Digital System Design

Topics to be covered

- Adders (Half, Full, Parallel Adders)
- Subtractor (Half and Full)
- N-bit Comparator
- Encoder, Decoder and Priority Encoder
- Multiplexer and DeMultiplexer

Digital System Design Concept

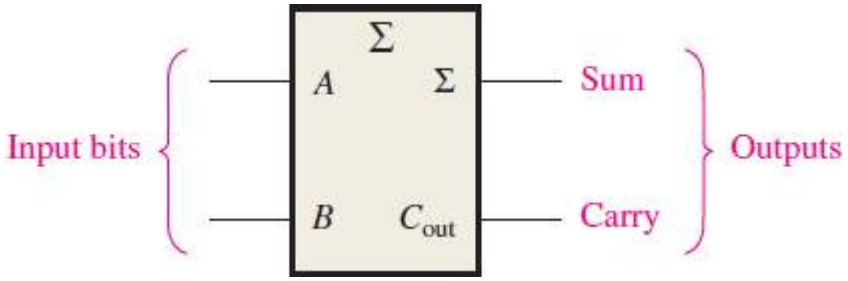
Designing a digital system requires the following steps to be performed:

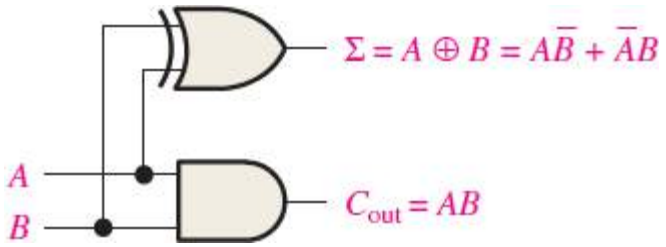
Topics covered in DLD

- Look for the problem statement
- Find out the inputs & outputs of the system
- Relate the inputs of your system with the outputs
- Develop a truth table/behavior table for your system using the input and output relationship that you have established
- From your truth table generate a standard output expression, relating the inputs to the output
- Reduce the output expression using either Boolean Algebra or k-MAP
- Write down your reduced expression (minimized expression)
- Design the system circuit with combinational logic gates
- **Convert the combinational logic gates to transistor level schematic (CMOS schematic)**

Topic covered in VLSI Circuit Design Course

HALF ADDER





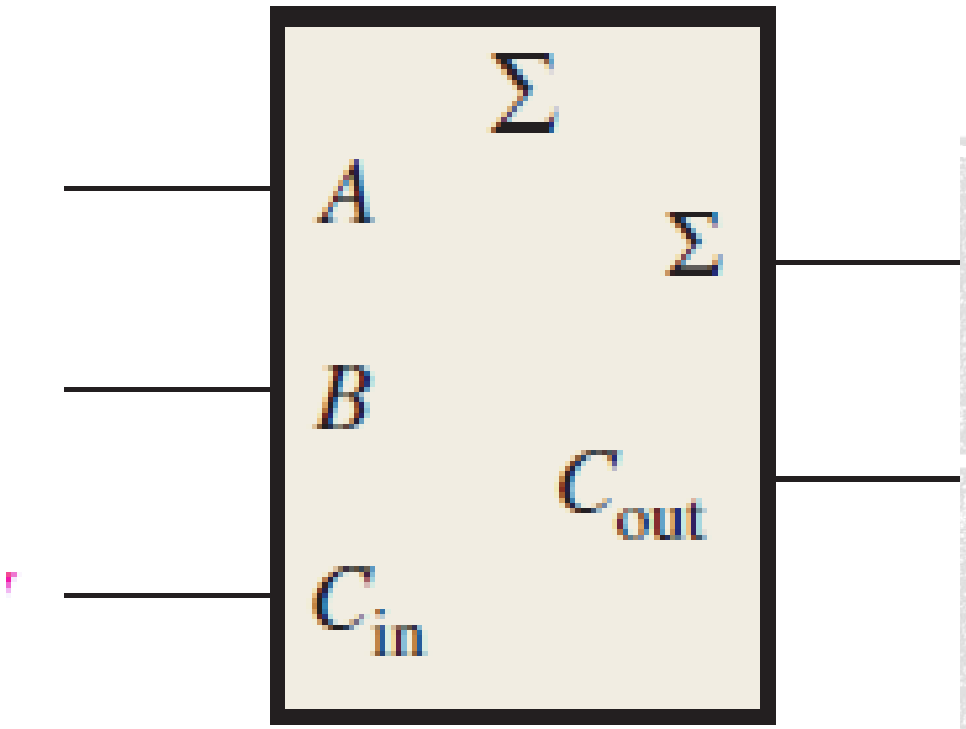
Basic Rules for Binary Addition

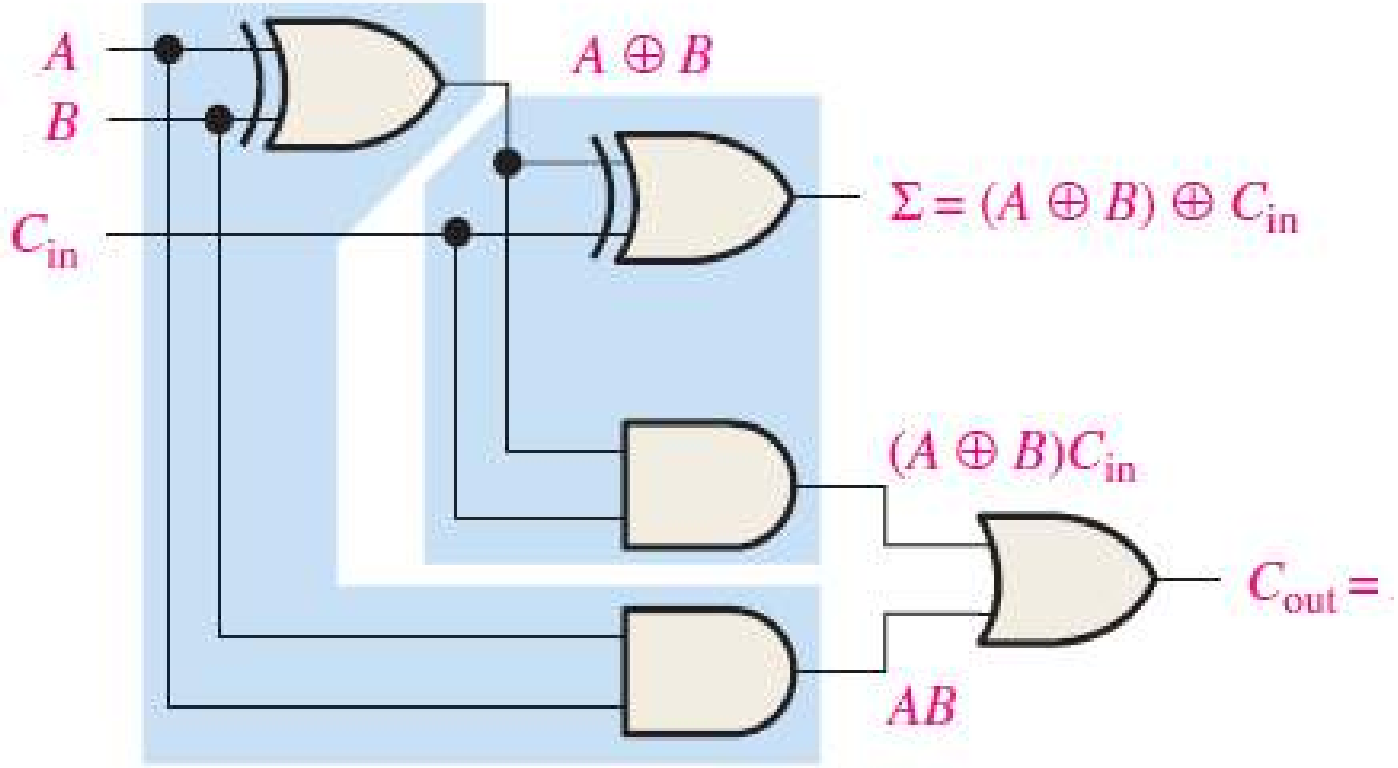
$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 10$$

Half-adder truth table.

A	B	C_{out}	Σ
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

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



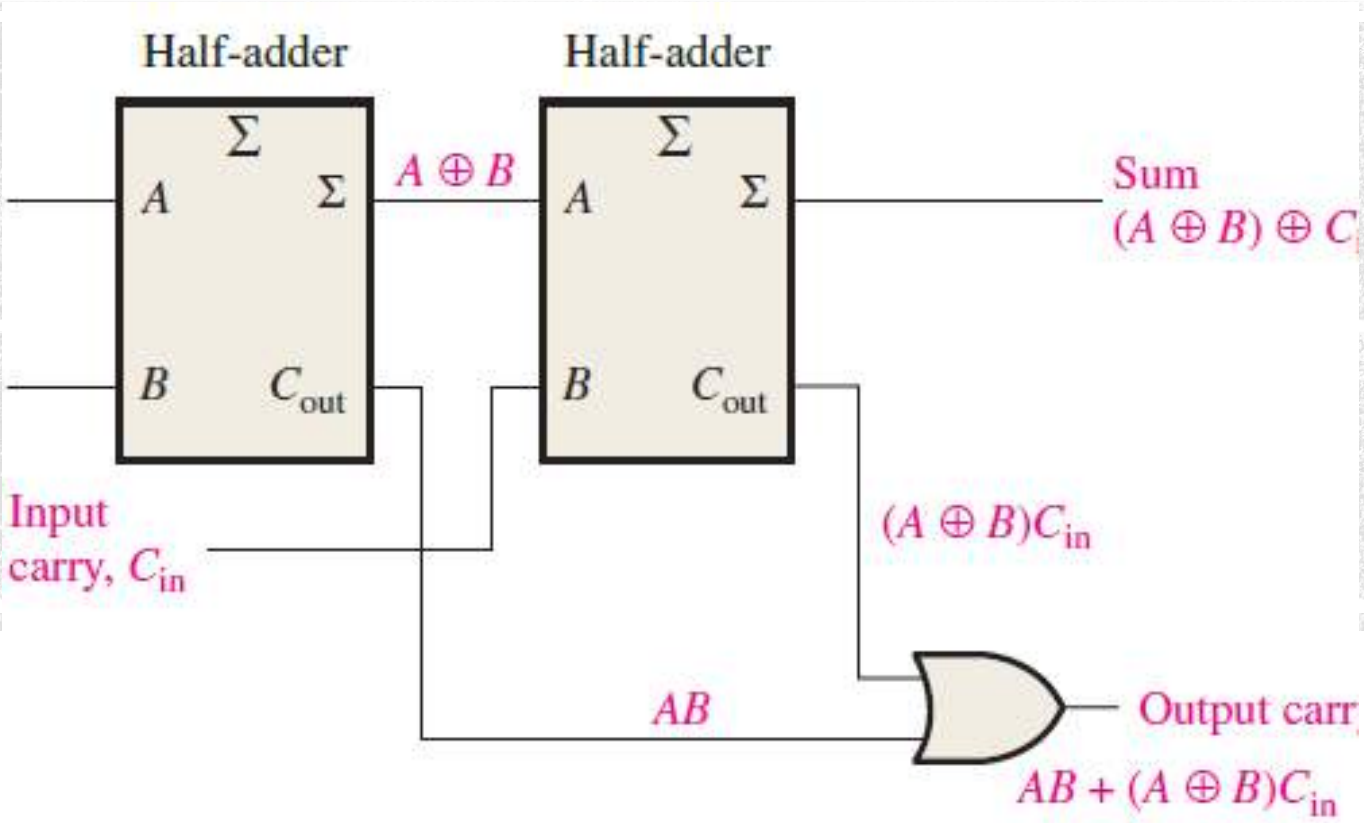


Full-adder truth table.

A	B	C_{in}	C_{out}	Σ
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0

FULL ADDER

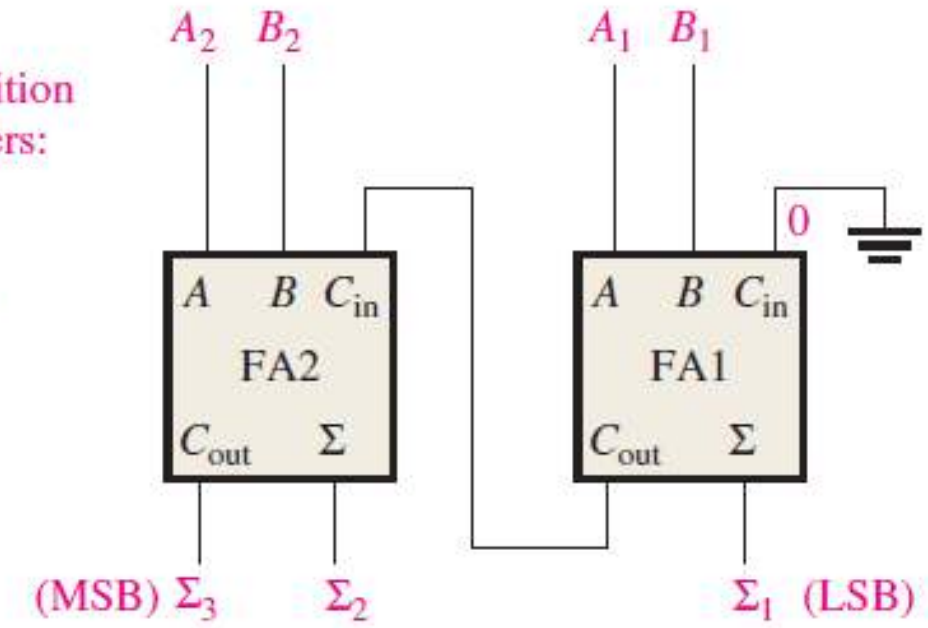
FULL ADDER WITH HALF ADDERS

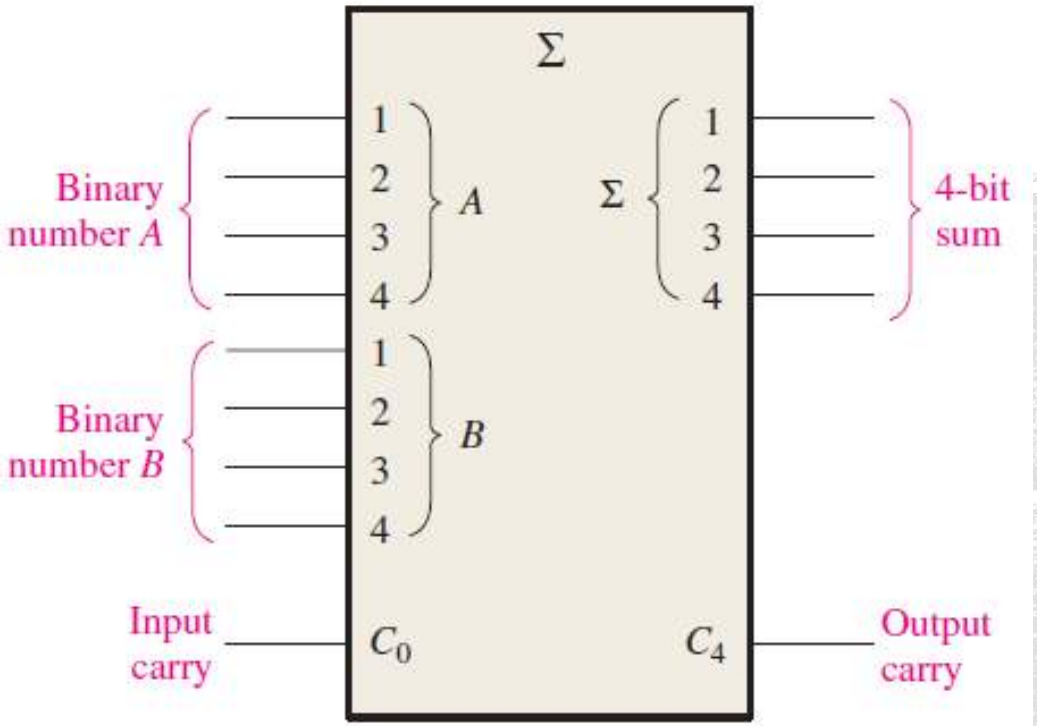


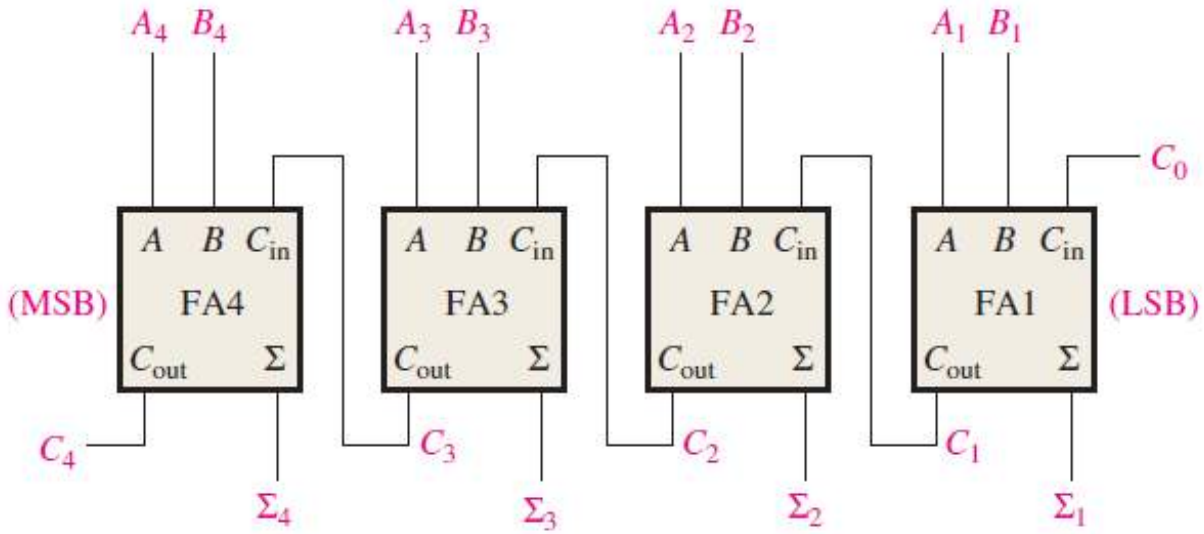
2-BIT PARALLEL ADDER

General format, addition of two 2-bit numbers:

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

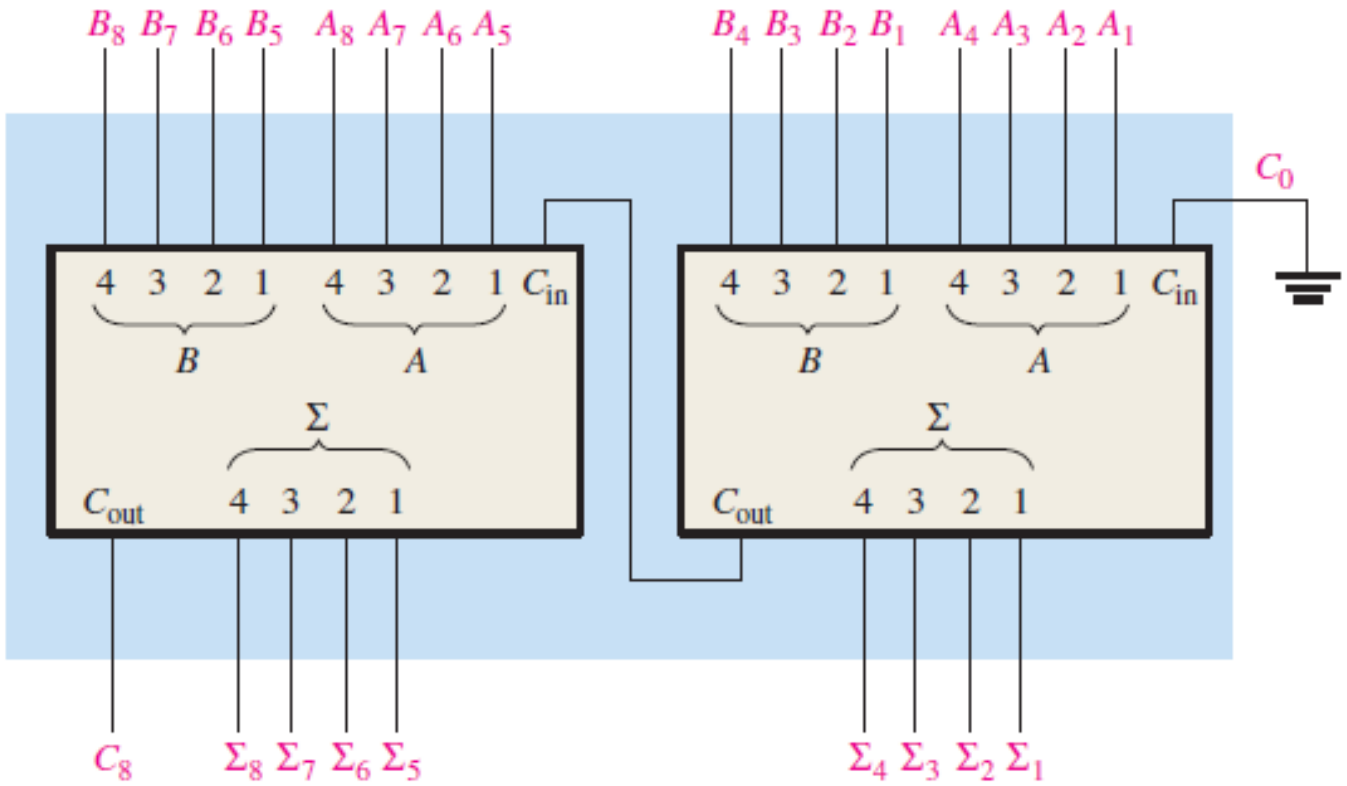




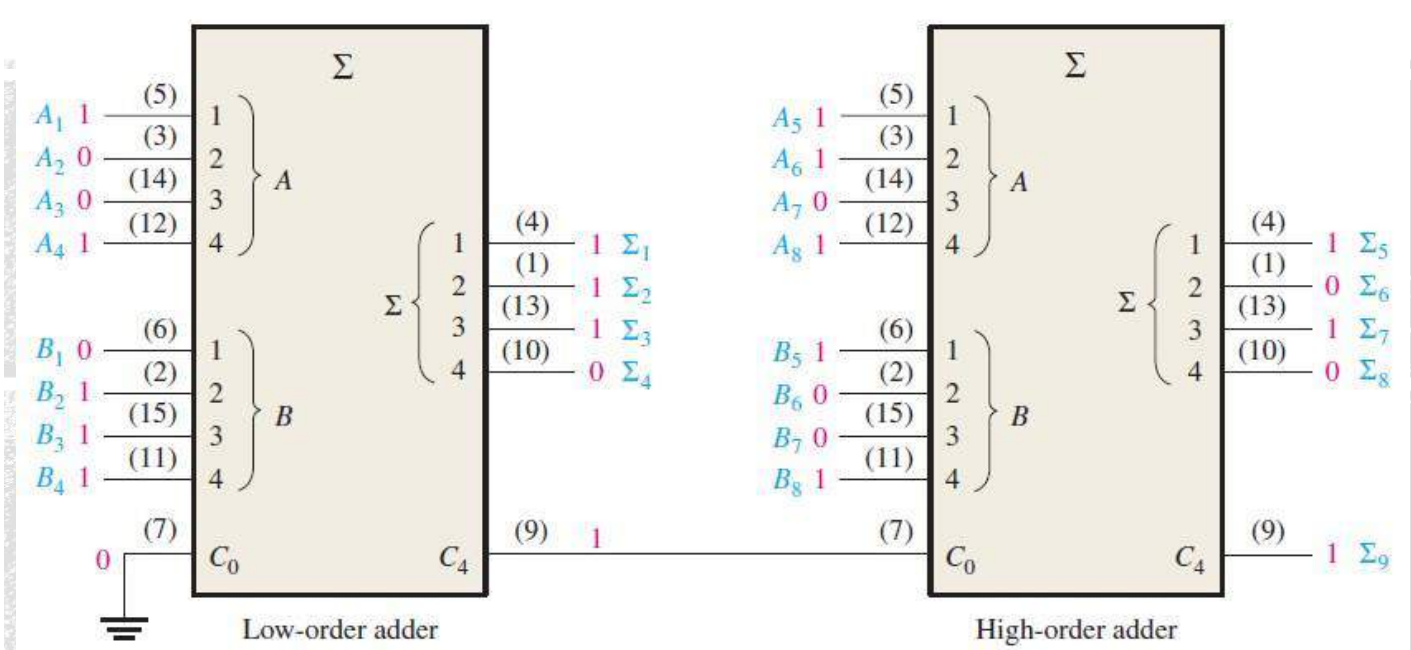


4-BIT PARALLEL ADDER

ADDER EXPANSION

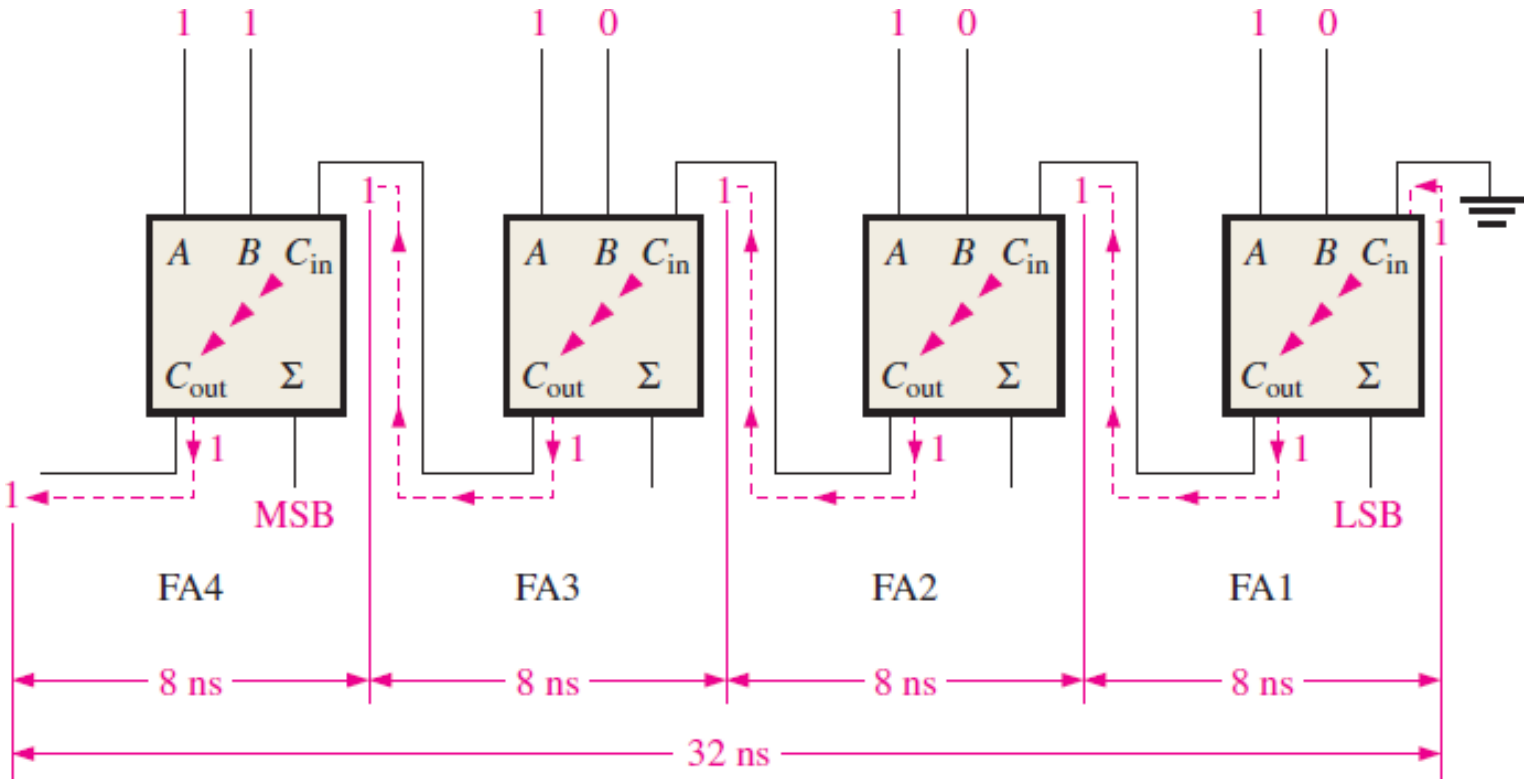


ADDER EXPANSION EXAMPLE



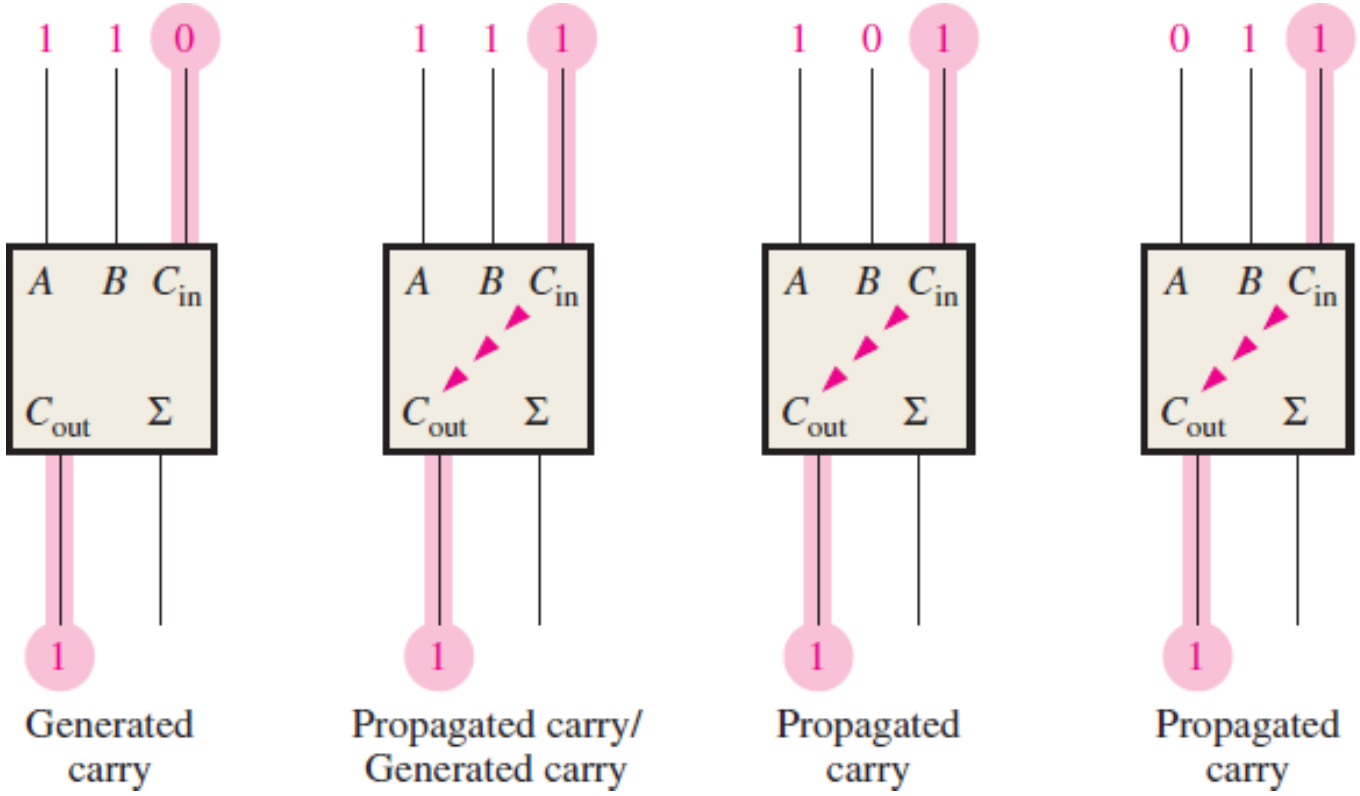
10111001
+10011110

RIPPLE CARRY ADDER



A ripple carry adder is one in which the carry output of each full-adder is connected to the carry input of the next higher order stage. The sum and output carry of any stage cannot be produced until the carry input occurs. Thus causing a delay in the overall addition process.

LOOK AHEAD CARRY ADDER



A look ahead carry adder can overcome the delay of a ripple adder. It generates a carry either by carry generation or by carry propagation.

From the conditions of carry, we can see carry is generated when both inputs are 1.

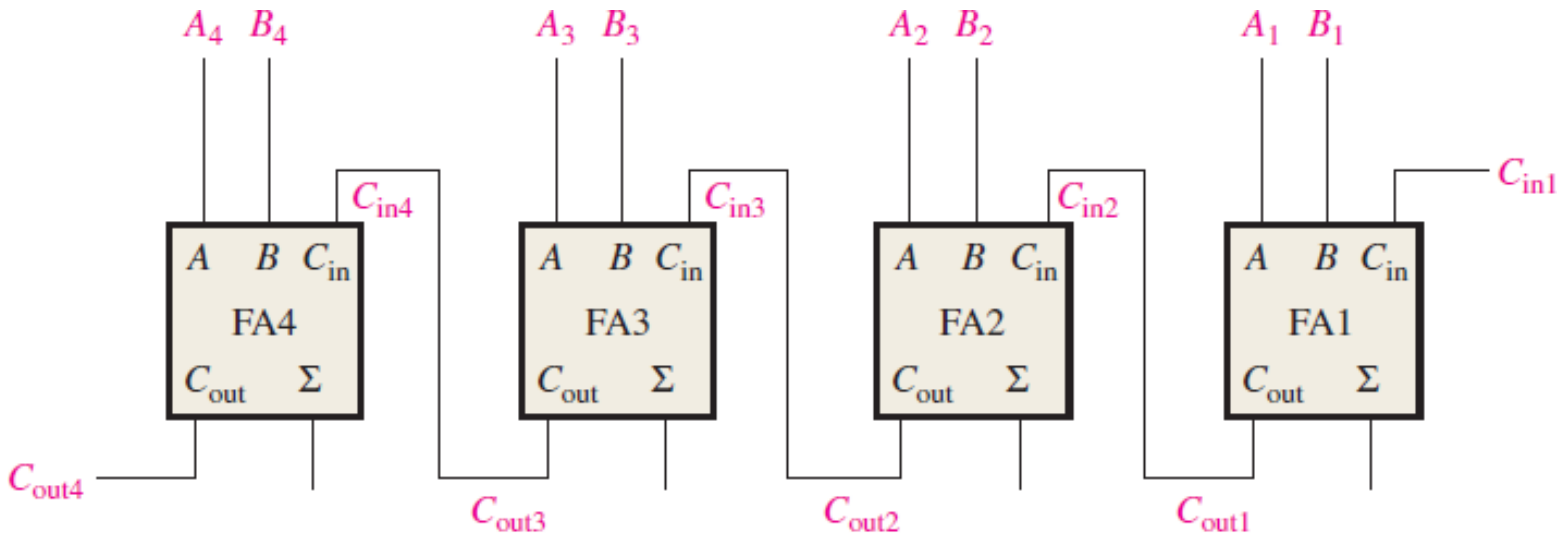
$$C_g = AB$$

And carry propagates when either of the inputs or both the inputs are 1.

$$C_p = A + B$$

$$C_{out} = C_g + C_p C_{in}$$

LOOK AHEAD CARRY ADDER EXAMPLE



Full-adder 4

$C_{g4} = A_4B_4$

$C_{p4} = A_4 + B_4$

Full-adder 3

$C_{g3} = A_3B_3$

$C_{p3} = A_3 + B_3$

Full-adder 2

$C_{g2} = A_2B_2$

$C_{p2} = A_2 + B_2$

Full-adder 1

$C_{g1} = A_1B_1$

$C_{p1} = A_1 + B_1$

The carry output is dependent on the generated carry, propagation carry and carry input(C_{IN}). The carry generation and carry propagation for each stage are immediately available as soon as the inputs are available.

CARRY OUTPUT EQUATION

Full-adder 1:

$$C_{out1} = C_{g1} + C_{p1}C_{in1}$$

Full-adder 2:

$$C_{in2} = C_{out1}$$

$$C_{out2} = C_{g2} + C_{p2}C_{in2} = C_{g2} + C_{p2}C_{out1} = C_{g2} + C_{p2}(C_{g1} + C_{p1}C_{in1})$$
$$= C_{g2} + C_{p2}C_{g1} + C_{p2}C_{p1}C_{in1}$$

Full-adder 3:

$$C_{in3} = C_{out2}$$

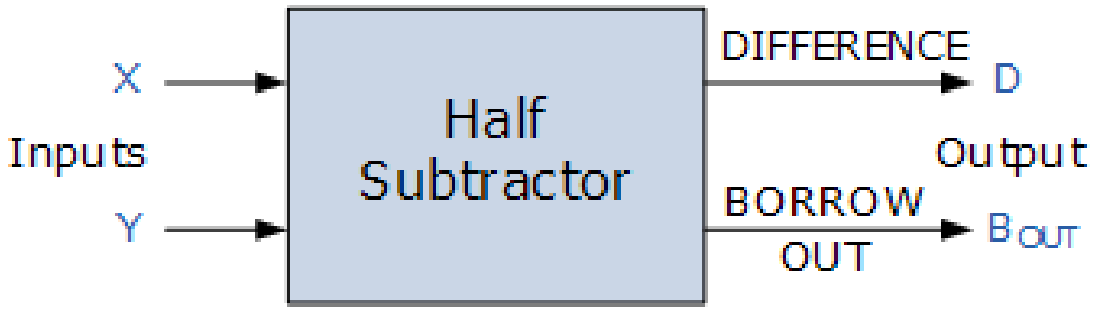
$$C_{out3} = C_{g3} + C_{p3}C_{in3} = C_{g3} + C_{p3}C_{out2} = C_{g3} + C_{p3}(C_{g2} + C_{p2}C_{g1} + C_{p2}C_{p1}C_{in1})$$
$$= C_{g3} + C_{p3}C_{g2} + C_{p3}C_{p2}C_{g1} + C_{p3}C_{p2}C_{p1}C_{in1}$$

Full-adder 4:

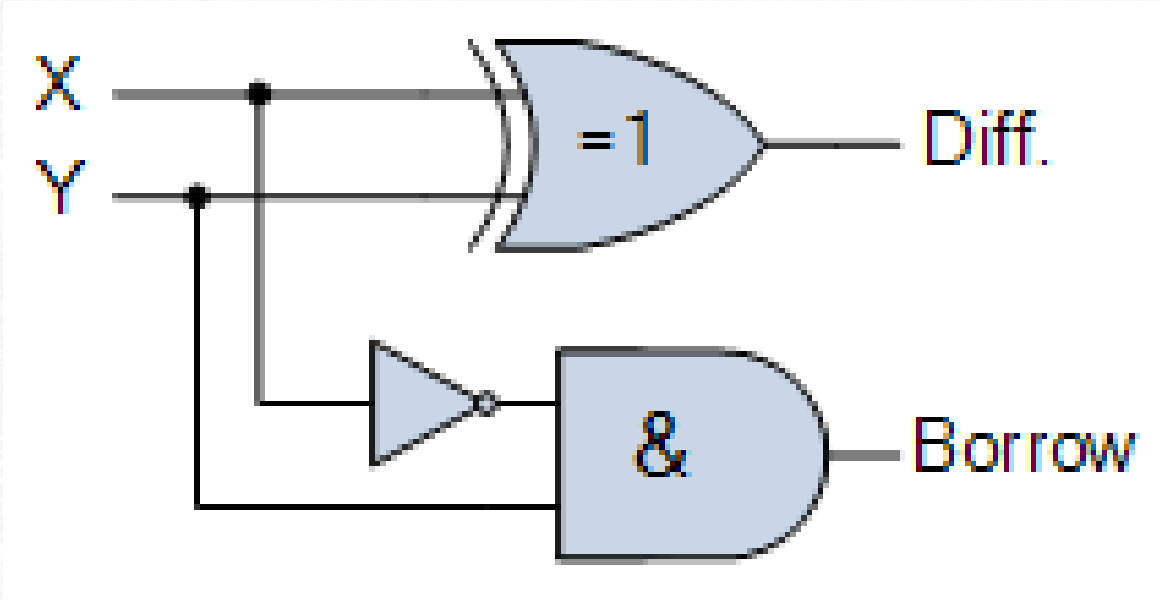
$$C_{in4} = C_{out3}$$

$$C_{out4} = C_{g4} + C_{p4}C_{in4} = C_{g4} + C_{p4}C_{out3}$$
$$= C_{g4} + C_{p4}(C_{g3} + C_{p3}C_{g2} + C_{p3}C_{p2}C_{g1} + C_{p3}C_{p2}C_{p1}C_{in1})$$
$$= C_{g4} + C_{p4}C_{g3} + C_{p4}C_{p3}C_{g2} + C_{p4}C_{p3}C_{p2}C_{g1} + C_{p4}C_{p3}C_{p2}C_{p1}C_{in1}$$

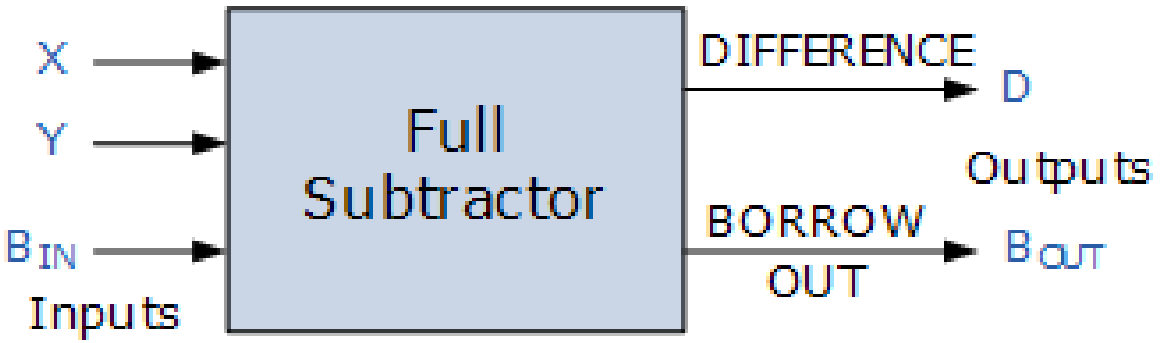
HALF SUBTRACTOR



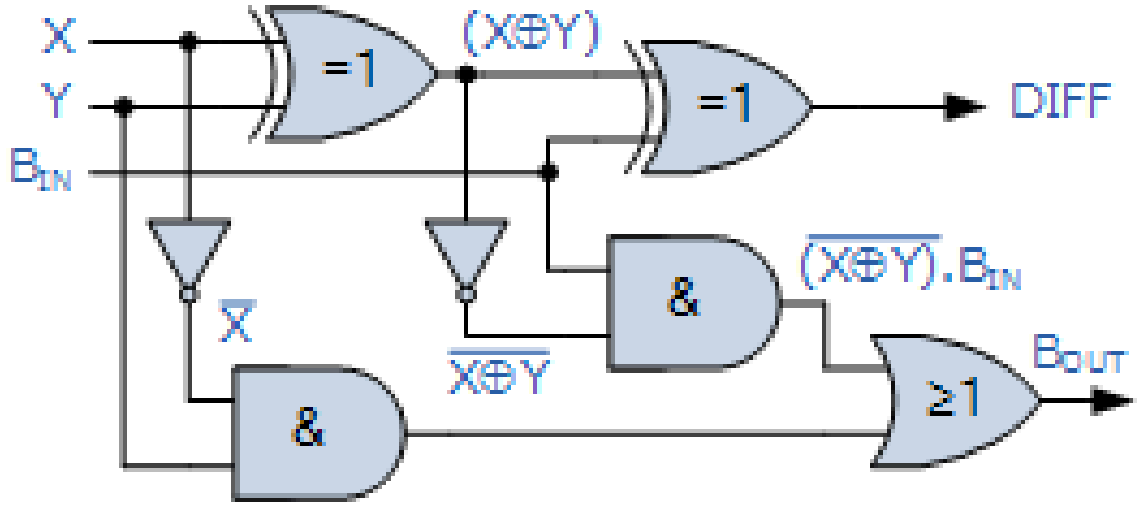
Y	X	DIFFERENCE	BORROW
0	0	0	0
0	1	1	0
1	0	1	1
1	1	0	0



FULL SUBTRACTOR



B-in	Y	X	Diff.	B-out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	1
0	1	1	0	0
1	0	0	1	1
1	0	1	0	0
1	1	0	0	1
1	1	1	1	1



The Digital Comparator

Another common and very useful combinational logic circuit is that of the **Digital Comparator** circuit. Digital or Binary Comparators are made up from standard AND, NOR and NOT gates that compare the digital signals present at their input terminals and produce an output depending upon the condition of those inputs.

For example, along with being able to add and subtract binary numbers we need to be able to compare them and determine whether the value of input A is greater than, smaller than or equal to the value at input B etc. The digital comparator accomplishes this using several logic gates that operate on the principles of Boolean Algebra. There are two main types of Digital Comparator available and these are.

1. **Identity Comparator** – an *Identity Comparator* is a digital comparator that has only one output terminal for when A = B either "HIGH" A = B = 1 or "LOW" A = B = 0

2. **Magnitude Comparator** – a *Magnitude Comparator* is a type of digital comparator that has three output terminals, one each for equality, **A = B** greater than, **A > B** and less than **A < B**

The purpose of a **Digital Comparator** is to compare a set of variables or unknown numbers, for example A (A1, A2, A3, …. An, etc) against that of a constant or unknown value such as B (B1, B2, B3, …. Bn, etc) and produce an output condition or flag depending upon the result of the comparison. For example, a magnitude comparator of two 1-bits, (A and B) inputs would produce the following three output conditions when compared to each other.

$$A > B, \quad A = B, \quad A < B$$

Which means: A is greater than B, A is equal to B, and A is less than B

1-bit Digital Magnitude Comparator:

Truth Table:

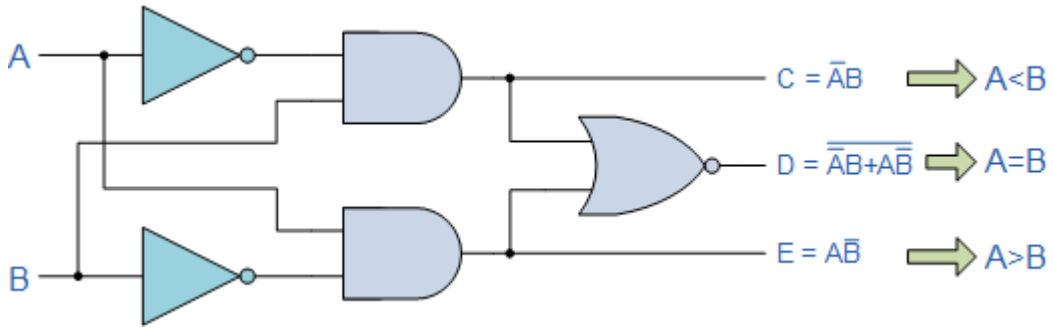
Inputs		Outputs		
B	A	A > B	A = B	A < B
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

Logic Expression:

$X_0 = A_0B_0+A_0'B_0'$

$(A=B)=X_0$, $(A<B)= A_0'B_0$, $(A>B)= A_0B_0'$

Logic Diagram:



2-bit Digital Magnitude Comparator:

Logic Expression: $X_n = A_nB_n + A_n'B_n'$ $(A=B) = X_1X_0$

$(A<B) = A_1'B_1 + X_1A_0'B_0$ $(A>B) = A_1B_1' + X_1A_0B_0'$

Block Diagram:

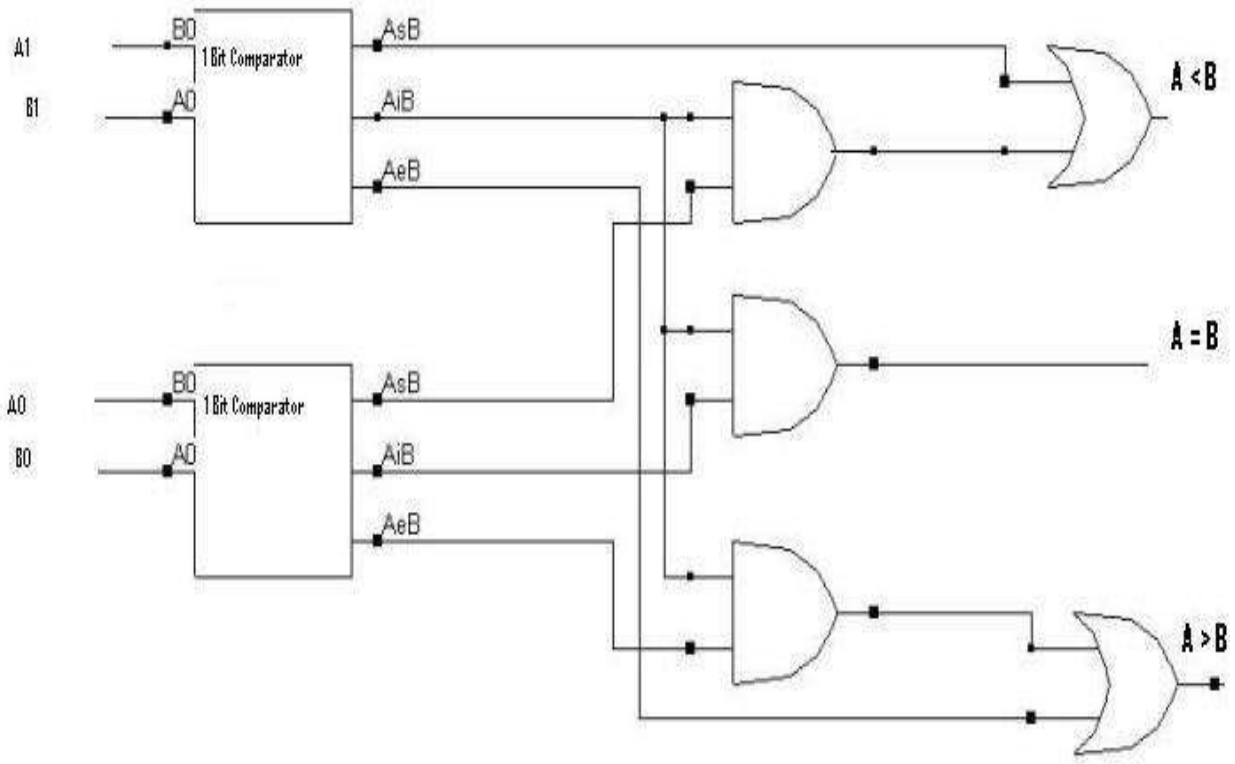


Figure 2: 2 Bit Comparator

3-bit Digital Magnitude Comparator:

Logic Expression:

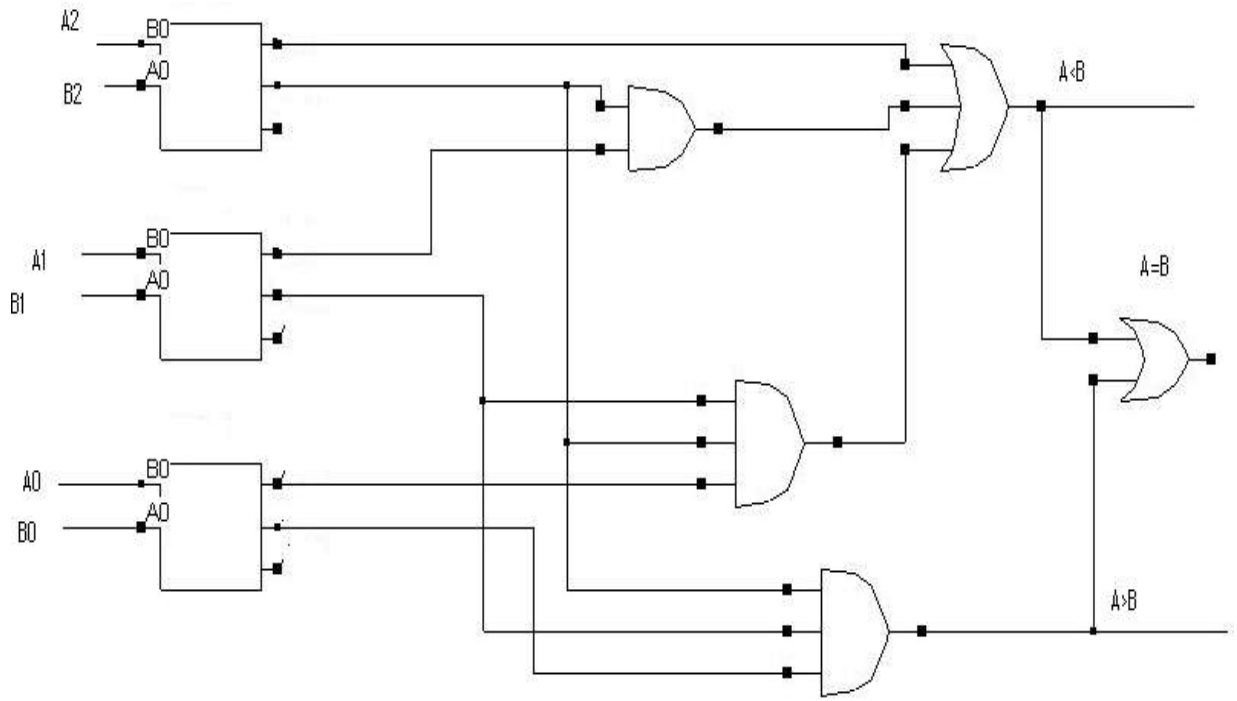
$X_n = A_nB_n + A_n'B_n'$

$(A=B) = X_2X_1X_0$

$(A<B) = A_2'B_2 + X_2A_1'B_1 + X_2X_1A_0'B_0$

$(A>B) = A_2B_2' + X_2A_1B_1' + X_2X_1A_0B_0'$

Block Diagram:



5-bit Digital Magnitude Comparator:

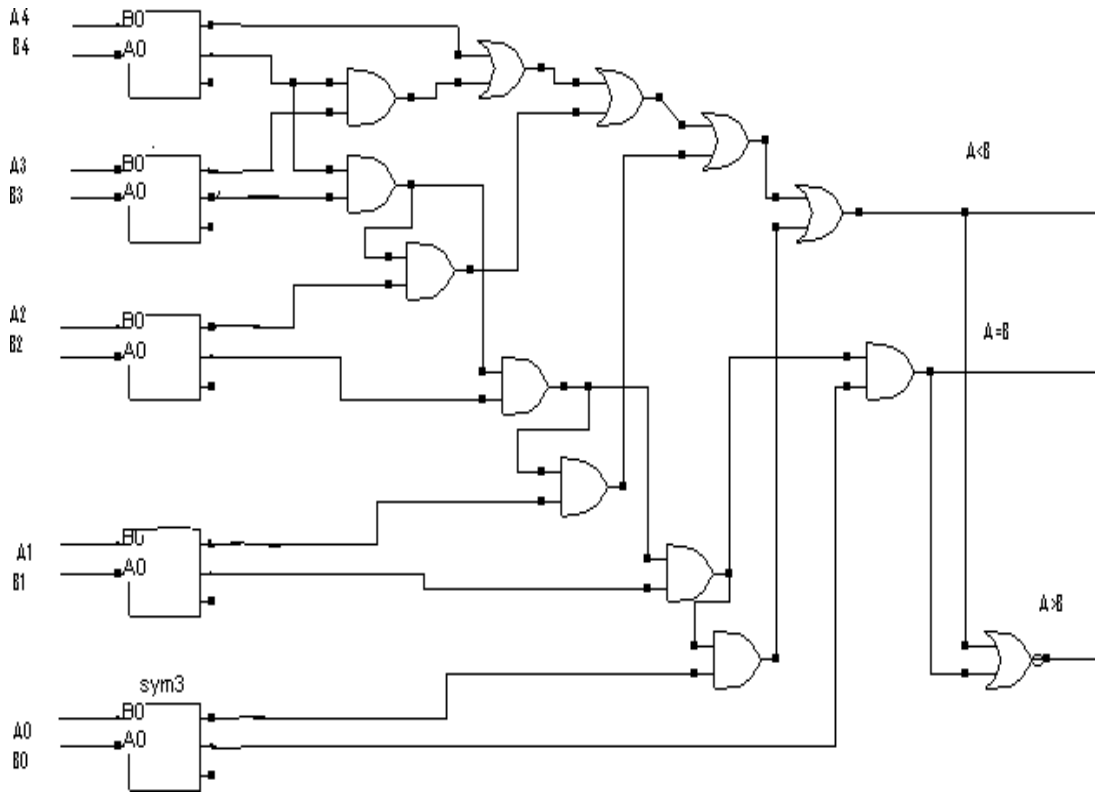
Logic Expression:

$X_n = A_nB_n + A_n'B_n'$

$(A=B)=X_4X_3X_2X_1X_0$

$(A<B)=A_4'B_4+X_4A_3'B_3+X_4X_3A_2'B_2+X_4X_3X_2A_1'B_1+X_4X_3X_2X_1A_0'B_0$

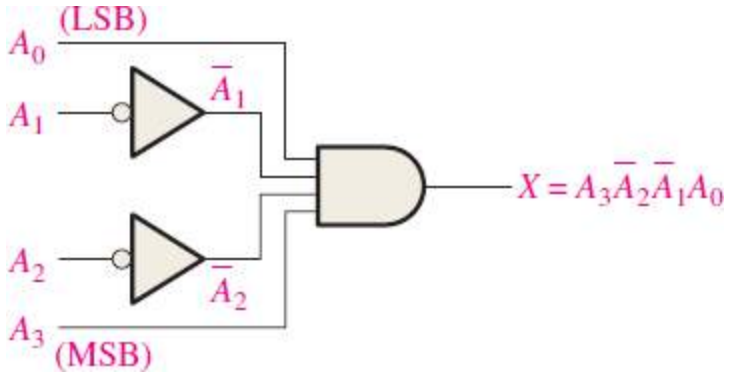
$(A>B)=A_4B_4'+X_4A_3B_3'+X_4X_3A_2B_2'+X_4X_3X_2A_1B_1'+X_4X_3X_2X_1A_0B_0'$



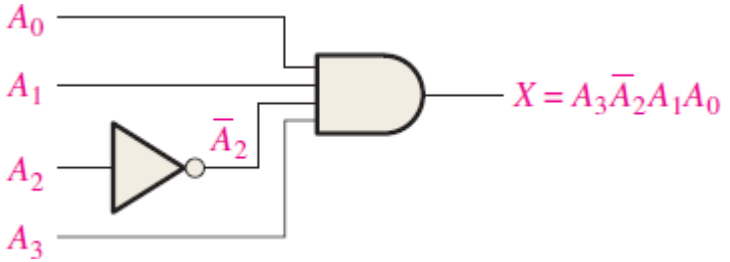
DECODERS:

A decoder is a digital circuit that detects the presence of a specified combination of bits on its inputs and indicates the presence of that code by a specified output level. In general form, a decoder has n-input lines to handle n-bits and 2^n outputs to indicate 2^n combinations.

Determine the logic required to decode the binary number 1001 by producing a HIGH level on the output.



Determine the logic required to decode the binary number 1011 by producing a HIGH level on the output.

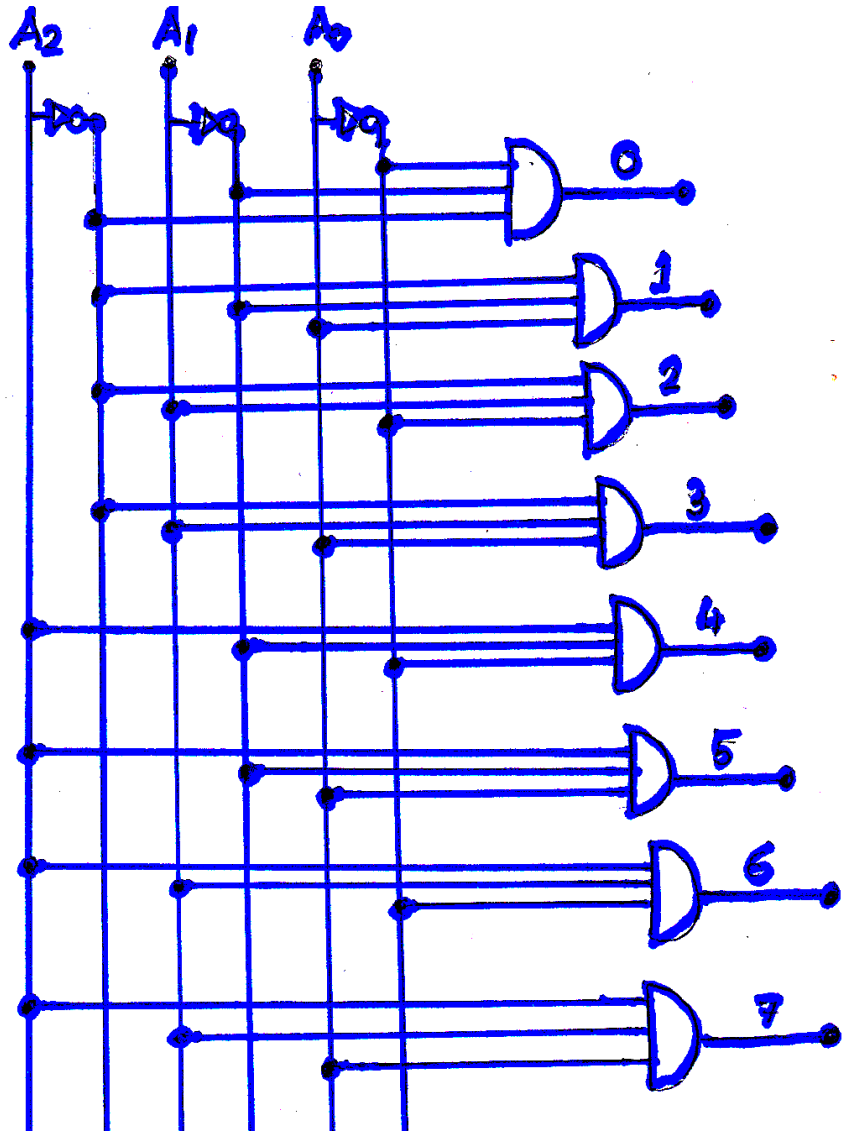


Determine the logic required to decode the binary number 1101 and produce an active-LOW output.

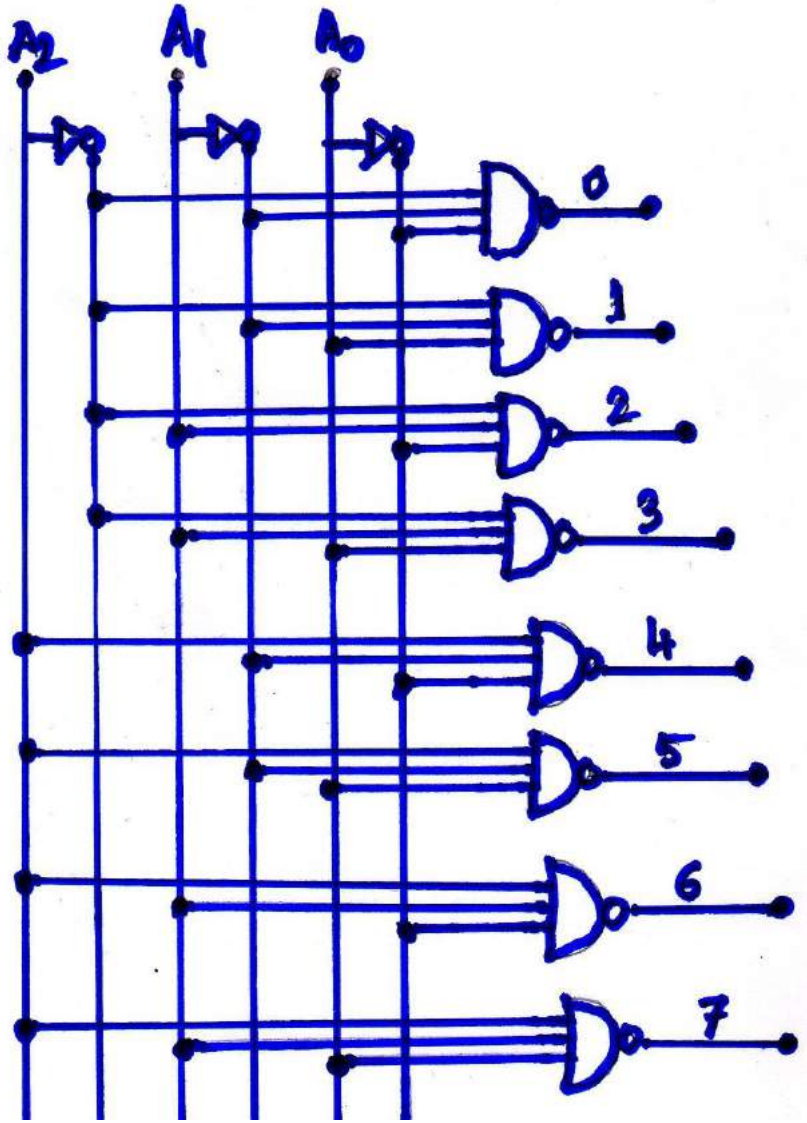
Determine the logic required to decode the binary number 1110 and produce an active-LOW output.

3-Line to 8-Line Decoding Function for Active-HIGH and Active-LOW

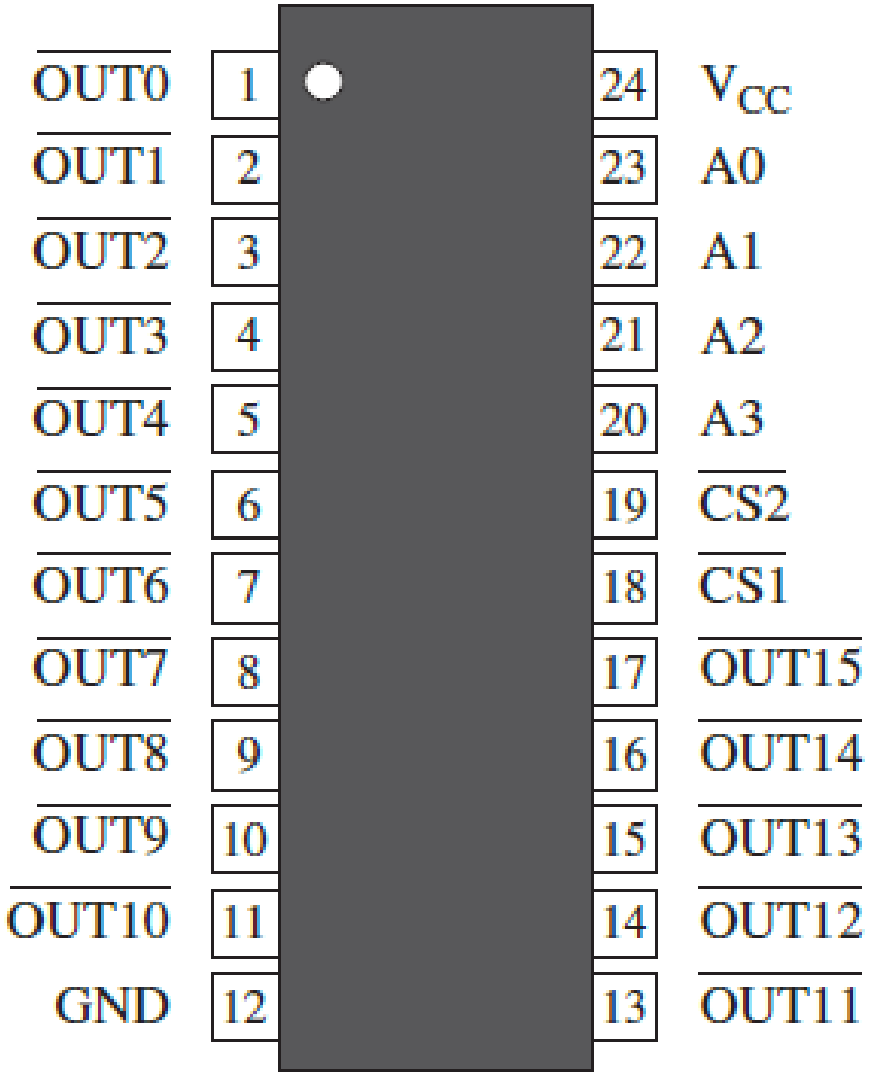
Decimal Digit	Binary Inputs			Decoding Function Active-HIGH	Decoding Function Active-LOW
	A_2	A_1	A_0		
0	0	0	0	$\overline{A_2}A_1A_0$	$\overline{\overline{A_2}A_1A_0}$
1	0	0	1	$\overline{A_2}\overline{A_1}A_0$	$\overline{\overline{A_2}\overline{A_1}A_0}$
2	0	1	0	$\overline{A_2}A_1\overline{A_0}$	$\overline{\overline{A_2}A_1\overline{A_0}}$
3	0	1	1	$\overline{A_2}A_1A_0$	$\overline{\overline{A_2}A_1A_0}$
4	1	0	0	$A_2\overline{A_1A_0}$	$\overline{A_2\overline{A_1A_0}}$
5	1	0	1	$A_2\overline{A_1}A_0$	$\overline{A_2\overline{A_1}A_0}$
6	1	1	0	$A_2A_1\overline{A_0}$	$\overline{A_2A_1\overline{A_0}}$
7	1	1	1	$A_2A_1A_0$	$\overline{A_2A_1A_0}$



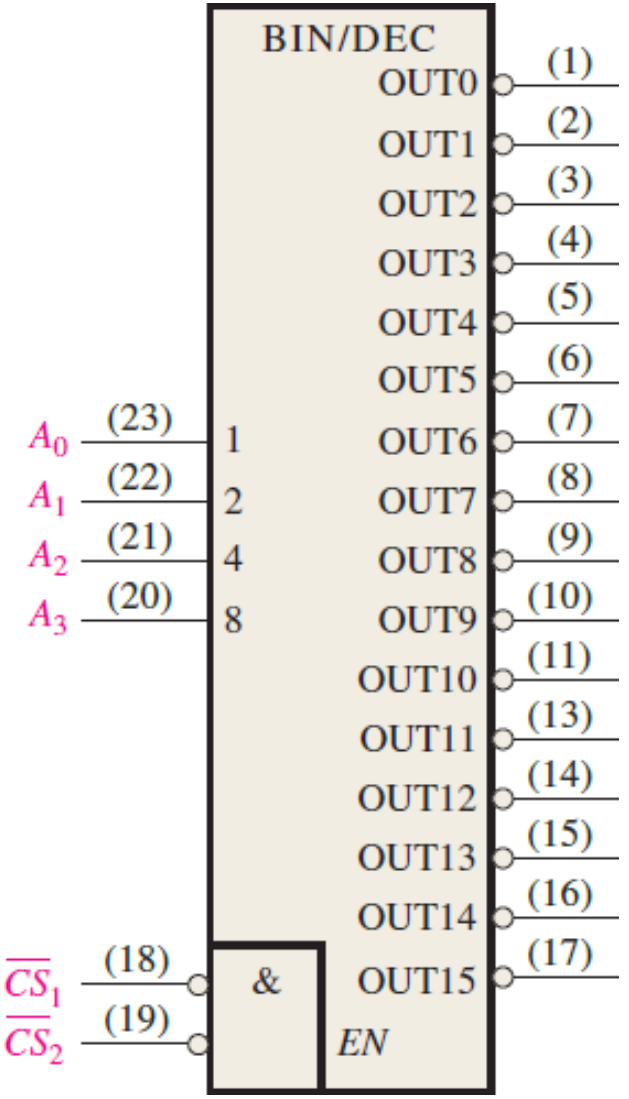
Active-HIGH output



Active-LOW output



74HC154 Pin Diagram

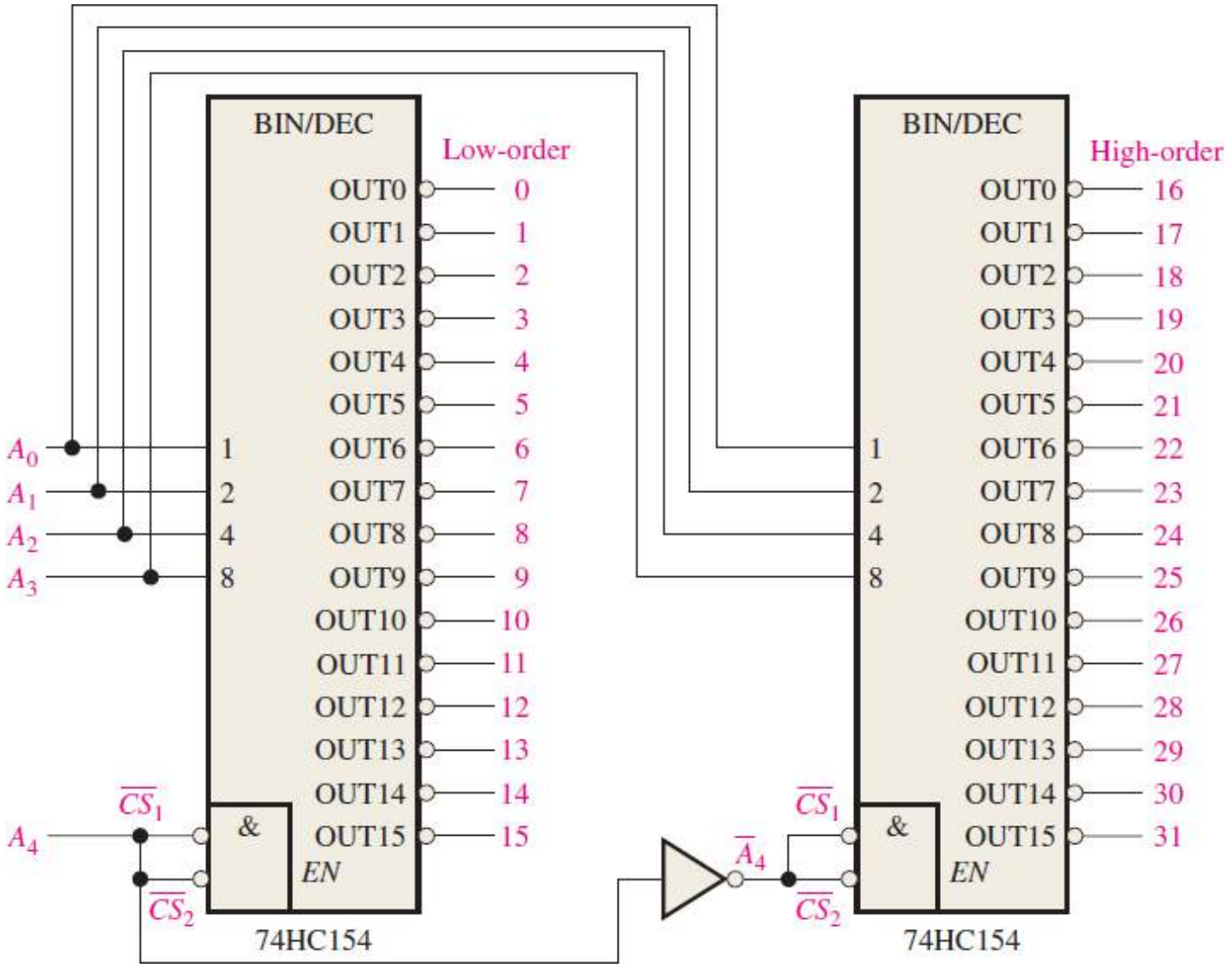


7CHC154 Logic Symbol

Design a 5- bit Decoder using 74HC154.

Solution:

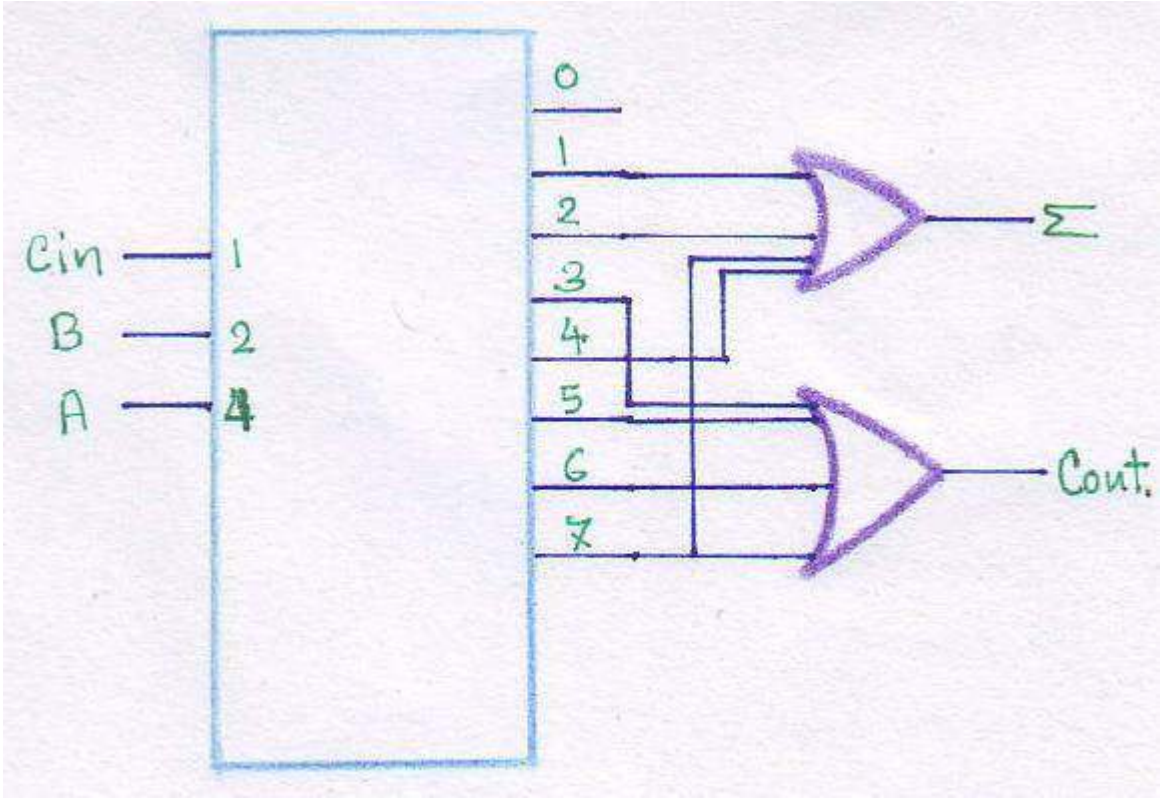
Since the 74HC154 can handle only 4-bits, two decoders must be used to form a 5-bit expansion. The fifth bit, A_4, is connected to the chip select inputs, $\overline{CS_1}$ and $\overline{CS_2}$, of one decoder, and $\overline{A_4}$, is connected to the chip select inputs of the other decoder. When the decimal number is 15 or less, $A_4 = 0$, the lower order decoder is enabled, and the high-order decoder is disabled. When the decimal number is greater than 15, $A_4 = 1$ and $\overline{A_4}$ =0, the higher order is enabled, and the lower order decoder is disabled.



Implement a Full-Adder using 3-bit decoder.

A	B	Cin	Σ	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Full-Adder TruthTable



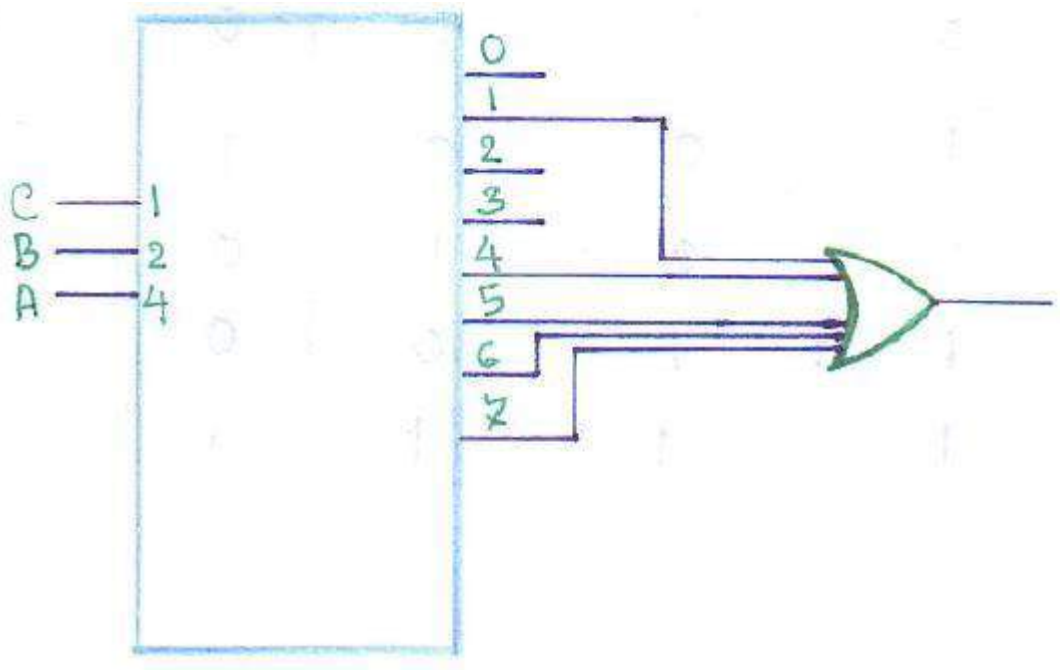
Full-Adder Logical Circuit with Decoder

Implement the following function a 3-bit Decoder and necessary gates.

$$F = A + BC$$

$$F = A + BC$$

$$\therefore F = ABC + ABC + AB\,C + AB\,C + ABC$$



Implement the following function a 4-bit Decoder and necessary gates.

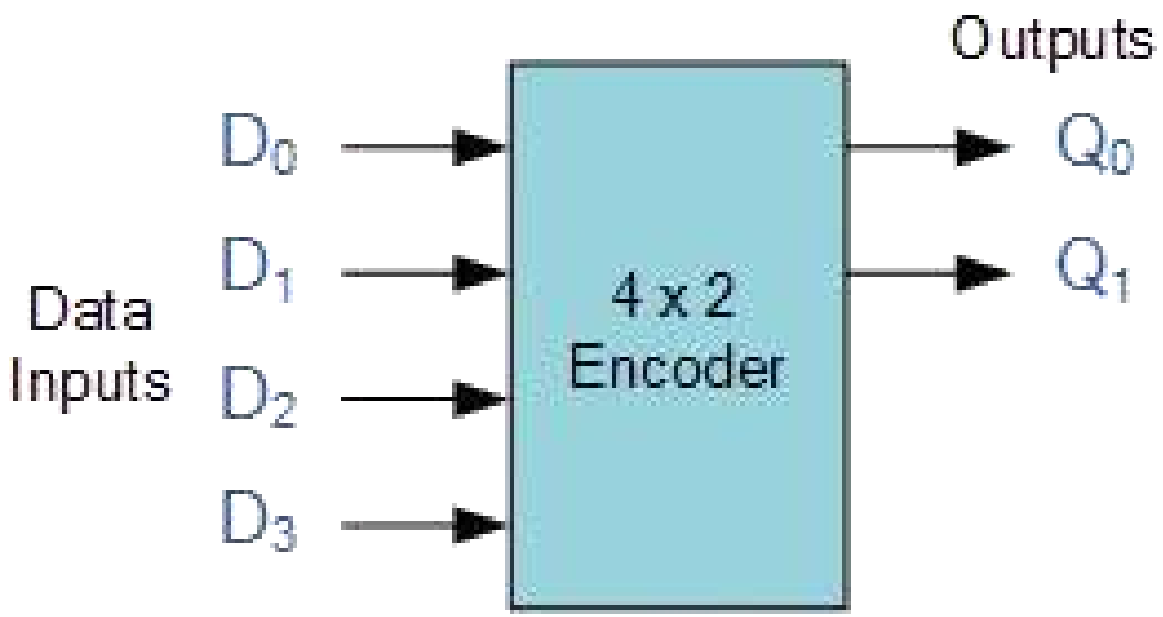
$$F = AD + BC$$

Implement the following function a 3-bit Decoder and necessary gates.

- Implement the following function a $F = B + AC$
- 3-bit Decoder and necessary gates.
  - $F = A + CB$
- Implement a BCD to 7-segment display decoder/driver.

ENCODERS:

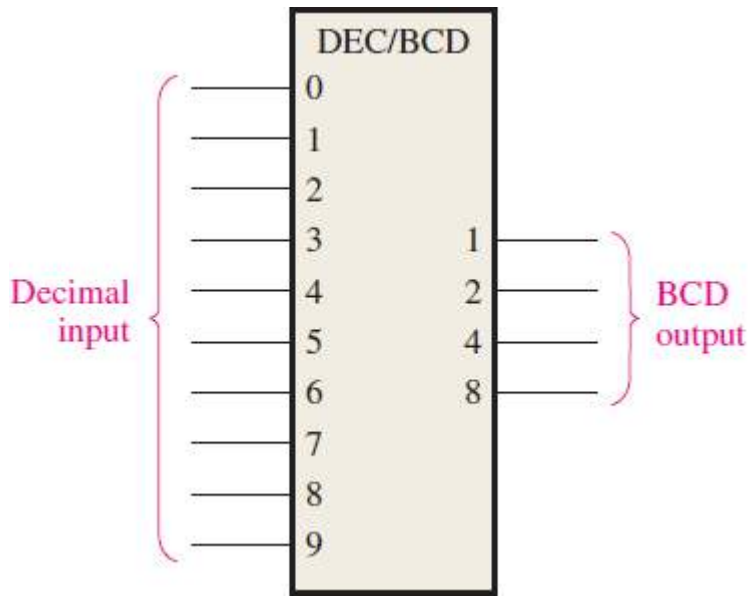
An encoder is a combinational circuit that detects the presence of an active level in its input and converts it to a coded output. E.g. Decimal to Binary encoders, Decimal to BCD encoders, Octal to Binary encoders, Priority Encoders, Irregular Sequence Encoder etc.



Inputs				Outputs	
D_3	D_2	D_1	D_0	Q_1	Q_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1
0	0	0	0	x	x

4X2 Decimal to Binary Encoder

Decimal to BCD Encoder: A decimal to BCD encoder encodes the decimal numbers to its equivalent BCD codes.



Decimal Digit	BCD Code A_3	A_2	A_1	A_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

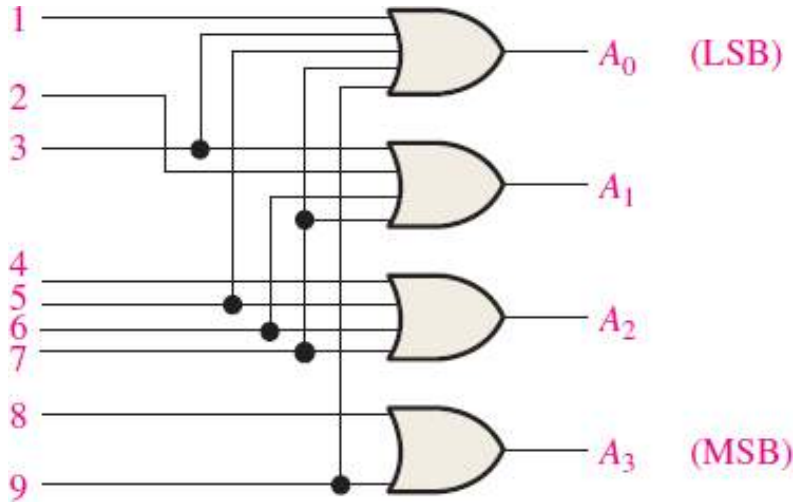
Functions for the output pins:

$$A_3 = 8 + 9$$

$$A_2 = 4 + 5 + 6 + 7$$

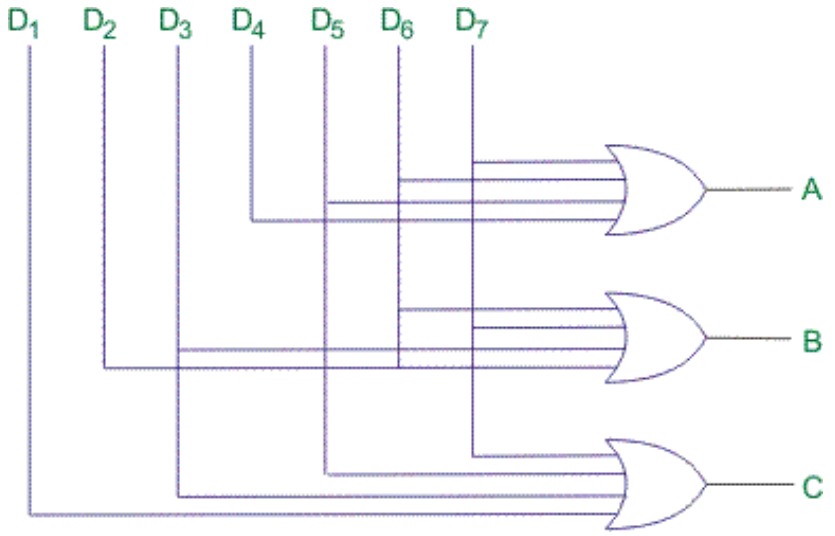
$$A_1 = 2 + 6 + 7$$

$$A_0 = 1 + 3 + 5 + 7 + 9$$



Octal to Binary Encoder: An Octal to Binary encoder encodes an octal number to its equivalent binary code.

Octal Numbar	INPUTS								OUTPUTS		
	D0	D1	D2	D3	D4	D5	D6	D7	a	b	c
0	1	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	1
2	0	0	1	0	0	0	0	0	0	1	0
3	0	0	0	1	0	0	0	0	0	1	1
4	0	0	0	0	1	0	0	0	1	0	0
5	0	0	0	0	0	1	0	0	1	0	1
6	0	0	0	0	0	0	1	0	1	1	0
7	0	0	0	0	0	0	0	1	1	1	1



Functions for the output pins:

$$A_3 = 4 + 5 + 6 + 7$$

$$A_1 = 2 + 3 + 6 + 7$$

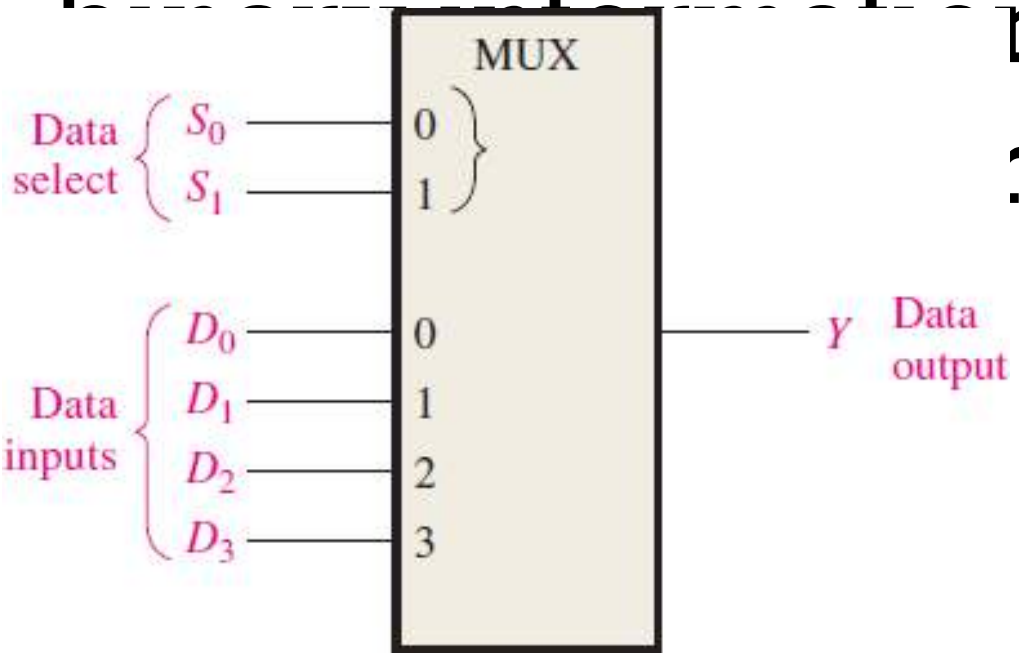
$$A_0 = 1 + 3 + 5 + 7$$

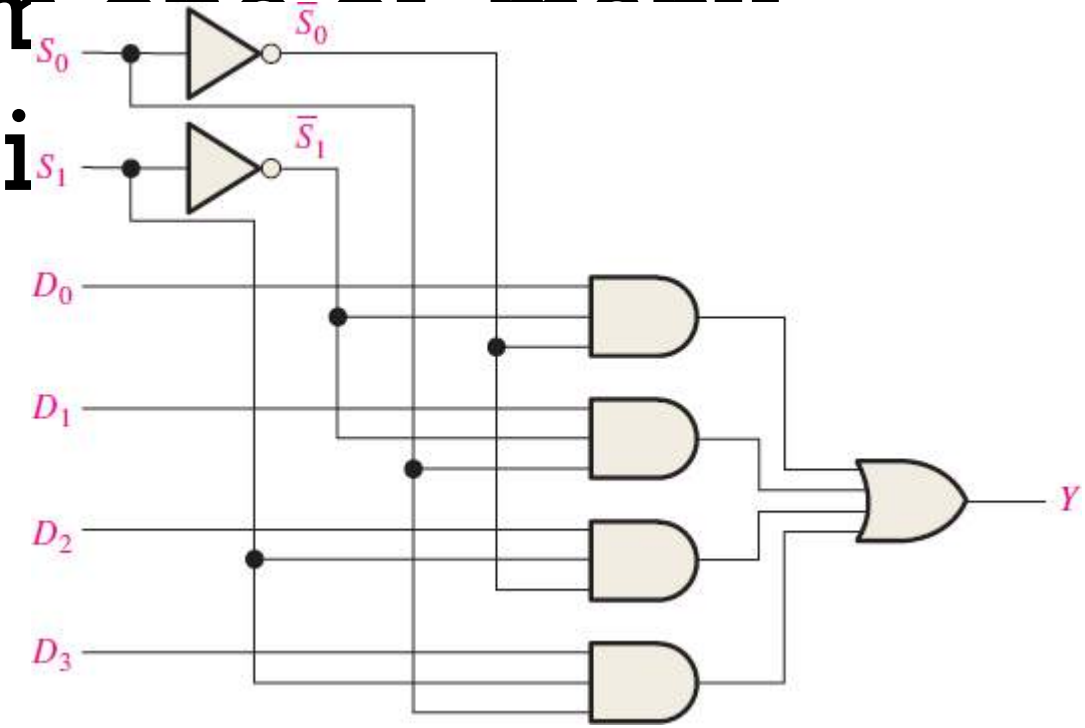
Priority Encoder: A priority encoder also encodes a given input however, the flexibility of a priority encoder is that, we can assign the priority of inputs. E.g Highest Priority Encoder, Lowest Priority Encoders, Irregular Sequence Encoders.

1. Design a 4X2 Highest Priority Encoder.
2. Design a 4X2 Lowest Priority Encoder.
3. Design a Decimal to BCD Highest Priority Encoder.
4. Design a Decimal to BCD Lowest Priority Encoder.
5. Design a Decimal to BCD Irregular Sequence Encoder with priority sequence of 1,5,6,7,0,4,3,2,8,9.

Follow Class Lecture

combinational circuit which selects
binary information from one of many
input lines and directs it to a single output line.



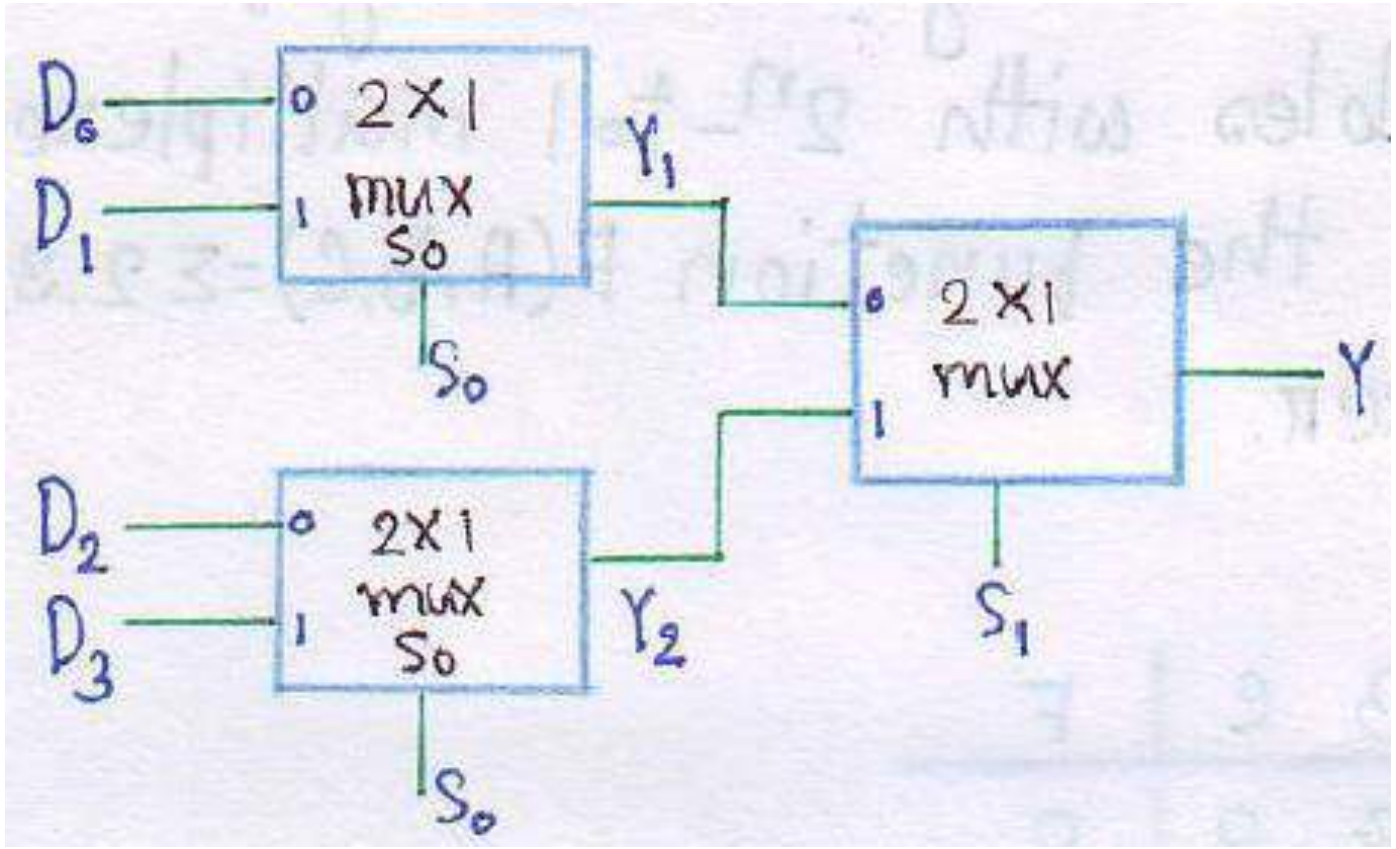


Data-Select Inputs		
S_1	S_0	Input Selected
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

$$Y = D_0\overline{S}_1\overline{S}_0 + D_1\overline{S}_1S_0 + D_2S_1\overline{S}_0 + D_3S_1S_0$$

Cascading Lower Input MUX to Design Higher Input MUX

S_1	S_0	Y_1	Y_2	Y
0	0	D_0	D_2	D_0
0	1	D_1	D_3	D_1
1	0	D_0	D_2	D_2
1	1	D_1	D_3	D_3



4X1 MUX with two 2X1 MUX

Boolean Function Implementation with MUX

It is possible to design a n+1 variable Function with 2^NX1 MUX.
Implement the function $F = (2, 3, 4, 6)$ using a MUX.

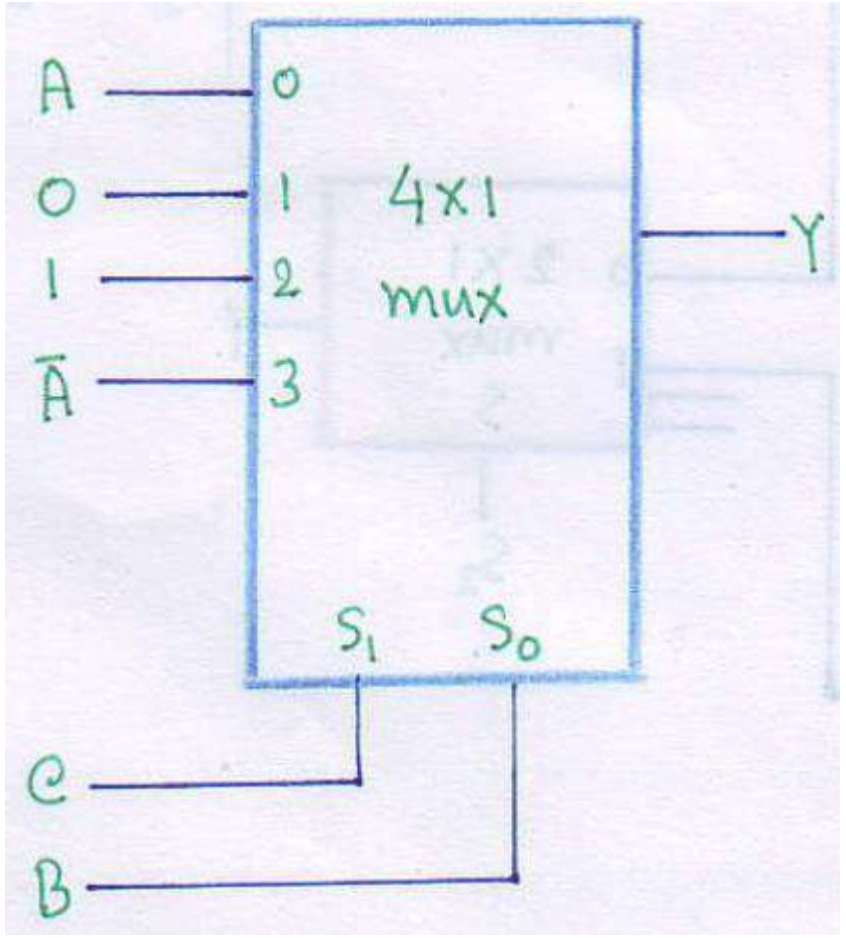
	A	B	C	F
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	0

BC=00, F=A

BC=01, F=0

BC=10, F=1

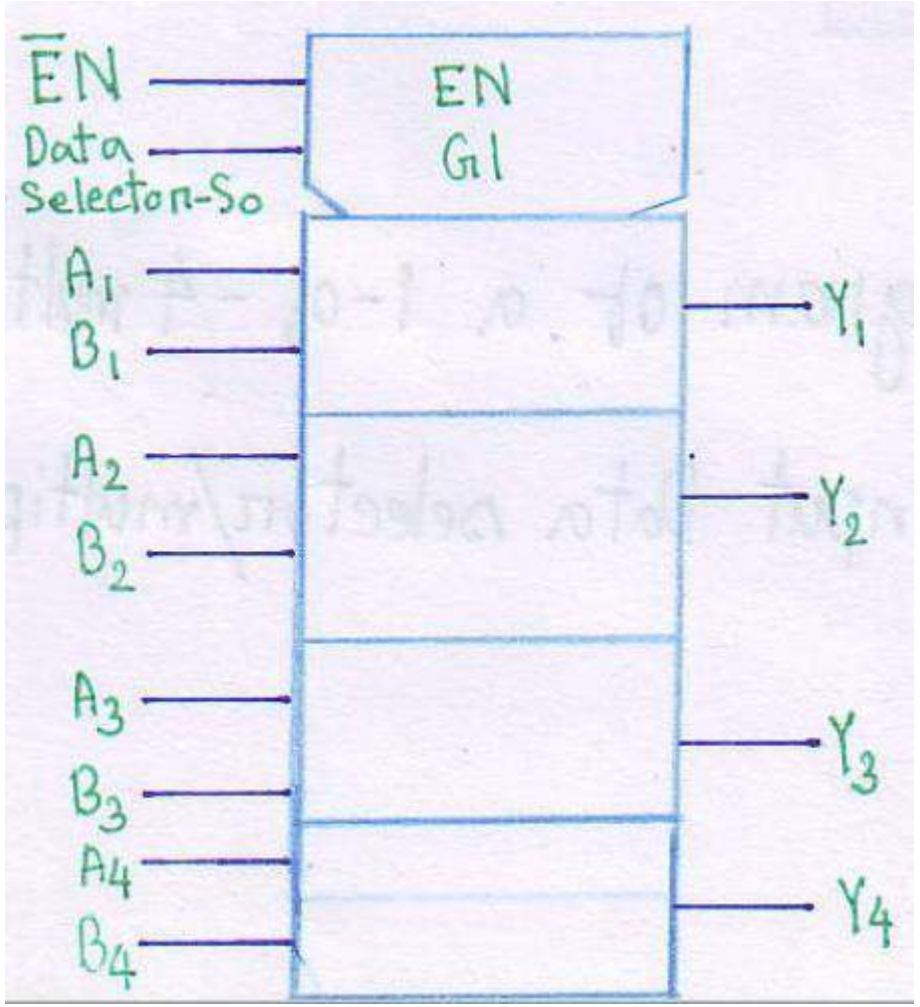
BC=11, F=$\bar{A}$



Quadruple 2-Input Data Selector Design

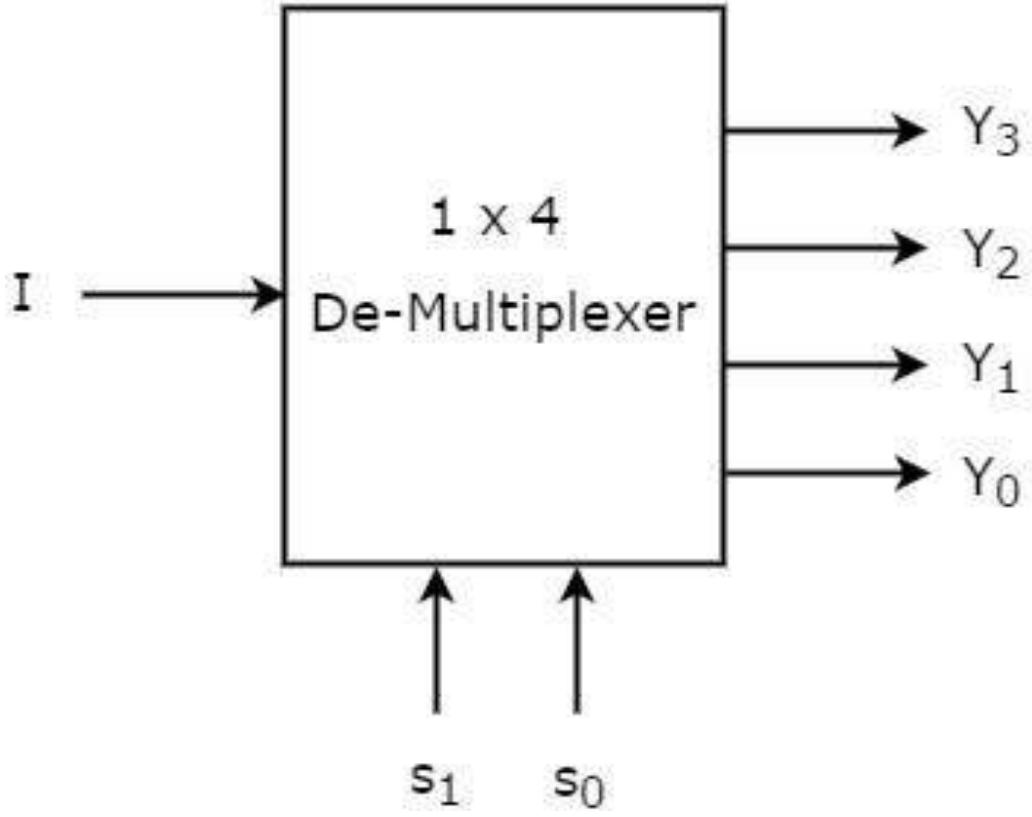
A MUX can also be considered as a data selector. A simple 2X1 MUX can be used to select between two 1-bit data. But what if, we have a 2-bit data or a 4-bit data. So we can integrate several 2X1 MUX and design multiple-bit data selectors. The figure shows the logical symbol of a 4-bit 2-input data selector with an Active-Low enable input.

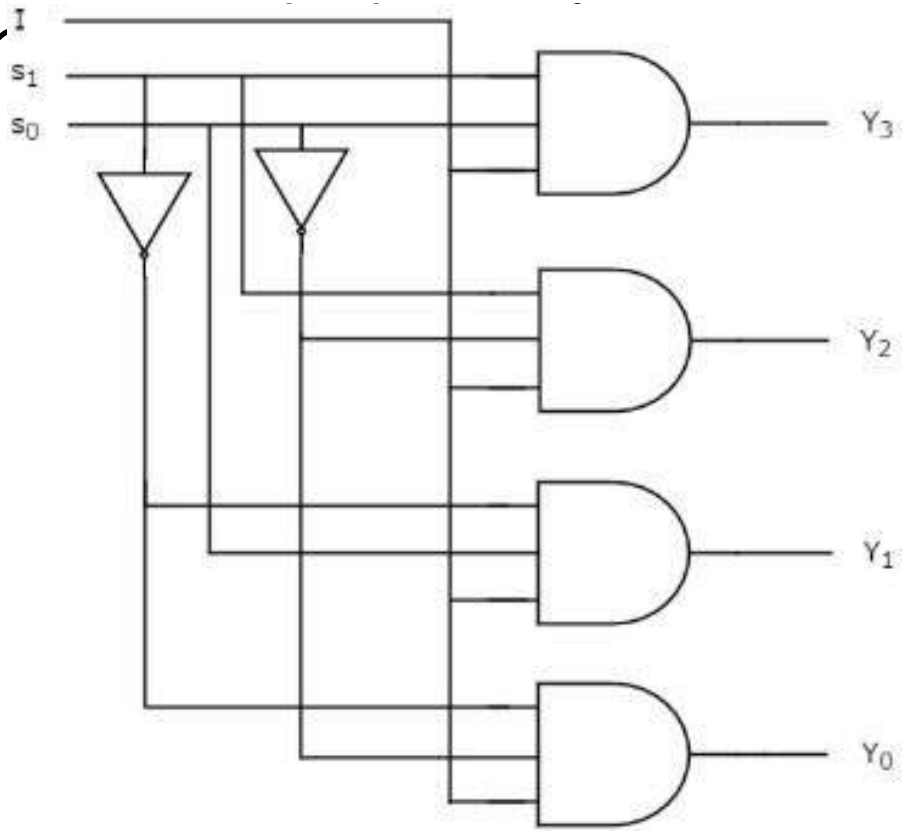
- Design the logical circuit for a 4-bit 2-input data selector.



Logic Symbol of a Quadruple 2-input Data Selector

Demultiplexer (DEMUX): A demultiplexer reverses the multiplexing function. It basically ingle lir its res





- Design a 3X8 Decoder Using 1X8 DEMUX.

Textbooks:

- [1] Thomas L. Floyd, “Digital Fundamentals” 11th edition, Prentice Hall.
- [2] M. Morris Mano, “Digital Logic & Computer Design” Prentice Hall.