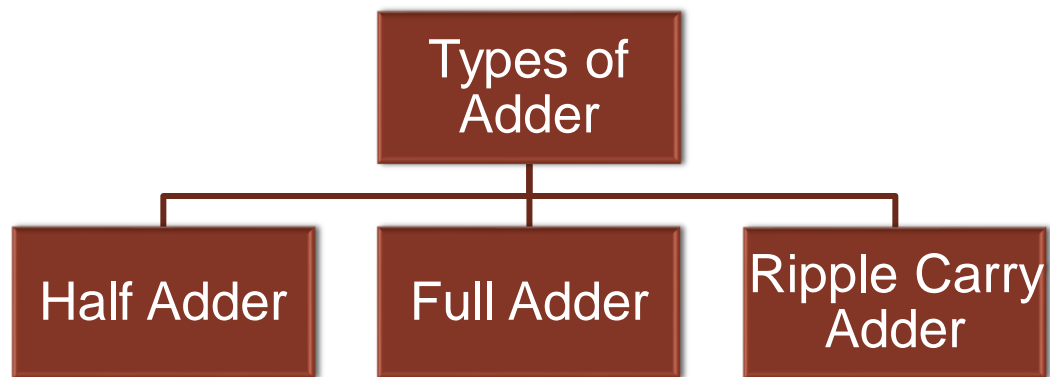


ADDERS

ADDERS & COMPARATOR

ADDERS: INTRODUCTION

- **Adders** combine two operand arithmetically using binary addition rules.



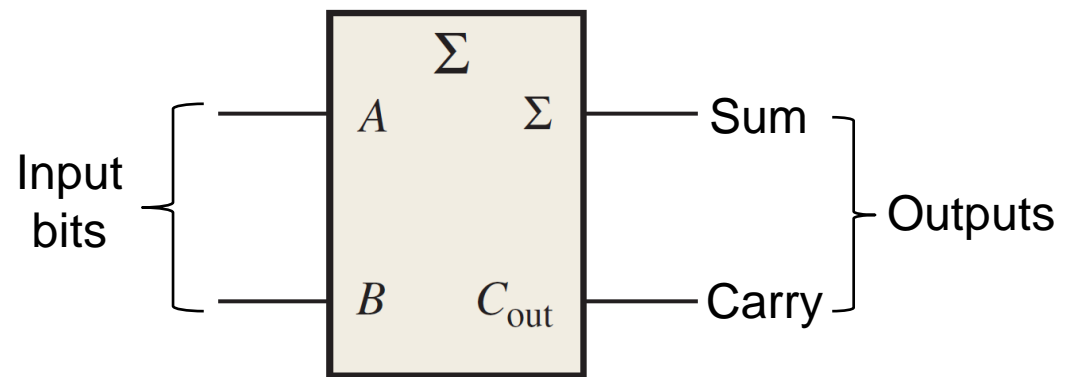
ADDERS & COMPARATOR

ADDERS: HALF ADDER

- **Half Adder** accepts **two binary** digits on its inputs and produce two binary digits on its outputs, **sum bit** and **carry bit**.

Truth Table for Half Adder

Inputs		Outputs	
A	B	C_{out}	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

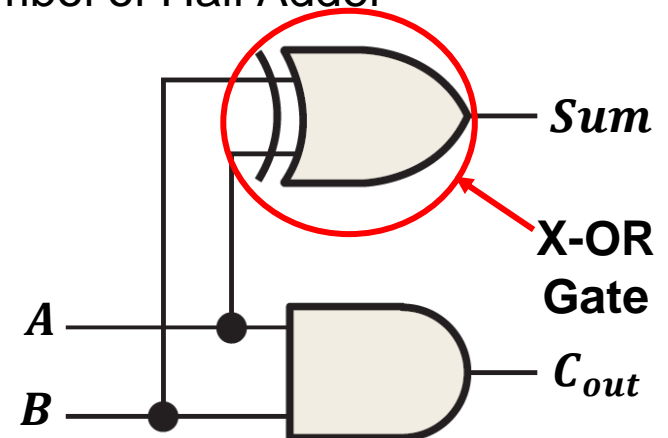


Logic Symbol of Half Adder

- From the operation of half adder, we can derived that:

$$C_{out} = AB$$

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



ADDERS & COMPARATOR

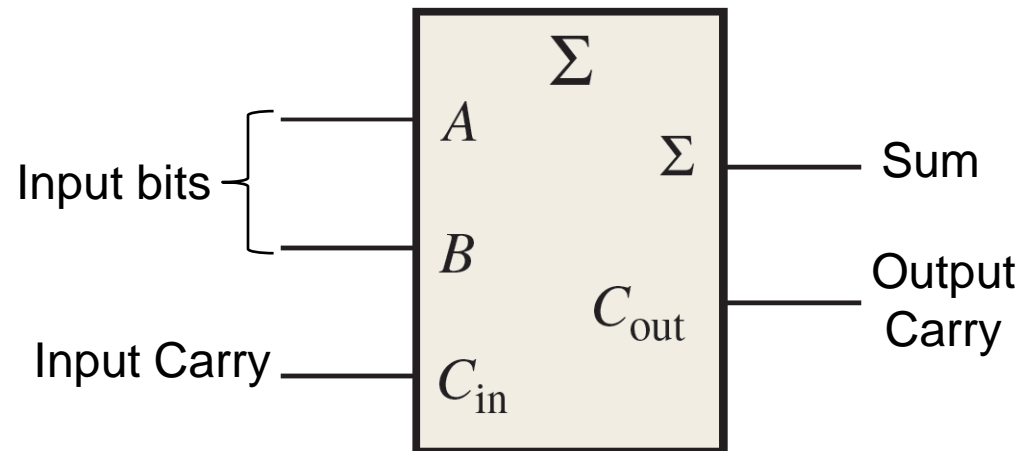
ADDERS: FULL ADDER

- **Full Adder** accepts **two inputs bits & input carry** and generate **sum output & output carry**.

Truth Table for Full Adder

Inputs			Outputs	
A	B	C_{in}	C_{out}	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Logic Symbol of Full Adder



ADDERS & COMPARATOR

ADDERS: FULL ADDER

- From the operation of Full Adder, we can derive Boolean equation of C_{out} and Sum using K-map.

		C_{out}			
		00	01	11	10
AB	C_{in}	00	01	11	10
	0	0	0	1	0
1	0	1	1	1	

$C_{in}B$ (points to 1 in row 1, col 2)
 $C_{in}A$ (points to 1 in row 1, col 4)
 AB (points to 1 in row 2, col 3)

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

		Sum			
		00	01	11	10
BC _{in}	A	00	01	11	10
	0	0	1	0	1
1	1	0	1	0	

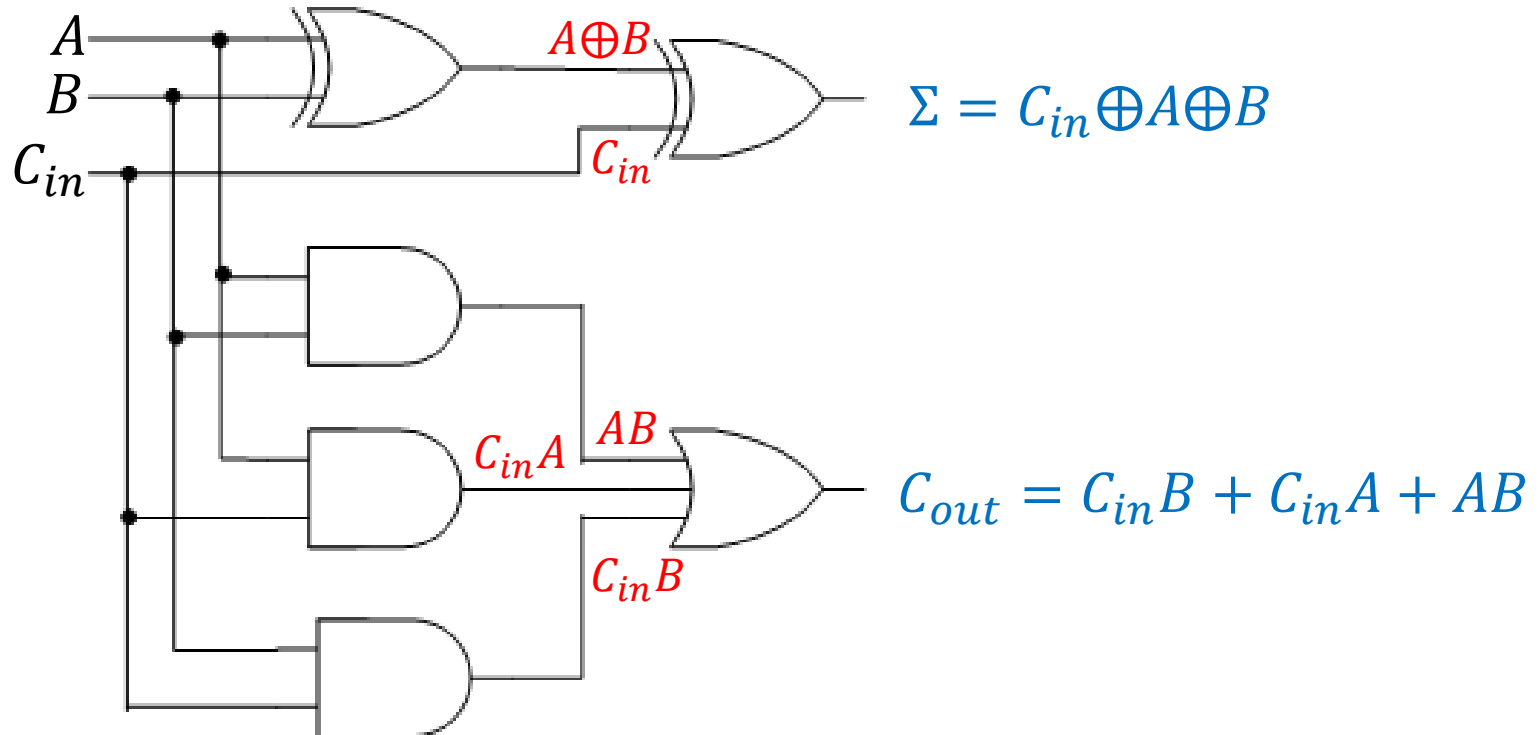
$C_{in}A$ (points to 1 in row 1, col 1)
 $C_{in}\bar{A}\bar{B}$ (points to 1 in row 2, col 2)
 $C_{in}AB$ (points to 1 in row 2, col 4)
 $\bar{C}_{in}\bar{A}B$ (points to 1 in row 1, col 4)

$$\begin{aligned}
 \Sigma &= C_{in}\bar{A}\bar{B} + C_{in}AB + \bar{C}_{in}\bar{A}B + \bar{C}_{in}A\bar{B} \\
 &= C_{in}(\bar{A}\bar{B} + AB) + \bar{C}_{in}(\bar{A}B + A\bar{B}) \\
 &= C_{in}(\overline{\bar{A}B + A\bar{B}}) + \bar{C}_{in}(\bar{A}B + A\bar{B}) \\
 &= C_{in}\oplus A\oplus B
 \end{aligned}$$

ADDERS & COMPARATOR

ADDERS: FULL ADDER

- Logic circuit of **Full Adder**:

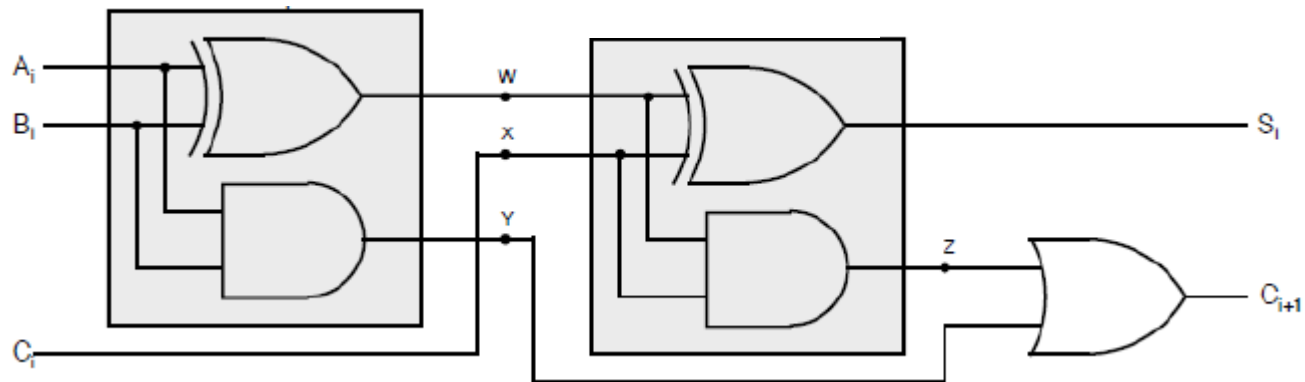
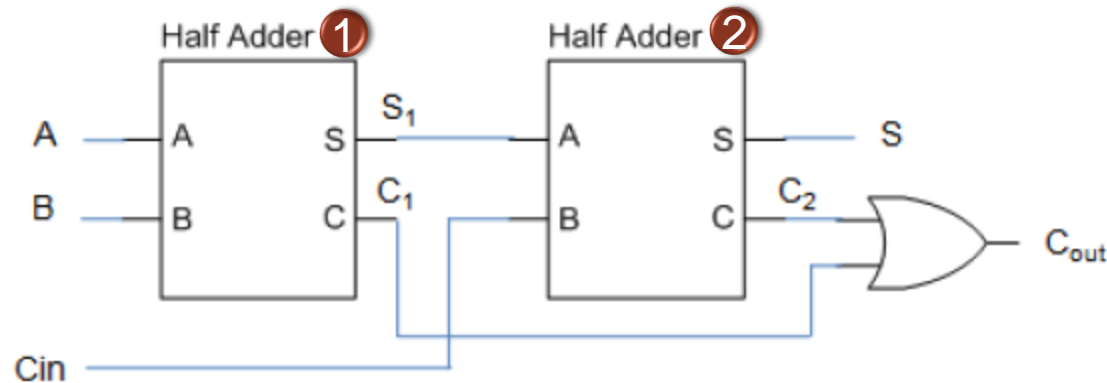


Logic Circuit of Full Adder

ADDERS & COMPARATOR

ADDERS: FULL ADDER

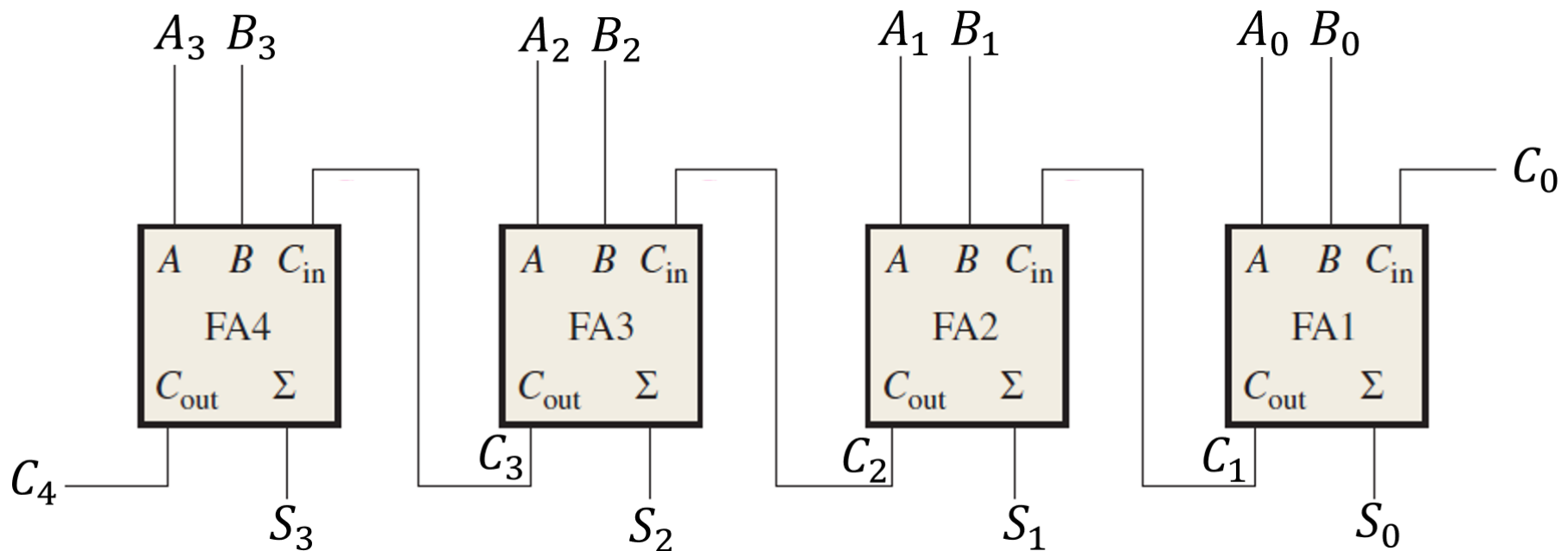
- How to design Full Adder using Half Adder?



ADDERS & COMPARATOR

ADDERS: RIPPLE CARRY ADDER

- **Ripple Carry Adder** is used to add multiple bit binary numbers.
- The **carry-out output** from a state is connected to the **carry-in input** of the next state.
- To design **4-bit ripple carry adder**, we need **4 full adders**.
- Input = $A_3A_2A_1A_0$, $B_3B_2B_1B_0$ and C_0 (C_0 initially set to 0).
- Output = $S_3S_2S_1S_0$ and C_4 .

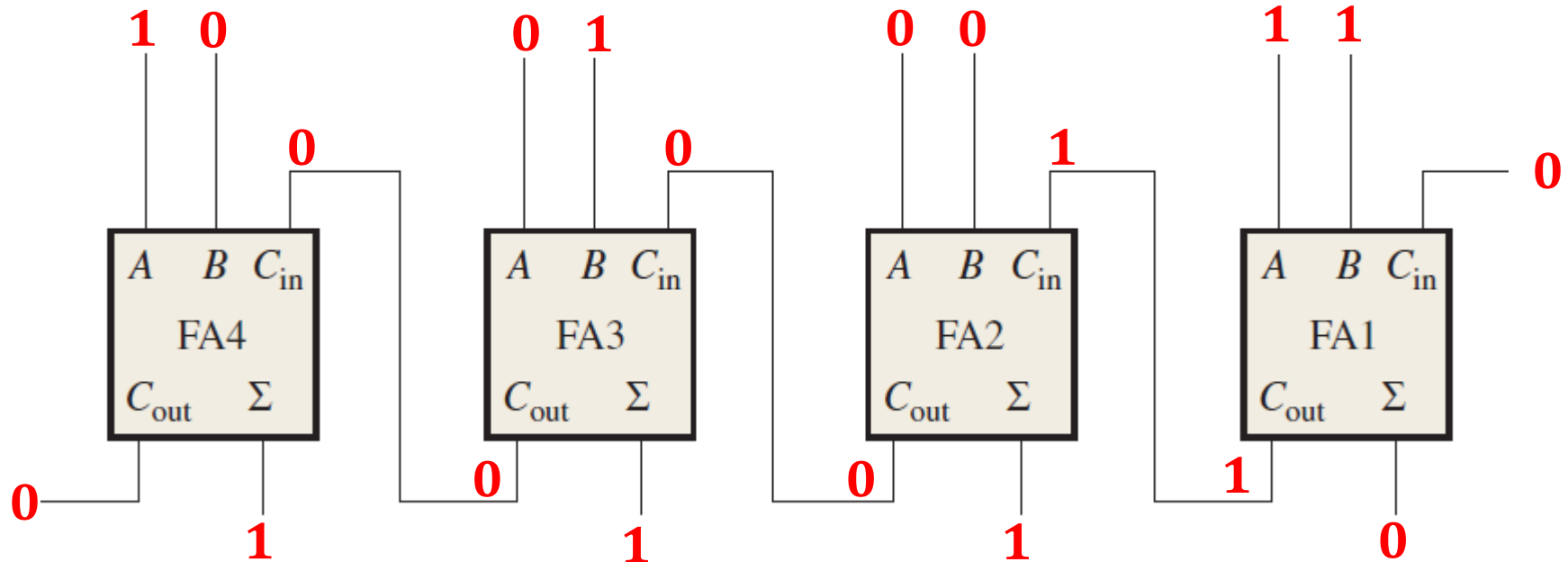


ADDERS & COMPARATOR

ADDERS: RIPPLE CARRY ADDER

Example 1

1001 + 0101



1 0 0 1 + 0 1 0 1

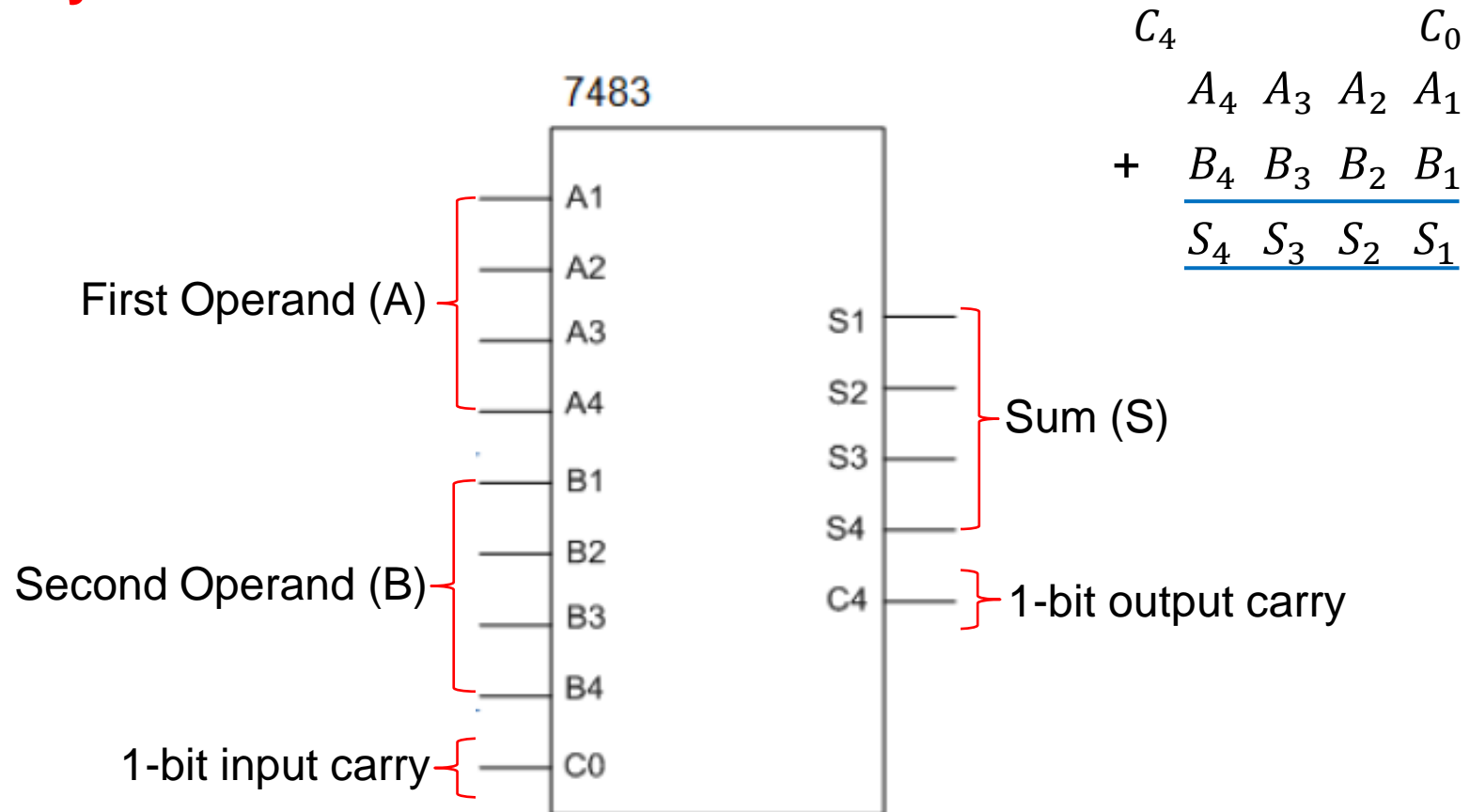
$A_3 A_2 A_1 A_0$ $B_3 B_2 B_1 B_0$

ANS = 1110

ADDERS & COMPARATOR

ADDERS: RIPPLE CARRY ADDER

- The **74x83** is an example of IC device for faster **4-bit ripple carry adder**.

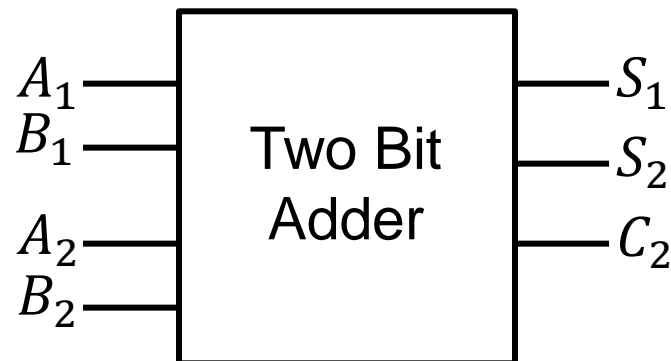


Logic Symbol of 74x83 Ripple Carry Adder

ADDERS & COMPARATOR

Example 2

Figure below shows the block diagram of a two bits adder A_2A_1 and B_2B_1 addition. The result should be in three bits binary number $C_2S_2S_1$. Obtain the truth table for output C_2 , S_2 and S_1 .



$$\begin{array}{r} C_1 \\ A_2 \ A_1 \\ + \ B_2 \ B_1 \\ \hline C_2 \ S_2 \ S_1 \end{array}$$

ADDERS & COMPARATOR

Example 2

Figure below shows the block diagram of a two bits adder A_2A_1 and B_2B_1 addition. The result should be in three bits binary number $C_2S_2S_1$. Obtain the truth table for output C_2 , S_2 and S_1 .

Truth Table for 2 bits Binary Addition

Input				Output		
A2	A1	B2	B1	C2	S2	S1
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	0	1	1	0	1	1
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0

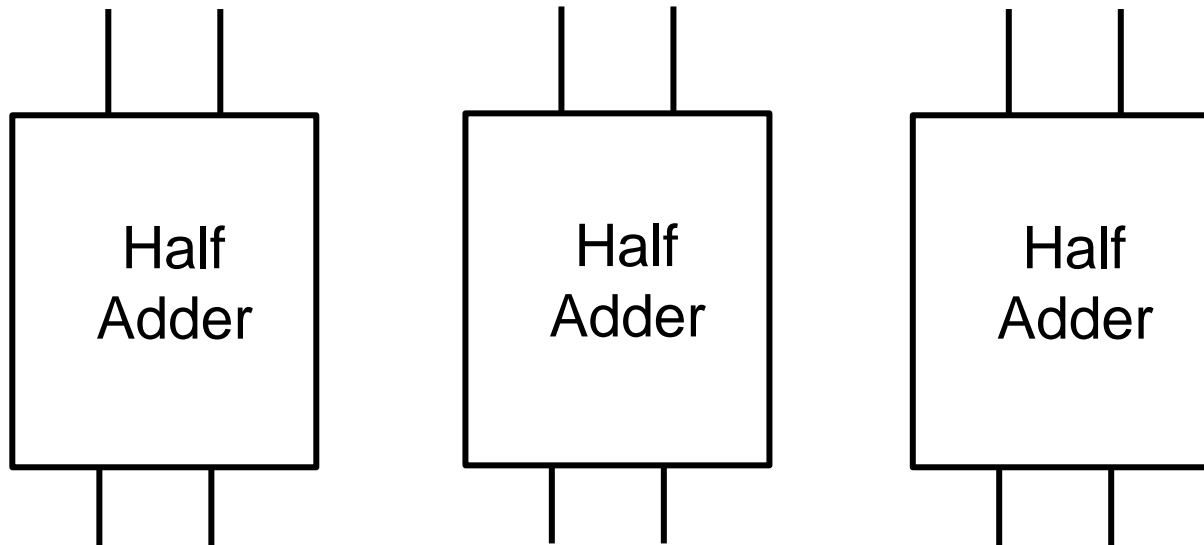
Input				Output		
A2	A1	B2	B1	C2	S2	S1
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	1	1	0

ADDERS & COMPARATOR

Example 2(cont.)

Design the adder using three half adder and logic gate by completing figure as follows:

A_1 B_1 A_2 B_2

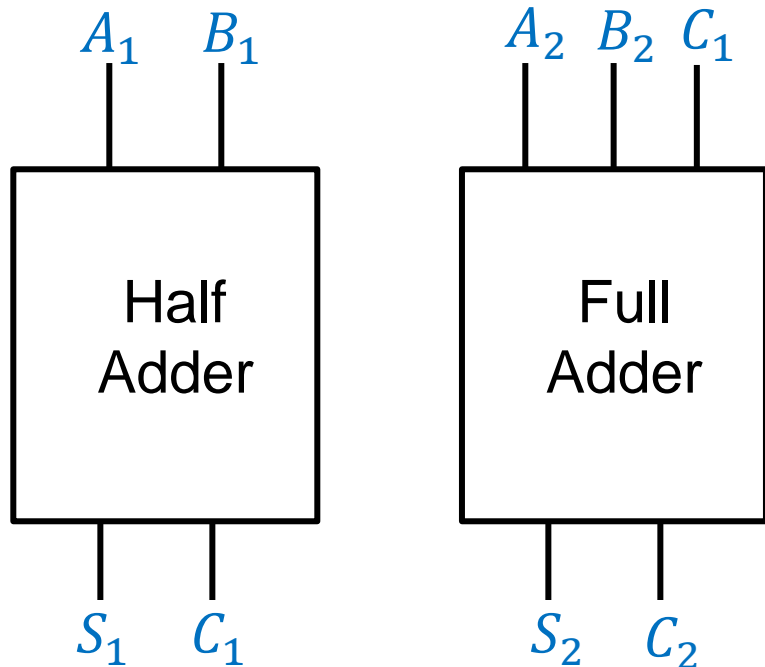


— C_2
— S_2
— S_1

ADDERS & COMPARATOR

Example 2(cont.)

Design the adder using three half adder and logic gate by completing figure as follows:

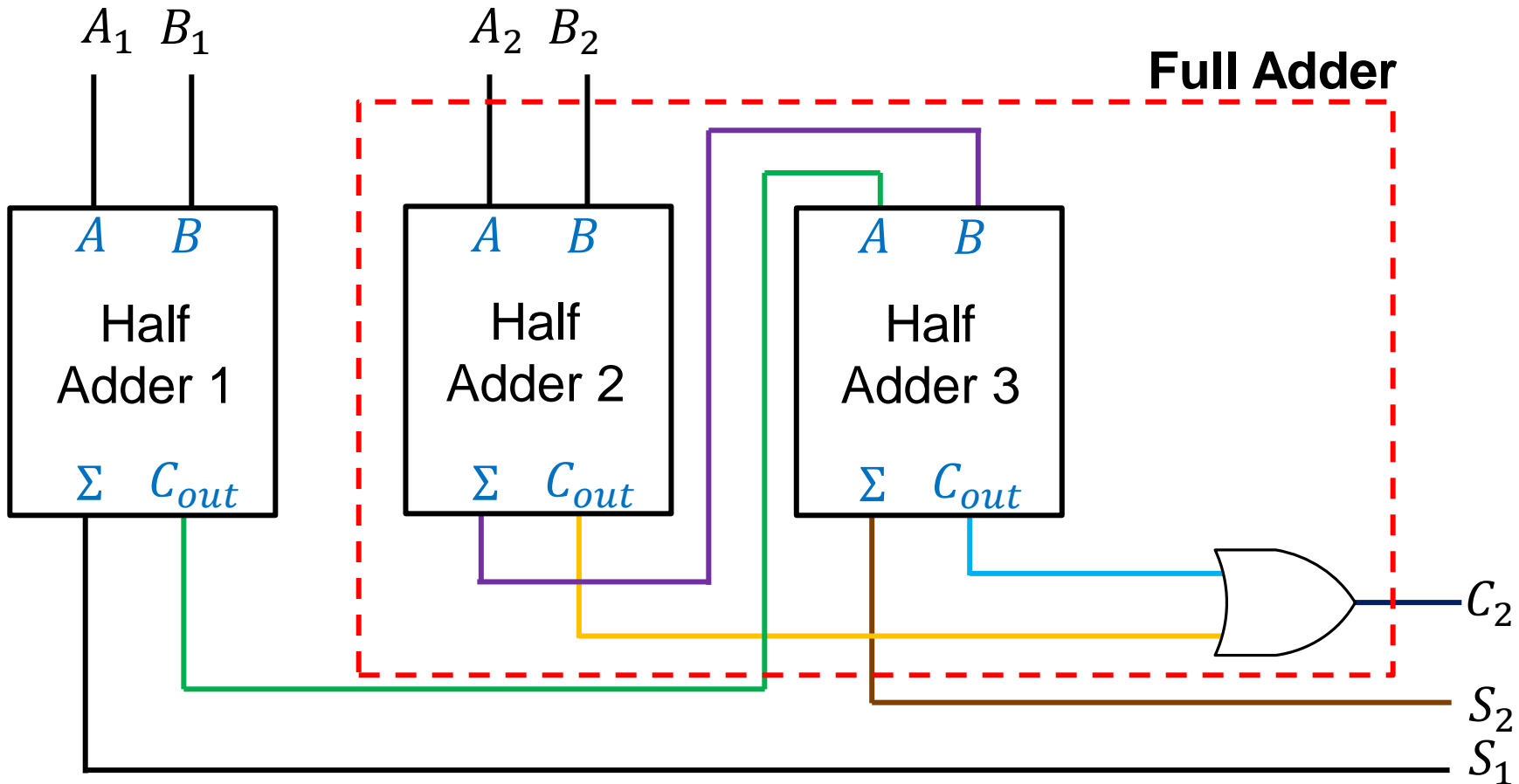


$$\begin{array}{r} C_1 \\ A_2 \ A_1 \\ + \ B_2 \ B_1 \\ \hline C_2 \ S_2 \ S_1 \end{array}$$

ADDERS & COMPARATOR

Example 2(cont.)

Design the adder using three half adder and logic gate by completing figure as follows:





COMPARATOR

ADDERS & COMPARATOR

COMPARATOR: EQUALITY

- **Comparator** is used to compare the magnitude of two binary quantities to determine the relationship of those quantities.
- As learned in Chapter 3, the **exclusive-NOR** gate can be used as a basic comparator.



$$S = \overline{\overline{A} \cdot B + A \cdot \overline{B}} = \overline{\overline{0} \cdot 0 + 0 \cdot \overline{0}} = \overline{1 \cdot 0 + 0 \cdot 1} = \overline{0} = 1$$

ADDERS & COMPARATOR

COMPARATOR: EQUALITY

- **Comparator** is used to compare the magnitude of two binary quantities to determine the relationship of those quantities.
- As learned in Chapter 3, the **exclusive-NOR** gate can be used as a basic comparator.



$$S = \overline{\overline{A} \cdot B + A \cdot \overline{B}} = \overline{\overline{0} \cdot 1 + 1 \cdot \overline{0}} = \overline{1 \cdot 1 + 1 \cdot 1} = \overline{1} = 0$$

ADDERS & COMPARATOR

COMPARATOR: EQUALITY

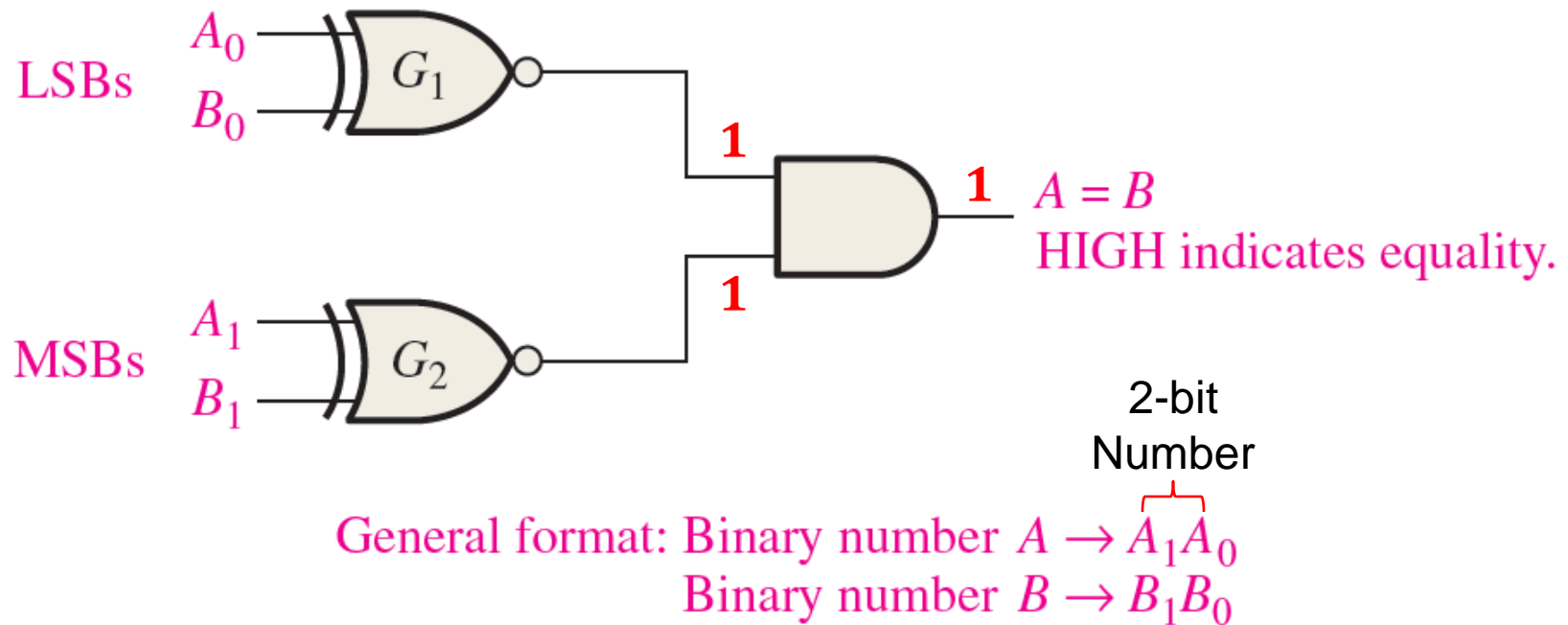
- **Comparator** is used to compare the magnitude of two binary quantities to determine the relationship of those quantities.
- As learned in Chapter 3, the **exclusive-NOR** gate can be used as a basic comparator.



- In order to compare binary number containing two each bits, an additional **exclusive-NOR**, **NOT** and **AND** gate are necessary.

ADDERS & COMPARATOR

COMPARATOR: EQUALITY

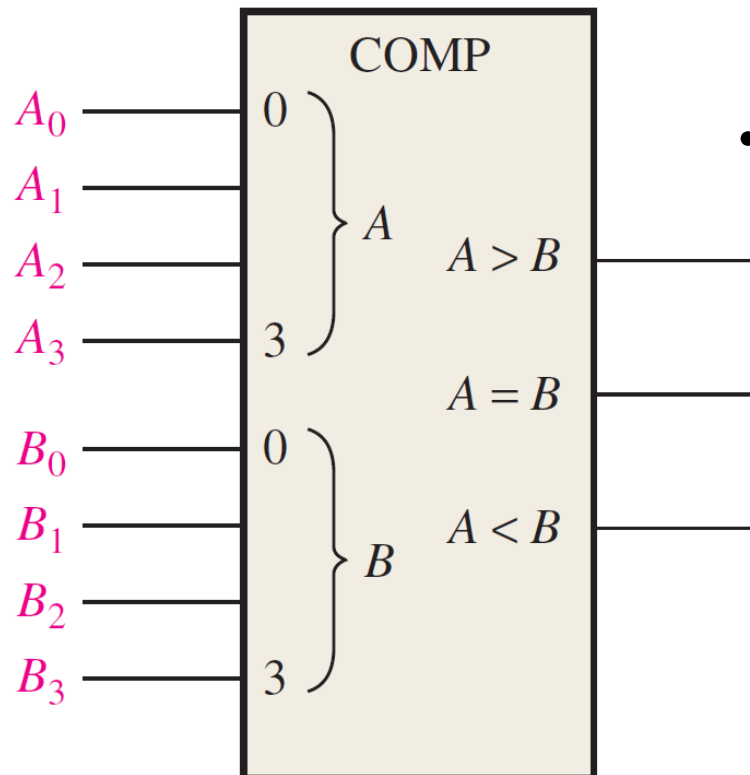


- The output indicate **(1) is equality** while **(0) is inequality**.

ADDERS & COMPARATOR

COMPARATOR: INEQUALITY

- In addition to **equality** output, many IC comparators (**74x85**) provide additional outputs that indicate which of the two binary numbers being compared is the larger.
- That is, **$(A > B) \& (A < B)$** .



- To determine inequality of numbers A and B, first examine the highest order bit in each number:
 - If $A_3 = 1$ and $B_3 = 0$; **$A > B$**
 - If $A_3 = 0$ and $B_3 = 1$; **$A < B$**
 - If $A_3 = B_3$; then examine the next lower bit position for an inequality.

ADDERS & COMPARATOR

COMPARATOR: INEQUALITY

- The truth table for **74x85 comparator**.

A_1	A_0	B_1	B_0	$A = B$ (F_1)	$A > B$ (F_2)	$A < B$ (F_3)
0	0	0	0	1	0	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	0	1	0
0	1	0	1	1	0	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1

A_1	A_0	B_1	B_0	$A = B$ (F_1)	$A > B$ (F_2)	$A < B$ (F_3)
1	0	0	0	0	1	0
1	0	0	1	0	1	0
1	0	1	0	1	0	0
1	0	1	1	0	0	1
1	1	0	0	0	1	0
1	1	0	1	0	1	0
1	1	1	0	0	1	0
1	1	1	1	1	0	0

ADDERS & COMPARATOR

COMPARATOR: INEQUALITY

- By using K-map, expression of F1, F2 and F3 are obtained as follows

For $A = B$:

$$F_1 = \overline{A_1} \cdot \overline{A_0} \cdot \overline{B_1} \cdot \overline{B_0} + \overline{A_1} \cdot A_0 \cdot \overline{B_1} \cdot B_0 + A_1 \cdot \overline{A_0} \cdot B_1 \cdot \overline{B_0} + A_1 \cdot A_0 \cdot B_1 \cdot B_0$$

For $A > B$:

$$F_1 = A_1 \cdot \overline{B_1} + A_0 \cdot \overline{B_1} \cdot \overline{B_0} + A_1 \cdot A_0 \cdot \overline{B_0}$$

For $A < B$:

$$F_1 = \overline{A_1} \cdot B_1 + \overline{A_0} \cdot B_1 \cdot B_0 + \overline{A_1} \cdot \overline{A_0} \cdot B_0$$

ADDERS & COMPARATOR

Example 4

The waveform are applied to comparator as shown. Determine the output ($A = B$) waveform.

