Digital Systems

18B11EC213

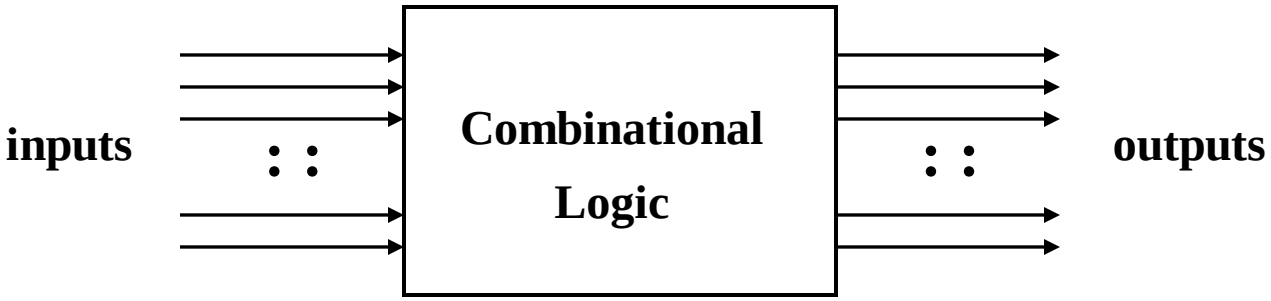
Combinational Circuits

Contents

- Introduction
- Analysis Procedure
- Half Adder
- Full Adder
- Half Subtractor
- Full Subtractor
- Decoder
- Encoder

Introduction

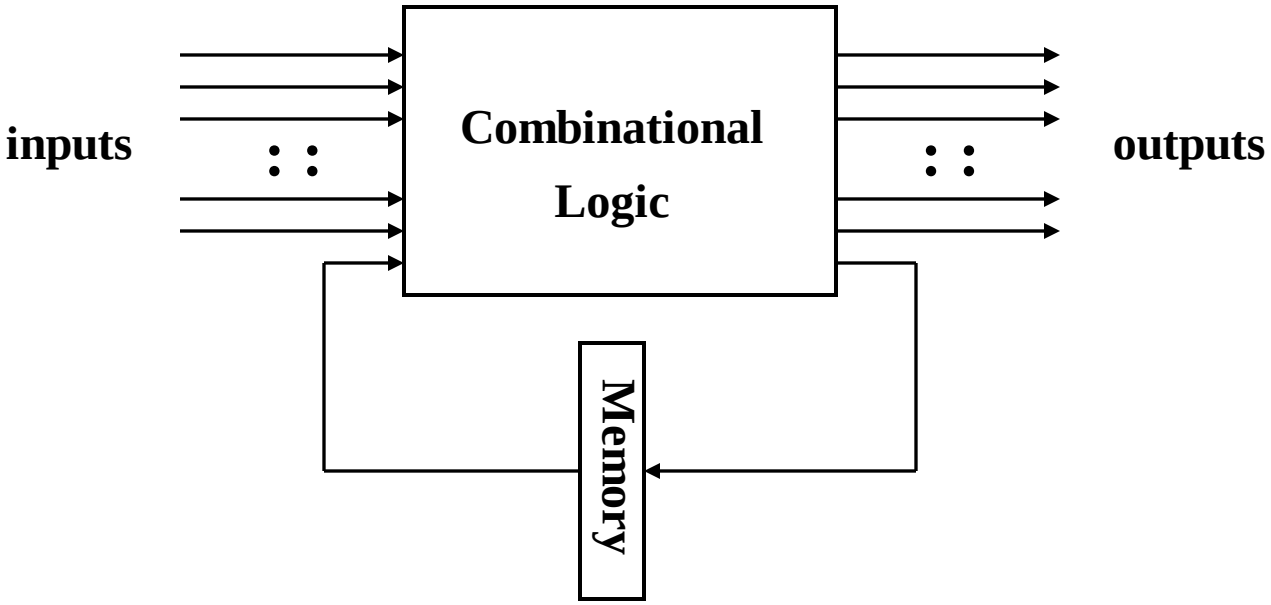
- Two classes of logic circuits:
  - combinational
  - sequential
- **Combinational Circuit**:



Each output depends entirely on the immediate (present) inputs.

Introduction

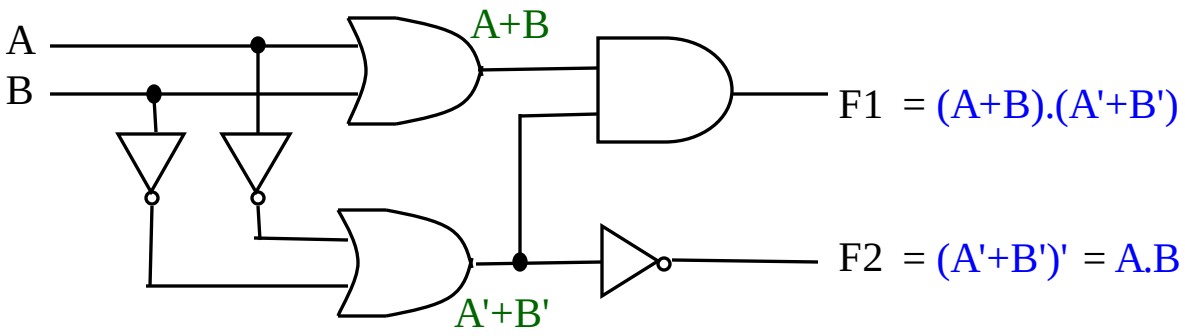
- **Sequential Circuit**: (to be covered later)



Output depends on both *present* and *past* inputs.
Memory (via feedback loop) contains past information.

Analysis Procedure

- Given a combinational circuit, can you analyze its function?



Steps:

1. Label the inputs and outputs.
2. Obtain the functions of intermediate points and the outputs.
3. Draw the truth table.
4. Deduce the functionality of the circuit ➲ half adder.

A	B	(A+B)	(A'+B')	F1	F2
0	0	0	1	0	0
0	1	1	1	1	0
1	0	1	1	1	0
1	1	1	0	0	1

Design Methods

- Different combinational circuit design methods:
  - Gate-level method (with logic gates)
  - Block-level design method (Integrated Circuit (IC) chips)
- Main objectives of circuit design:
  - (i) reduce cost
    - reduce number of gates (for SSI circuits)
    - reduce IC packages (for complex circuits)
  - (ii) increase speed
  - (iii) design simplicity (reuse blocks where possible)

Half Adder

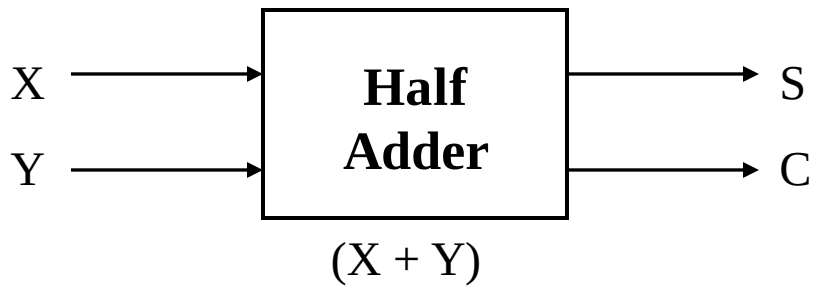
- Design procedure:

  1) State Problem

     Example: Build a Half Adder to add two bits

  2) Determine and label the inputs & outputs of circuit.

     Example: Two inputs and two outputs labelled, as follows:



X	Y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

  3) Draw truth table.

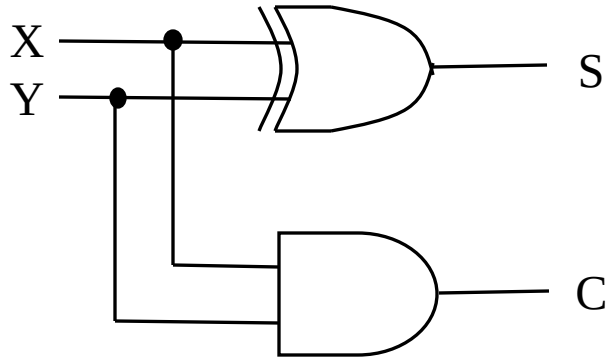
Gate-level Design: Half Adder

4) Obtain simplified Boolean function.

Example: C = X.Y

S = X'.Y + X.Y' = X⊕Y

5) Draw logic diagram.



Half Adder

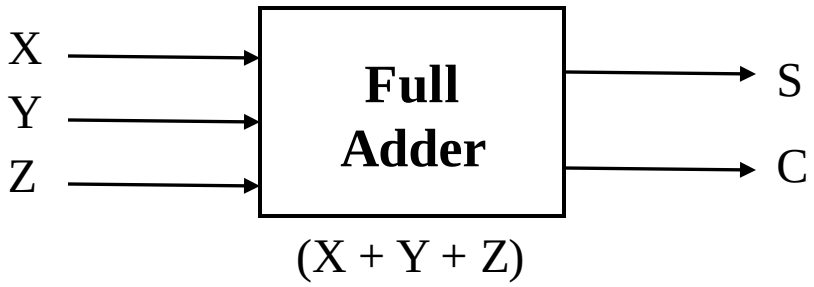
X	Y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Full Adder

- Half-adder adds up only two bits.
- To add two binary numbers, we need to add 3 bits (including the *carry*).
- Example:

	1	1	1		carry
	0	0	1	1	X
+	0	1	1	1	Y
	1	0	1	0	S

Need Full Adder (so called as it can be made from two half-adders).



Full Adder

- Truth table:

X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Note:
Z - carry in (to the current position)
C - carry out (to the next position)

C

X \ YZ	00	01	11	10
0			1	
1		1	1	1

S

X \ YZ	00	01	11	10
0		1		1
1	1		1	

Using K-map, simplified SOP form:

C = X.Y + X.Z + Y.Z

S = X'.Y'.Z + X'.Y.Z'+X.Y'.Z'+X.Y.Z

Full Adder

- Alternative formulae using algebraic manipulation:

$C = X.Y + X.Z + Y.Z$

$= X.Y + X\,(Y+Y\,').Z + (X+X')YZ$

$= X.Y + X\,YZ + XY\,'Z + X\,YZ + X'YZ$

$= X.Y\,(1+Z) + (X \oplus Y).Z$

$= X.Y + (X \oplus Y).Z$

$S = X'.Y'.Z + X'.Y.Z' + X.Y'.Z' + X.Y.Z$

$= X'.(Y'.Z + Y.Z') + X.(Y'.Z' + Y.Z)$

$= X'.(Y \oplus Z) + X.(Y \oplus Z)'$

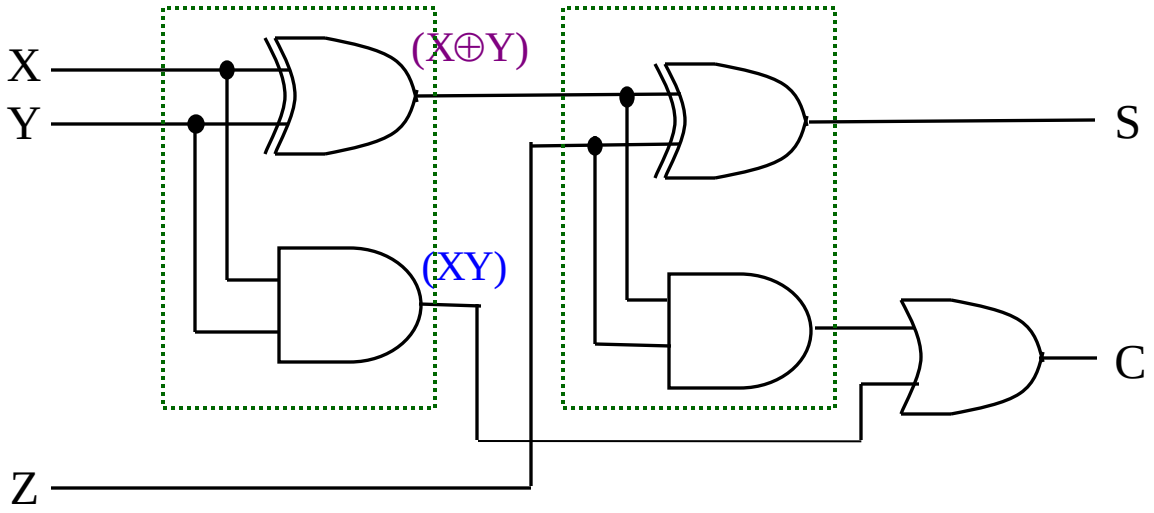
$= X \oplus (Y \oplus Z)$ or $(X \oplus Y) \oplus Z$

Gate-level Design: Full Adder

- Circuit for above formulae:

$$C = X.Y + (X \oplus Y).Z$$

$$S = (X \oplus Y) \oplus Z$$



Full Adder made from two Half-Adders (+ OR gate).

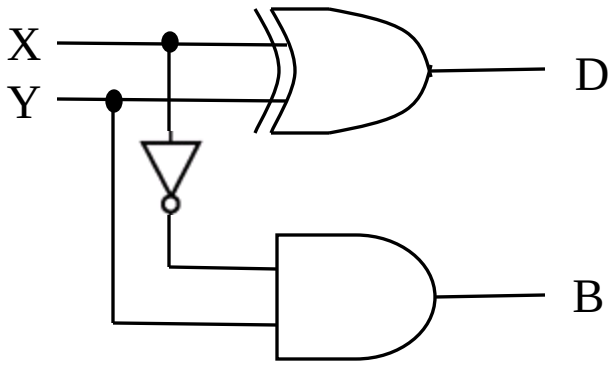
Half Subtractor

1) Obtain simplified Boolean function.

Example: B = X'.Y

D = X'.Y + X.Y' = X⊕Y

2) Draw logic diagram.



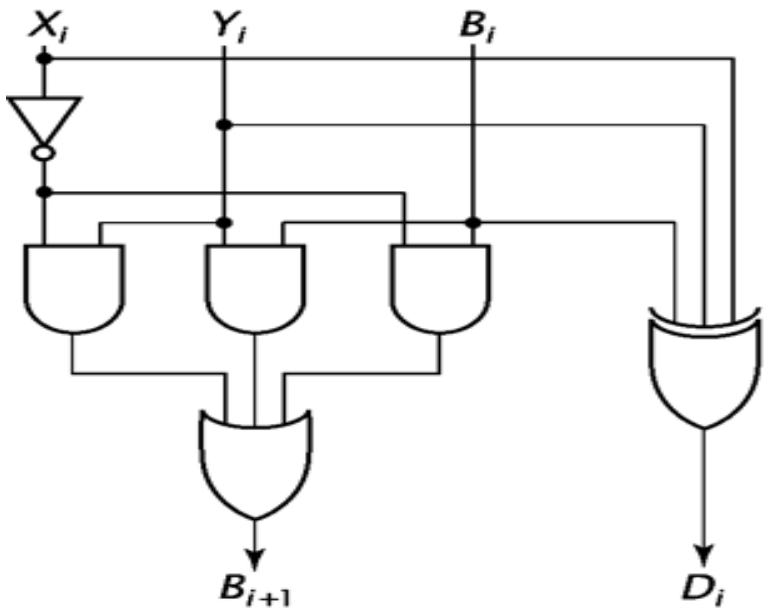
X	Y	B	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

Full Subtractor

- Full subtractor has 3 inputs, Two data inputs and One borrow input

$B=X'Y'Bi+X'YBi'+X'YBi+XYBi=X'Y+X'B_i+YB_i$

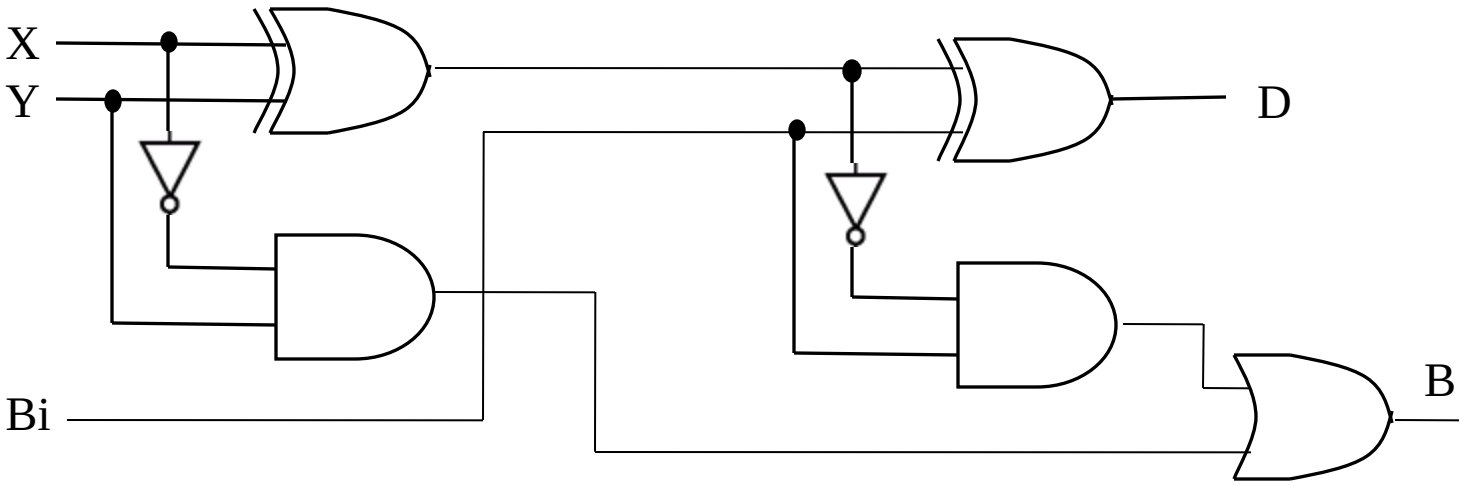
$D= X'Y'Bi+X'YBi'+XY'Bi'+XYBi= (X\oplus Y)\oplus B_i$



(a)

X	Y	Bi	B	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Full Subtractor using Two Half Subtractor

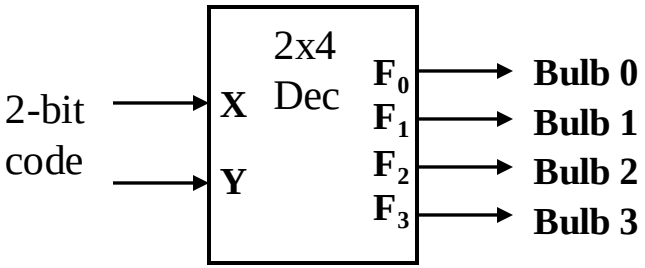


Decoders

- Convert binary information from n input lines to (max. of) 2^n output lines.
- Known as n-to-m-line decoder, or simply n:m or $n \times m$ decoder ($m \leq 2^n$).
- May be used to generate 2^n (or fewer) minterms of n input variables.

Decoder (2x4)

- Example: if codes 00, 01, 10, 11 are used to identify four light bulbs, we may use a 2-bit decoder:



This is a 2×4 decoder which selects an output line based on the 2-bit code supplied.

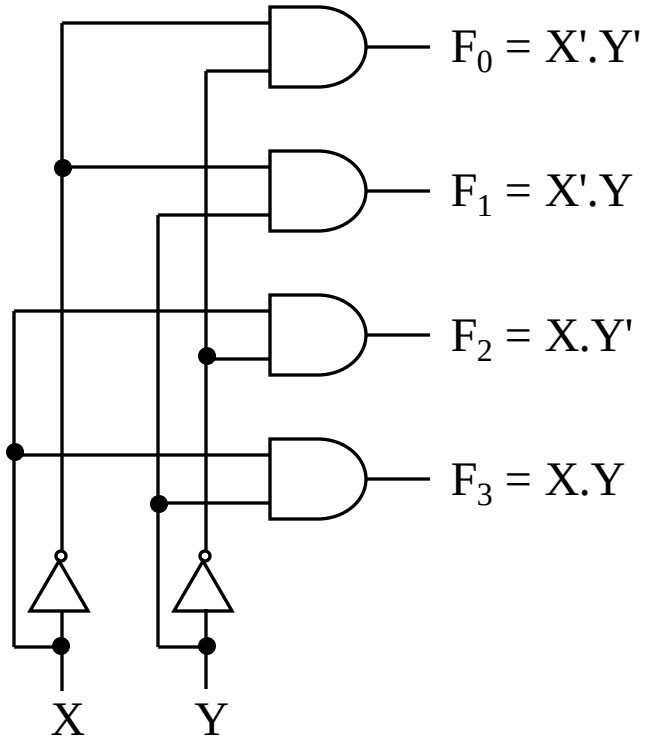
Truth table:

X	Y	F_0	F_1	F_2	F_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Decoders (2x4)

X	Y	F_0	F_1	F_2	F_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

- $F_0 = X'.Y'$,
- $F_1 = X'.Y$,
- $F_2 = X.Y'$
- $F_3 = X.Y$

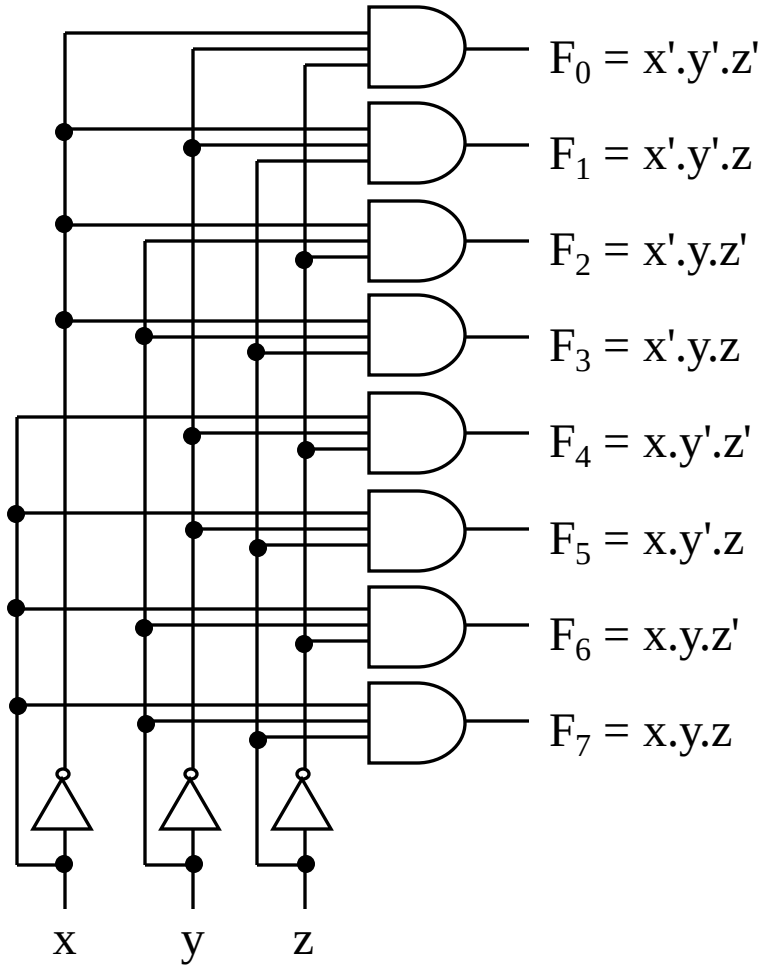


Decoder (3x8)

- Design a 3×8 decoder.

x	y	z	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Application? Binary-to-octal conversion.



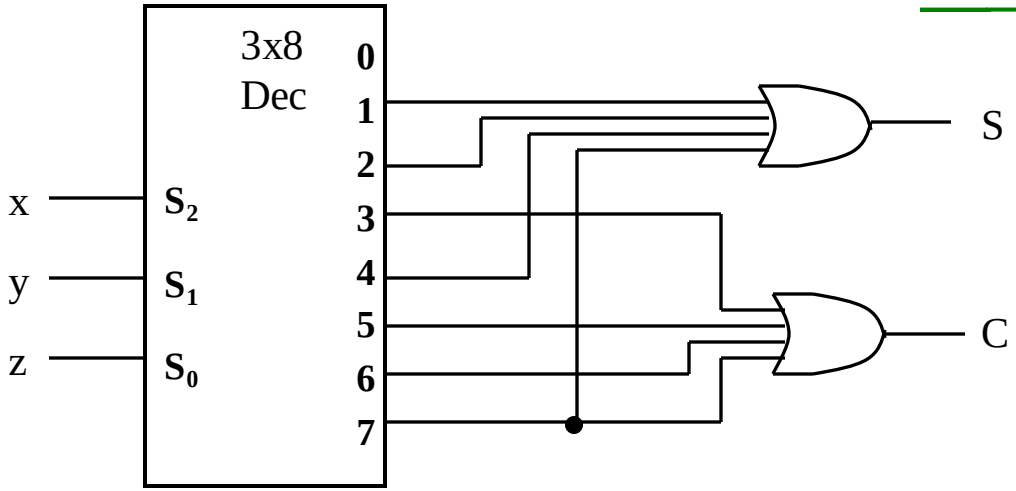
Decoders: Implementing Functions

- Example: Full adder

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

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

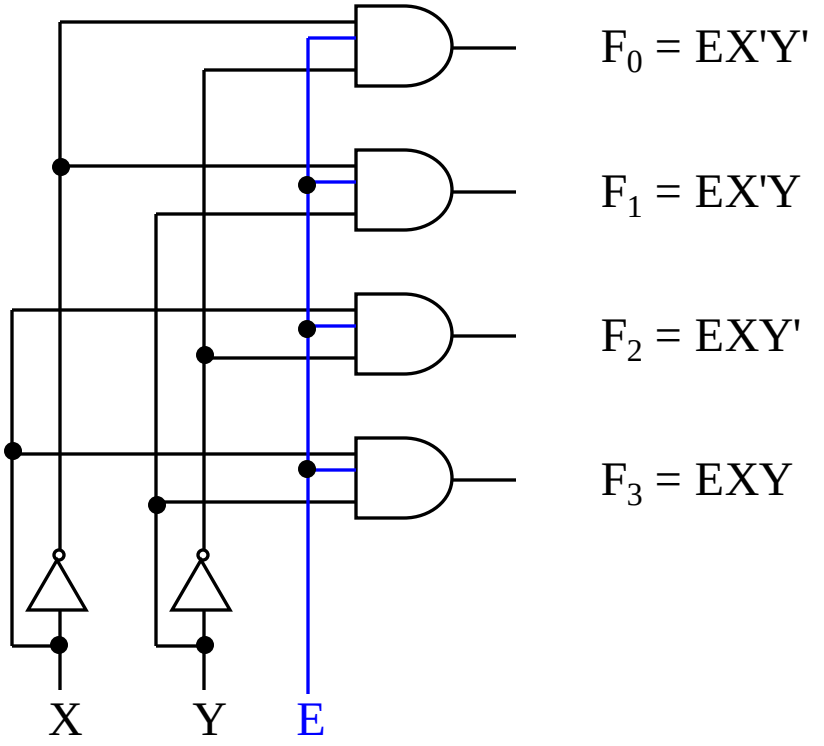
x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



Decoders with Enable

- Decoders often come with an enable signal, so that the device is only activated when the enable, E=1.
- Truth table:

E	X	Y	F_0	F_1	F_2	F_3
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1
0	X	X	0	0	0	0



Decoders with Enable

- In the previous slide, the decoder has a one-enable signal, that is, the decoder is enabled with E=1.
- In most MSI decoders, enable signal is zero-enable, usually denoted by E' (or E). The decoder is enabled when the signal is zero.

E	X	Y	F_0	F_1	F_2	F_3
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1
0	X	X	0	0	0	0

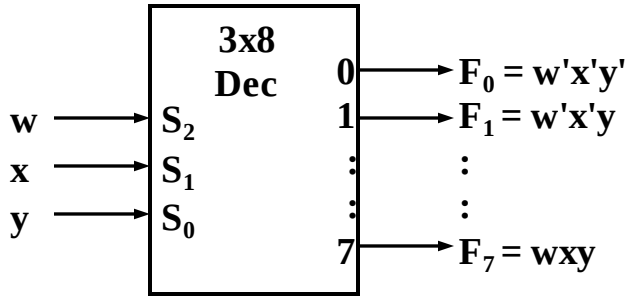
Decoder with 1-enable

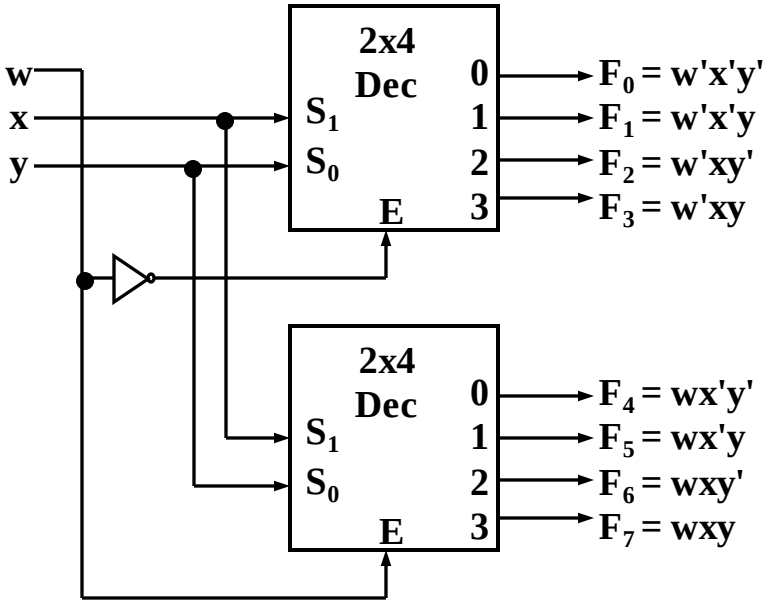
E'	X	Y	F_0	F_1	F_2	F_3
0	0	0	1	0	0	0
0	0	1	0	1	0	0
0	1	0	0	0	1	0
0	1	1	0	0	0	1
1	X	X	0	0	0	0

Decoder with 0-enable

Design (3x8) Decoder using (2x4) Decoders

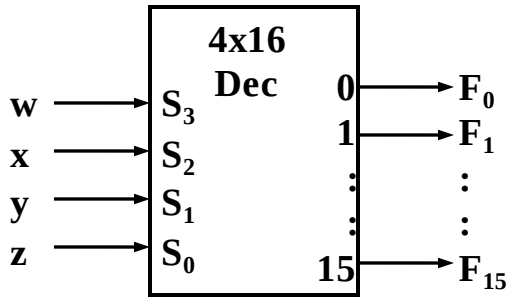
- Larger decoders can be constructed from smaller ones.
- For example, a 3-to-8 decoder can be constructed from two 2-to-4 decoders (with one-enable), as follows:

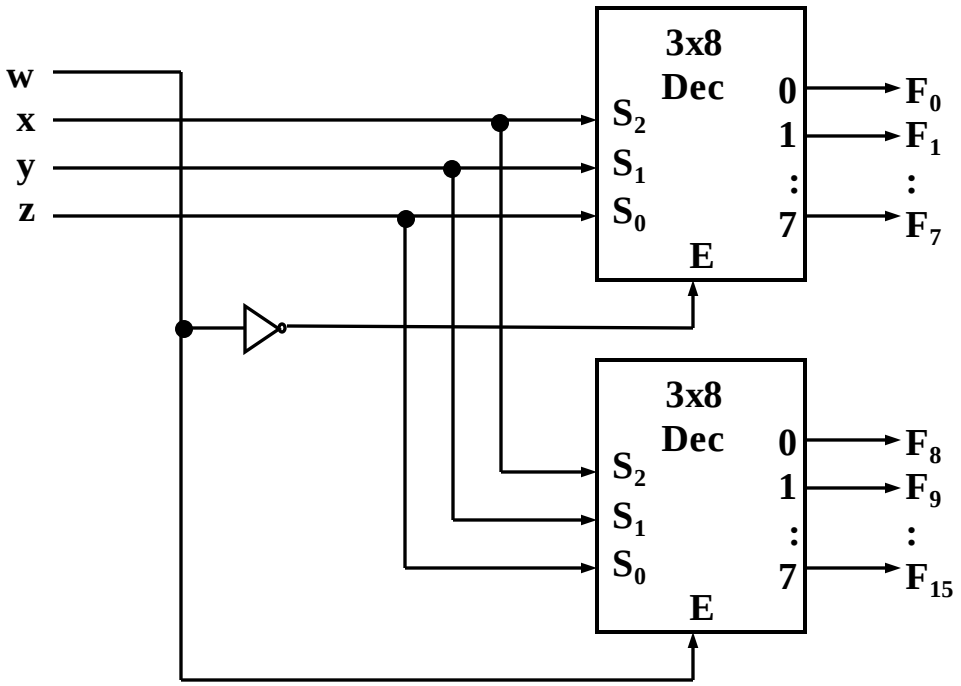


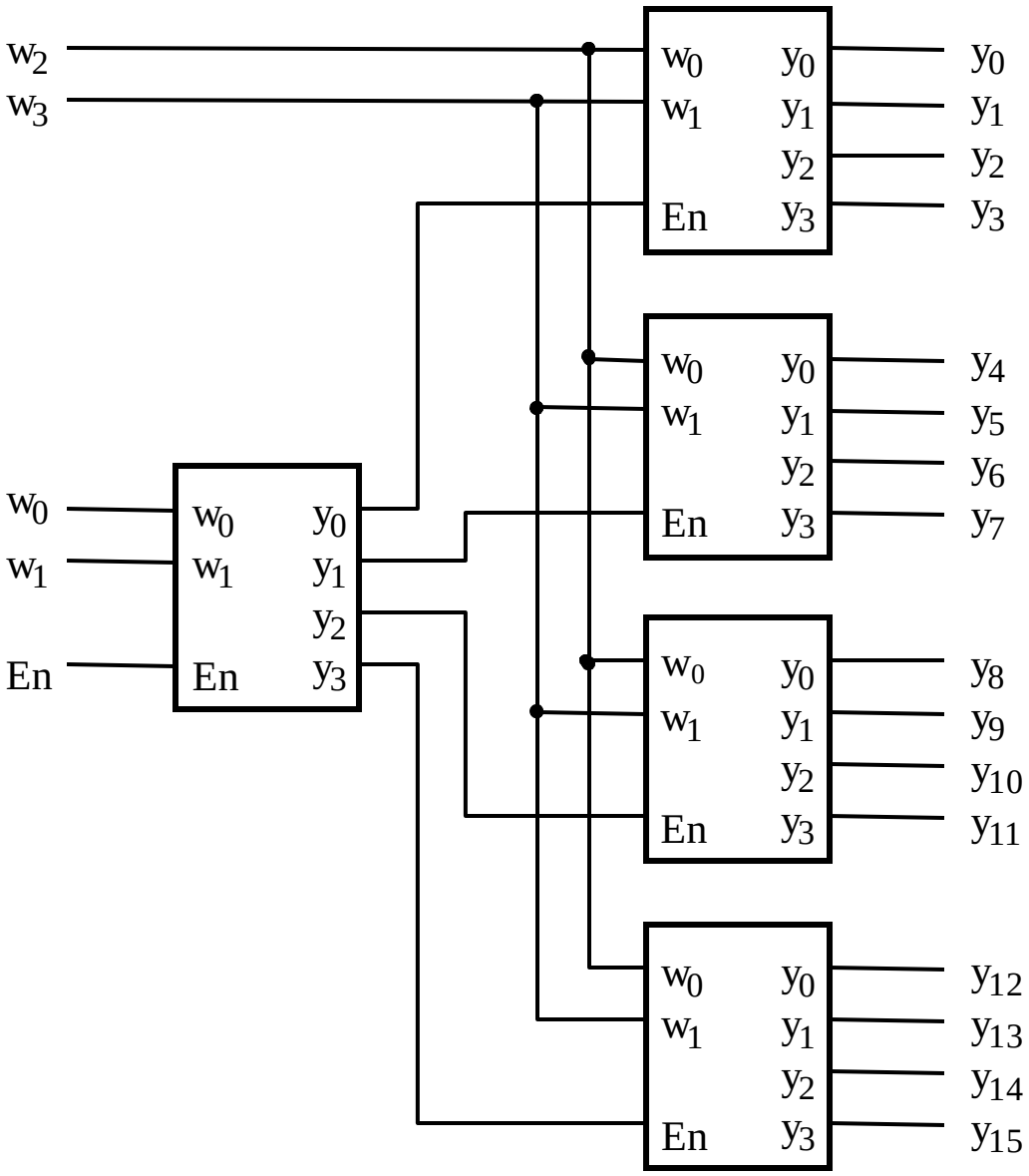


Design (4x16) Decoder using (3x8) Decoders

- Construct a 4x16 decoder from two 3x8 decoders with 1-enable.







A 4-to-16 decoder built using a decoder tree.

Encoder

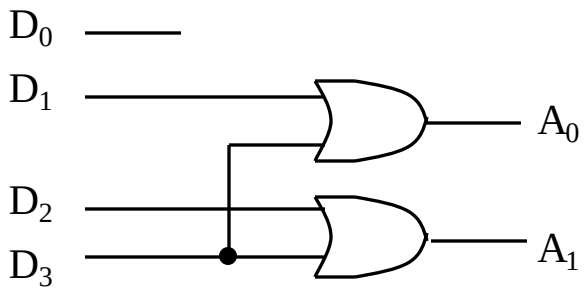
- Encoding is the converse of decoding.
- Given a set of input lines, where one has been selected, provide a code corresponding to that line.
- Contains 2^n (or fewer) input lines and n output lines.
- Implemented with OR gates.

(4x2) Encoder

$$A_0 = D_1 + D_3$$
$$A_1 = D_2 + D_3$$

	A_1	A_0
D_0	0	0
D_1	0	1
D_2	1	0
D_3	1	1

which correspond to circuit:



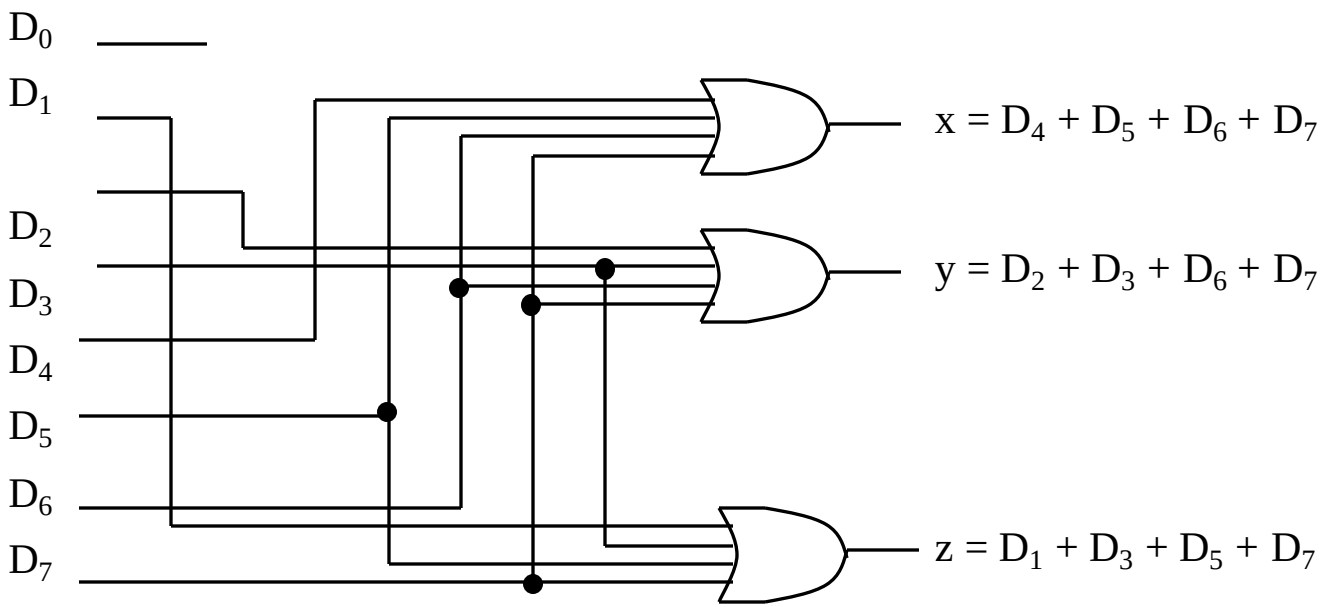
Simple 4-to-2 encoder

Octal-to-Binary Encoder

	x	y	z
D_0	0	0	0
D_1	0	0	1
D_2	0	1	0
D_3	0	1	1
D_4	1	0	0
D_5	1	0	1
D_6	1	1	0
D_7	1	1	1

Encoder

- Example: Octal-to-binary encoder.



References

1. A. Anand Kumar, “Fundamentals of Digital Circuits”, PHI, Fourth Edition.

2. S. Salivahanan and S. Arivazhagan, “Digital circuits and design”, Vikas Publishing House PVT Limited, Fifth edition.