Digital Design

Ms. Umarah Qaseem

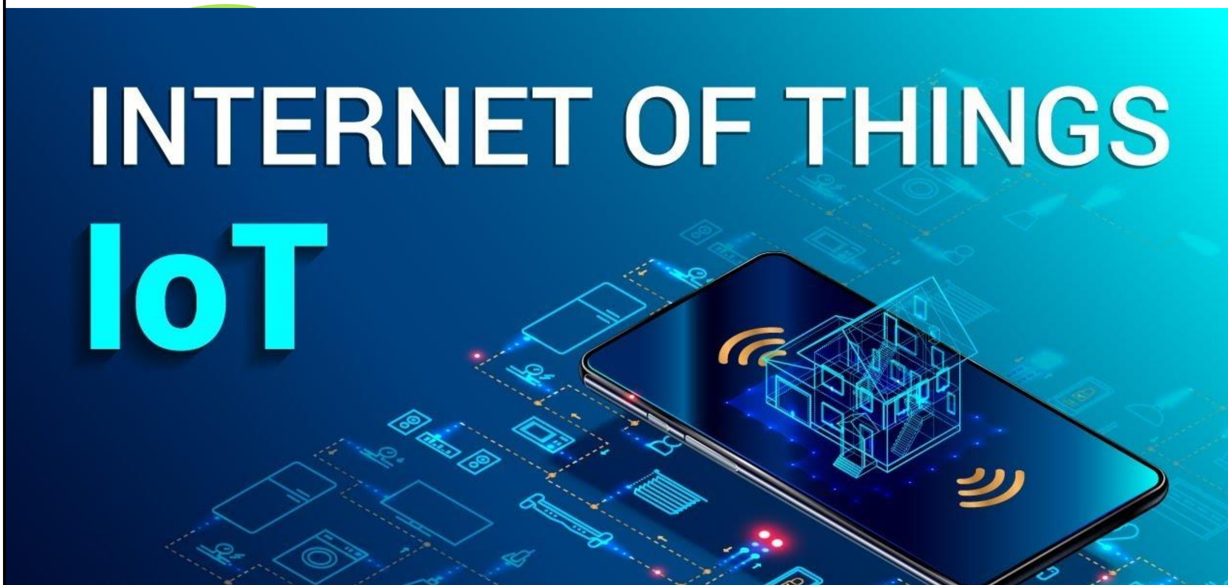
1

11

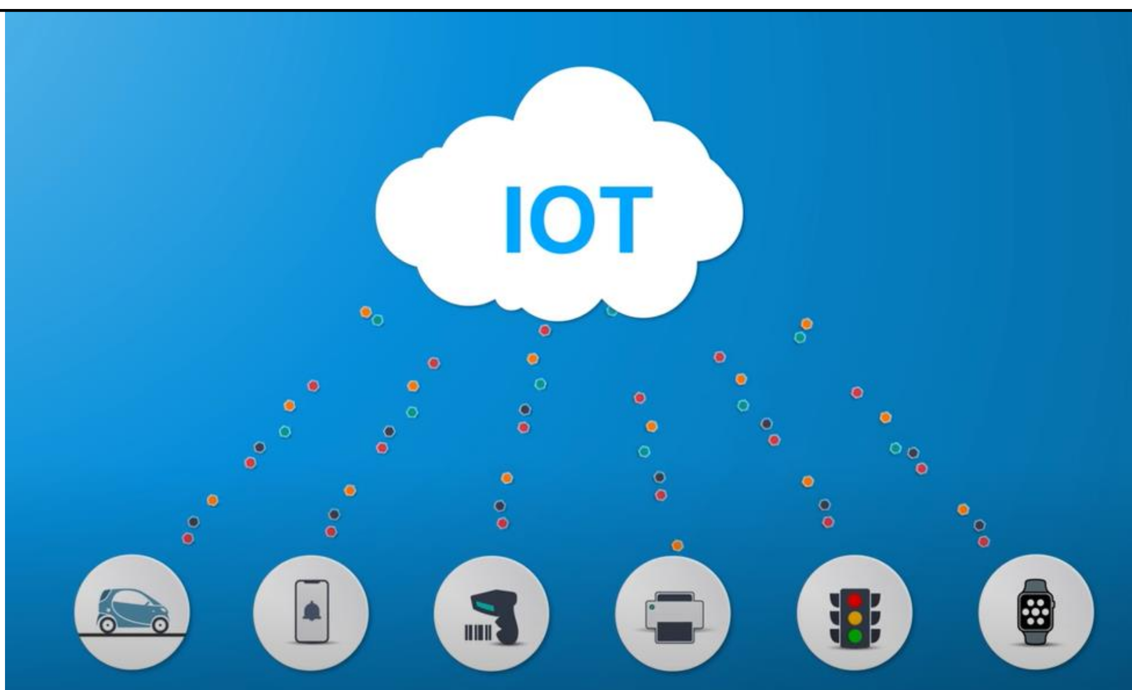
Chapter 4

Combinational Logic

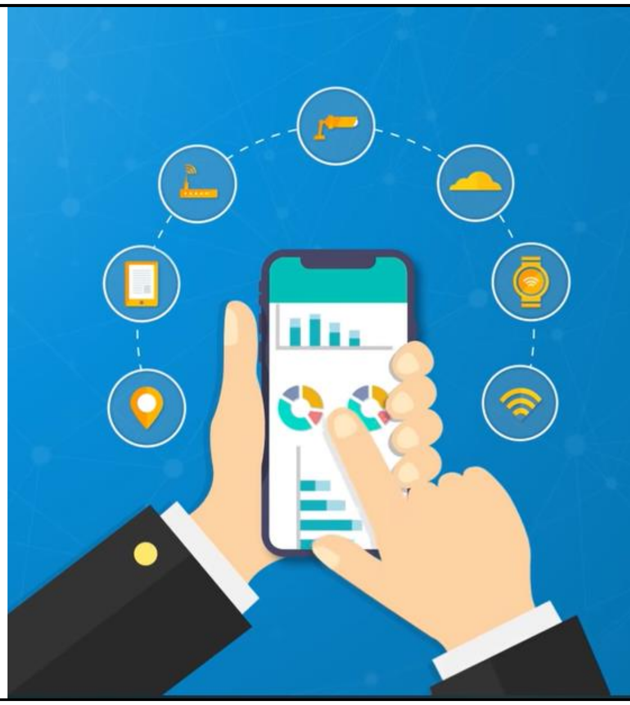
2



6



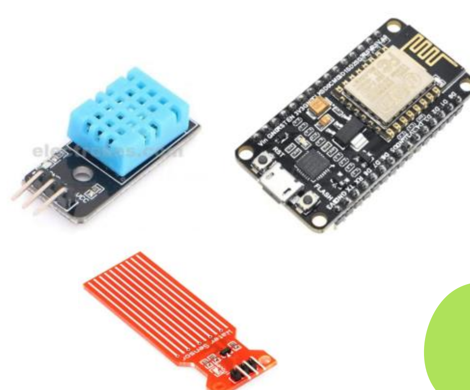
7



8

Material to be used in IOT Lab

- ESP8266 V3 (550/-)
- ESP8266 V2 (650/-)
- DHT11 (240/-)
- SR04 (170/-)
- Water Level Sensor (70/-)



9

Binary Multiplier

- Multiplication of binary numbers is done in the same way as decimal numbers
- Multiplicand *B* is multiplied by the multiplier *A* starting from the LSB.
- Successive partial products are shifted one position from the left and the final product is obtained from the sum of partial products.

$$
\begin{array}{cccc}
 & & B_1 & B_0 \\
 & & A_1 & A_0 \\
\cline{3-4}
 & & A_0B_1 & A_0B_0 \\
 & A_1B_1 & A_1B_0 & \\
\hline
C_3 & C_2 & C_1 & C_0
\end{array}
$$

10

10

Binary Multiplier

2-Bit by 2-Bit Binary Multiplier

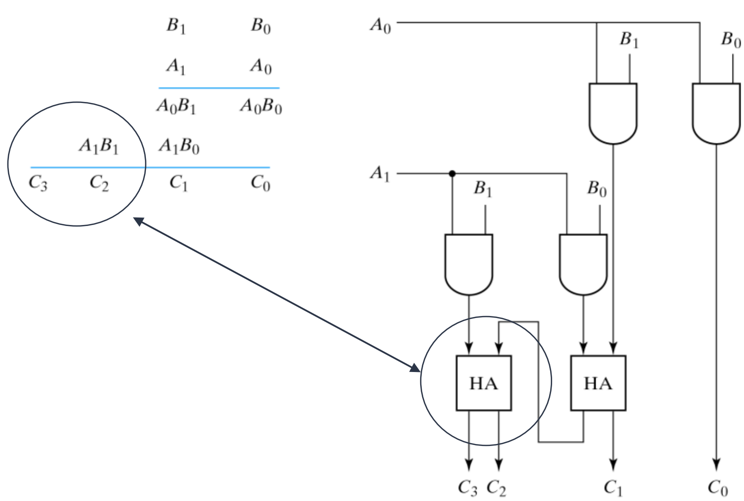


Fig. 4-15 2-Bit by 2-Bit Binary Multiplier

11

11

4-Bit by 3-Bit Binary Multiplier

Binary Multiplier

B3 B2 B1 B0
A2 A1 A0

A0B3 A0B2 A0B1 A0B0

A1B3 A1B2 A1B1 A1B0

A2B3 A2B2 A2B1 A2B0

J x *K*

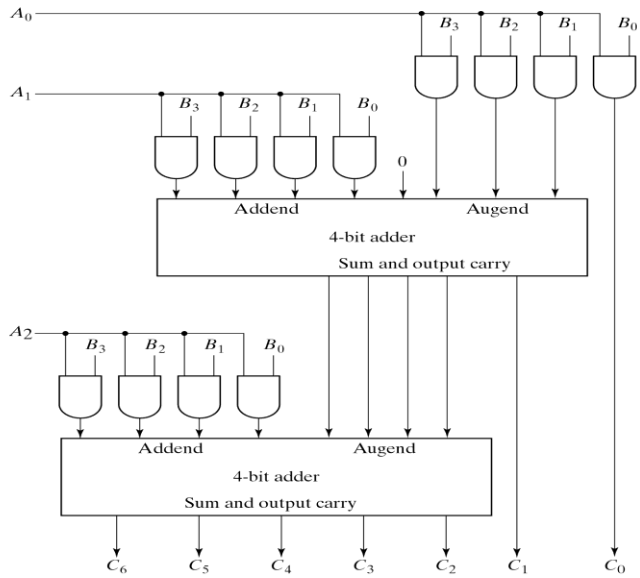


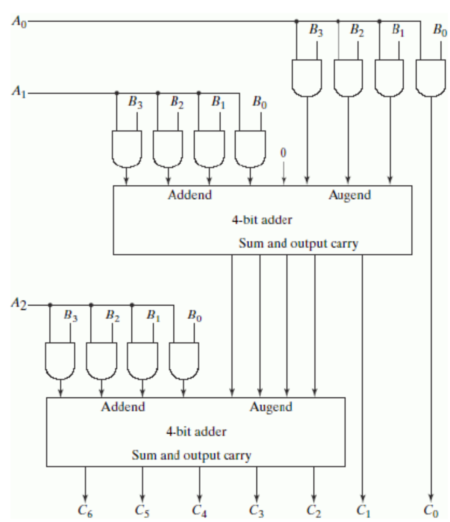
Fig. 4-16 4-Bit by 3-Bit Binary Multiplier

12

12

Multiplier – Last Class Queries

- 4-bit x 3-bit multiplier
  - J=3, K=4
- How many AND gates
  - J x K gates
- How many adders
  - (J-1) K-bit adders
- How many output bits
  - J + K bits



13

Practice Exercises

- Design a 3-bit by 4-bit multiplier
- Design a 4-bit by 4-bit multiplier

14

14

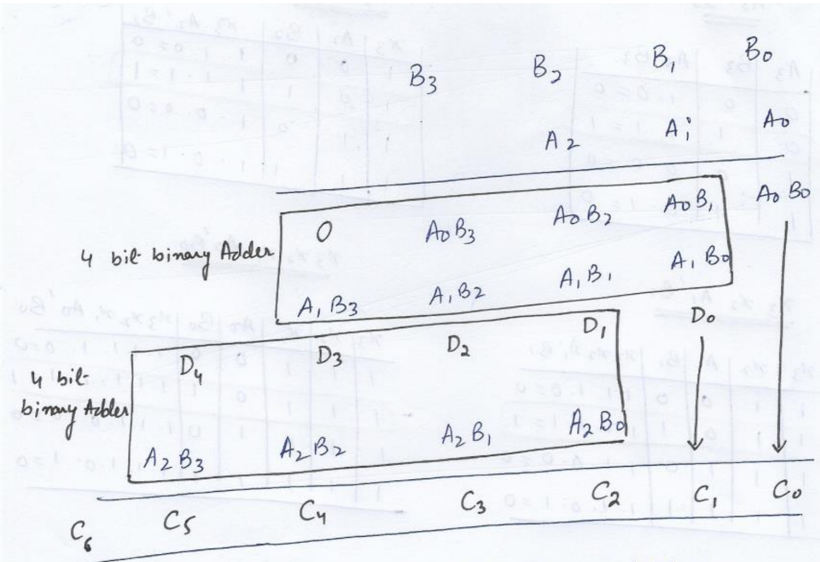
MULTIPLIER

$K = 4$
$J = 3$
And Gates $= J \times K$
$= 4 \times 3$
$= 12$

Output bits $= J + K$
$= 4 + 3$
$= 7$

			B_3	B_2	B_1	B_0
				A_2	A_1	A_0
			A_0B_3	A_0B_2	A_0B_1	A_0B_0
		A_1B_3	A_1B_2	A_1B_1	A_1B_0	
	A_2B_3	A_2B_2	A_2B_1	A_2B_0		
C_6	C_5	C_4	C_3	C_2	C_1	C_0

15



Number of Adder = (J-1) k-bit Adders
= (3-1) 4-bit Adders
= (2) 4 bit Adders.

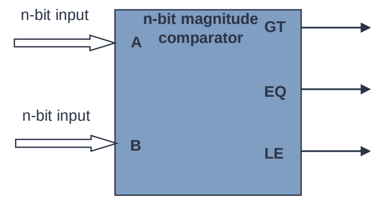
16

Magnitude Comparator

17

Magnitude Comparator

- **Definition:** A magnitude comparator is a combinational circuit that compares two numbers A & B to determine whether:
  - A > B, or
  - A = B, or
  - A < B
- **Inputs**
  - First n-bit number A
  - Second n-bit number B
- **Outputs**
  - 3 output signals (GT, EQ, LT), where:
    - GT = 1 IFF A > B
    - EQ = 1 IFF A = B
    - LT = 1 IFF A < B



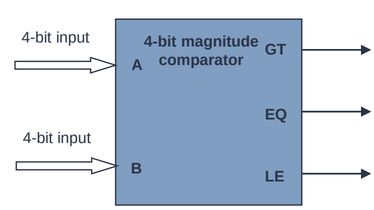
Note: Exactly One of these 3 outputs equals 1, while the other 2 outputs are 0`s

18

18

Magnitude Comparator (4 bit)

- **Problem:** Design a magnitude comparator that compares two 4-bit numbers A and B and determines whether:
  - A > B, or
  - A = B, or
  - A < B

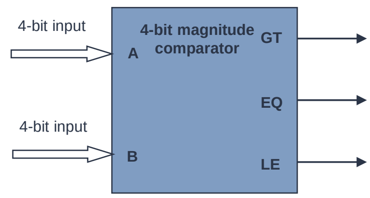


19

19

Magnitude Comparator (4 bit)

- Inputs: 8-bits (A ⇒ 4-bits , B ⇒ 4-bits)
  - A and B are two 4-bit numbers
- Let A = $A_3A_2A_1A_0$, and
- Let B = $B_3B_2B_1B_0$
- Inputs have 2^8 (256) possible combinations (size of truth table and K-map?)
- Not easy to design using conventional techniques



The circuit possesses certain amount of regularity ⇒ *can be designed algorithmically*.

20

20

Designing EQ

Magnitude Comparator (4 bit)

$$\boxed{\begin{matrix} A \\ B \end{matrix}} \begin{matrix} = \\ = \end{matrix} \boxed{\begin{matrix} A_3 \\ B_3 \end{matrix}}\boxed{\begin{matrix} A_2 \\ B_2 \end{matrix}}\boxed{\begin{matrix} A_1 \\ B_1 \end{matrix}}\boxed{\begin{matrix} A_0 \\ B_0 \end{matrix}}$$

Can you think of an operator which is 1 when both inputs are 1 or both inputs are 0

XNOR

21

21

Magnitude Comparator

- How can we determine that two numbers are equal?
  - Equal if every digit is equal
    - $A_3A_2A_1A_0 = B_3B_2B_1B_0$ if and only if
      $A_3 = B_3$ and $A_2 = B_2$ and $A_1 = B_1$ and $A_0=B_0$
- Which gate?
  - XNOR (Equivalence)

22

COMPARATOR

Case 1 : $A = B$

$$
\begin{array}{cccc}
A_3 & A_2 & A_1 & A_0 \\
B_3 & B_2 & B_1 & B_0 \\
\hline
x_3 & x_2 & x_1 & x_0
\end{array}
$$

A	B	XOR	XNOR
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

$A=B$ If $x_3 \cdot x_2 \cdot x_1 \cdot x_0 = 1$

where $x_3 = \overline{A_3 \oplus B_3} = \overline{A_3 B_3' + A_3' B_3}$

$x_2 = \overline{A_2 \oplus B_2} = \overline{A_2 B_2' + A_2' B_2}$

$x_1 = \overline{A_1 \oplus B_1} = \overline{A_1 B_1' + A_1' B_1}$

$x_0 = \overline{A_0 \oplus B_0} = \overline{A_0 B_0' + A_0' B_0}$

23

Magnitude Comparator (4 bit)

Designing EQ

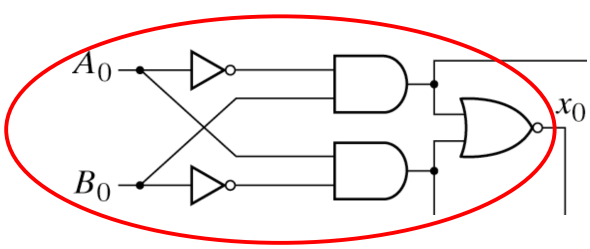
- Define X_i = A_i XNOR B_i ; $A_i B_i + A_i' B_i'$
  - ➔ $X_i = 1$ IFF $A_i = B_i$ ∀ i =0, 1, 2 and 3
  - ➔ $X_i = 0$ IFF $A_i \neq B_i$
- Therefore, the condition for A = B or EQ=1 IFF
  - $A_3 = B_3 \rightarrow (X_3 = 1)$, and
  - $A_2 = B_2 \rightarrow (X_2 = 1)$, and
  - $A_1 = B_1 \rightarrow (X_1 = 1)$, and
  - $A_0 = B_0 \rightarrow (X_0 = 1)$.
- Thus, EQ=1 IFF $X_3 X_2 X_1 X_0 = 1$. In other words, $EQ = X_3 X_2 X_1 X_0$

$$A = A_3 A_2 A_1 A_0$$
$$B = B_3 B_2 B_1 B_0$$

24

24

Magnitude Comparator (4 bit)

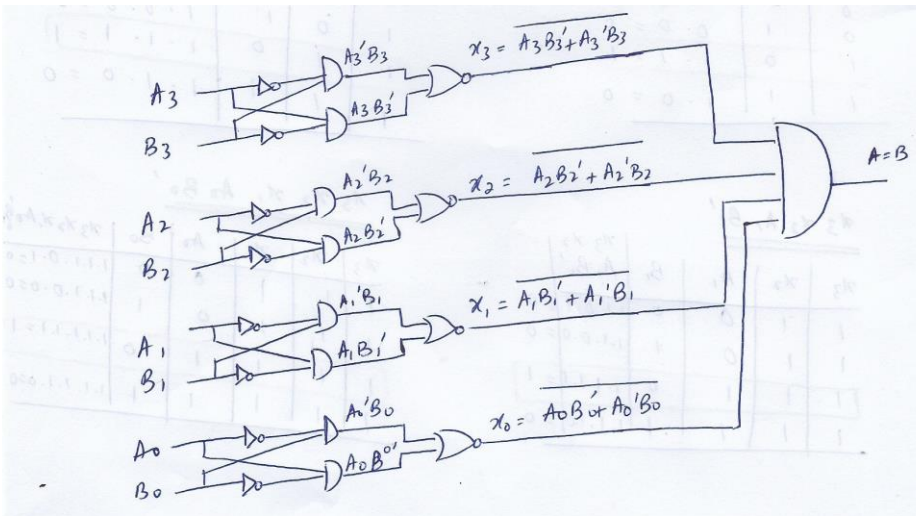


$$x_0 = (A_0' B_0 + A_0 B_0')' = A_0 B_0 + A_0' B_0'$$

XNOR

25

25



26

Magnitude Comparator (4 bit)

Designing LT & GT

- GT = 1 if A > B:
  - If $A_3 > B_3 \rightarrow A_3 = 1$ and $B_3 = 0$
  - If $A_3 = B_3$ and $A_2 > B_2$
  - If $A_3 = B_3$ and $A_2 = B_2$ and $A_1 > A_1$
  - If $A_3 = B_3$ and $A_2 = B_2$ and $A_1 = B_1$ and $A_0 > B_0$

$$GT = (A > B) = A_3B_3^{'} + x_3A_2B_2^{'} + x_3x_2A_1B_1^{'} + x_3x_2x_1A_0B_0^{'}$$
$$LT = (A < B) = A_3^{'}B_3 + x_3A_2^{'}B_2 + x_3x_2A_1^{'}B_1 + x_3x_2x_1A_0^{'}B_0$$

Compare: $A = 1010 \text{ and } B = 0101 \rightarrow (A > B) = 1$

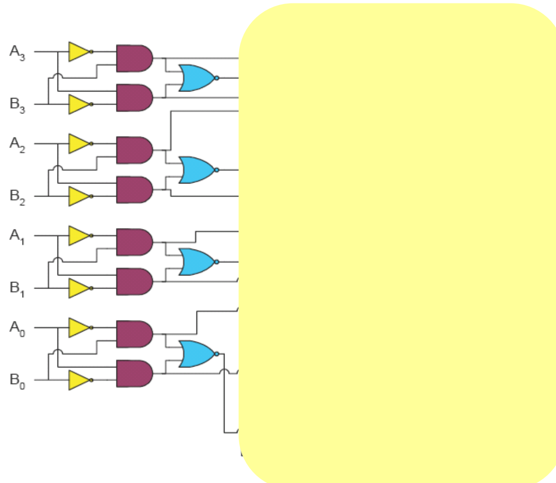
$A = 0101 \text{ and } B = 1010 \rightarrow (A < B) = 1$

27

27

Magnitude Comparator (4 bit)

- $EQ = X_3 X_2 X_1 X_0$
- $GT = A_3B_3'$
  $+ X_3A_2B_2'$
  $+ X_3X_2A_1B_1'$
  $+ X_3X_2X_1A_0B_0'$
- $LT = B_3A_3'$
  $+ X_3B_2A_2'$
  $+ X_3X_2B_1A_1'$
  $+ X_3X_2X_1B_0A_0'$

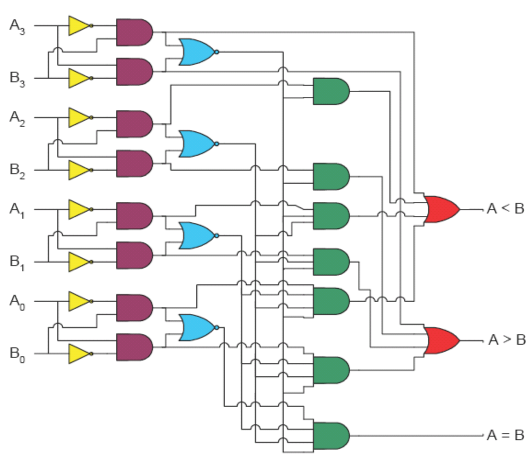


28

28

Magnitude Comparator (4 bit)

- $EQ = X_3 X_2 X_1 X_0$
- $GT = A_3B_3'$
  $+ X_3A_2B_2'$
  $+ X_3X_2A_1B_1'$
  $+ X_3X_2X_1A_0B_0'$
- $LT = B_3A_3'$
  $+ X_3B_2A_2'$
  $+ X_3X_2B_1A_1'$
  $+ X_3X_2X_1B_0A_0'$



29

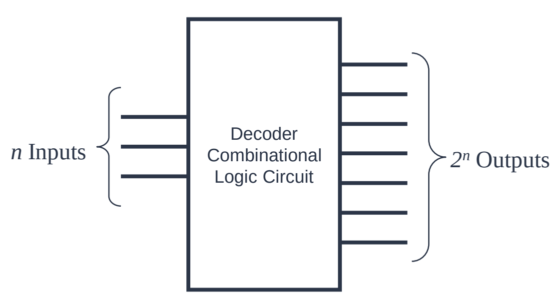
29

Decoders

30

Decoders

A decoder is a combinational circuit that converts binary information from *n* input lines to <= 2^n unique output lines

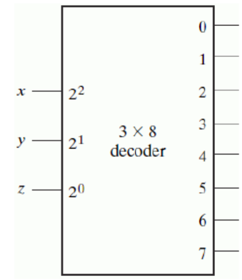


31

31

Decoders

- A decoder selects one output based on binary input
- Converts *n-bit code into* 2^n *outputs, only one being active for* any combination of inputs
- Selects output *x if input is binary representation of x*
- Also called *n*-to-*m* line decoders for example:
  - 2-to-4 line decoder
  - 3-to-8 line decoder



32

32

Decoders

- Selection of any kind
  - *Microprocessor memory system: selecting different banks of memory.*
  - *Microprocessor I/O: Selecting different devices.*
- Implementing arbitrary logic functions

33

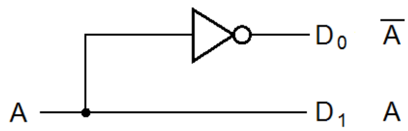
33

Decoder Examples

- 1-to-2-line Decoder

A	D_0	D_1
0	1	0
1	0	1

(a)



(b)

34

34

Decoder Examples

- 2-to-4-Line Decoder

A_1	A_0	D_0	D_1	D_2	D
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

35

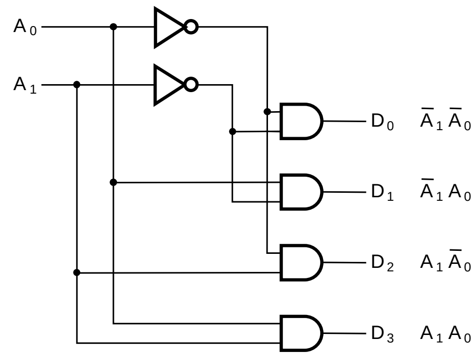
35

Decoder Examples

- 2-to-4-Line Decoder

A_1	A_0	D_0	D_1	D_2	D_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

(a)



(b)

36

36

Decoder Examples

- 3-to-8-Line Decoder

37

37

Decoder Examples

- 3-to-8-Line Decoder

Binary Inputs			Outputs							
			D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_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

•Three inputs are decoded into eight outputs, each representing one of the minterms of the three input variable

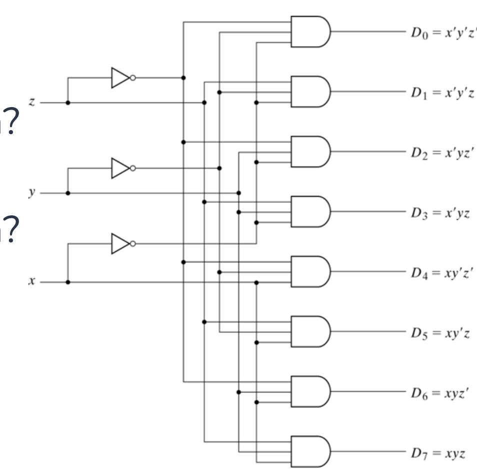
•If the input corresponds to minterm m_i then the decoder ouput$_i$ will be the corresponding single output

38

38

3-to-8-Line Decoder

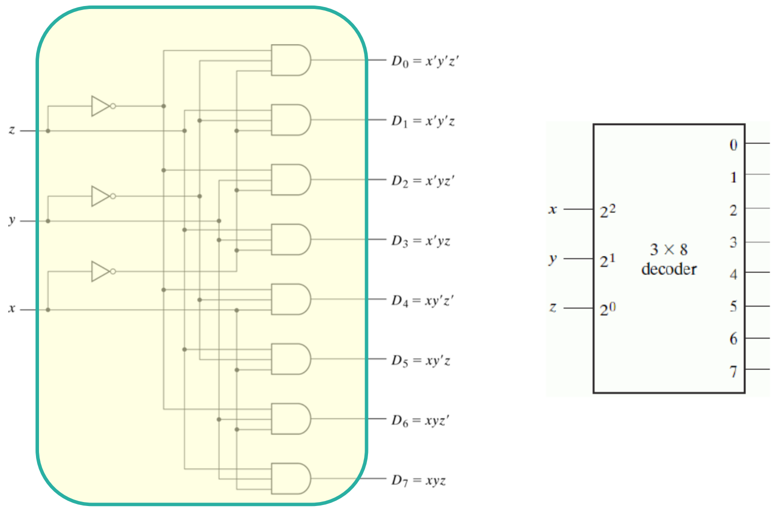
- When is output 0 chosen?
  - If x' y' z'
- When is output 1 chosen?
  - If x' y' z
- … and so on …



39

39

3-to-8-Line Decoder



40

40

Demultiplexer

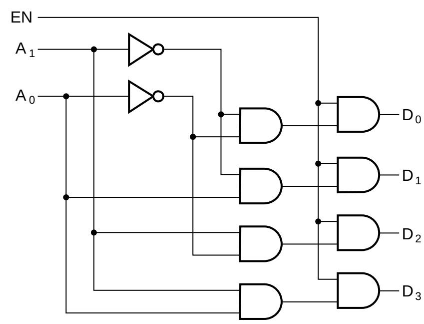
41

Decoder with Enable Input

- The decoder is enabled when *EN* = 1. The output whose value = 1 represents the minterm is selected by inputs *A* and *B.*
- The decoder is disabled when *EN* = 0 → $D_0 \ldots D_3 = 0$

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

(a)



(b)

42

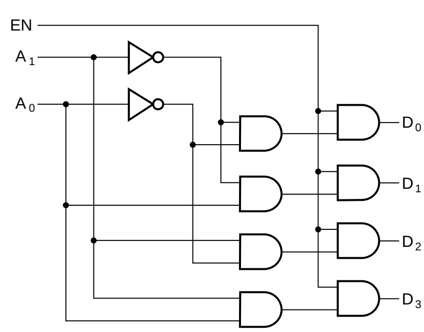
42

Decoder with Enable Input

- **A Decoder with enable input is called a demultiplexer.**
- Demultiplexer receives information from a single line and directs it to the output lines.

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

(a)

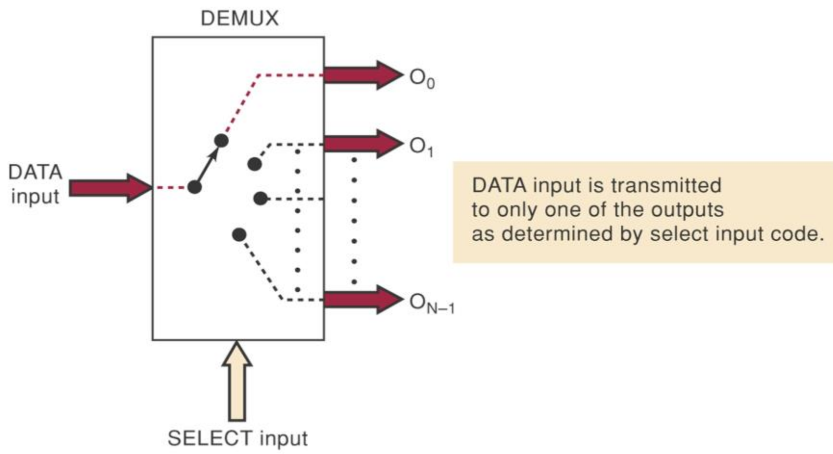


(b)

43

43

Demultiplexer



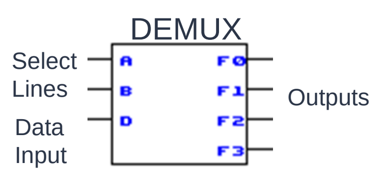
44

44

Demultiplexer

- A demultiplexer "connects" a data input to one and only one output. The selected output is specified by a decoding of the control inputs.

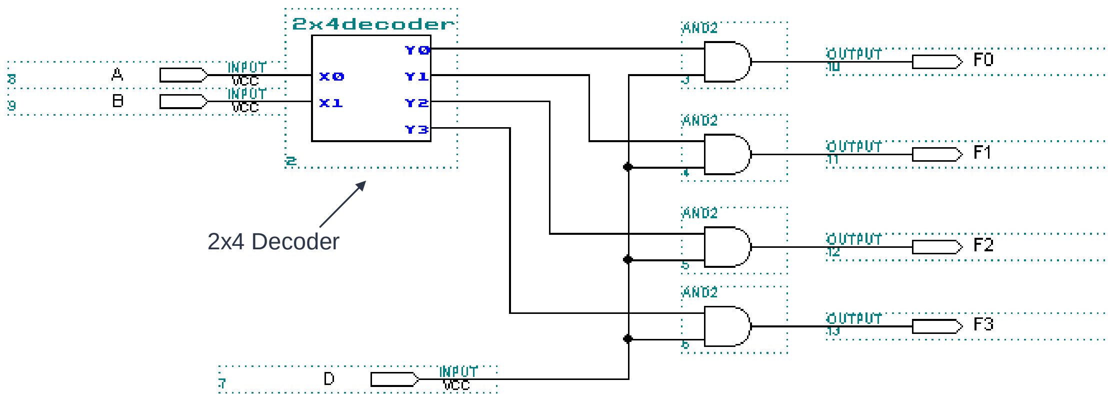
D	A	B	F3	F2	F1	F0
D	0	0	0	0	0	D
D	0	1	0	0	D	0
D	1	0	0	D	0	0
D	1	1	D	0	0	0



45

45

1 to 4 Demux



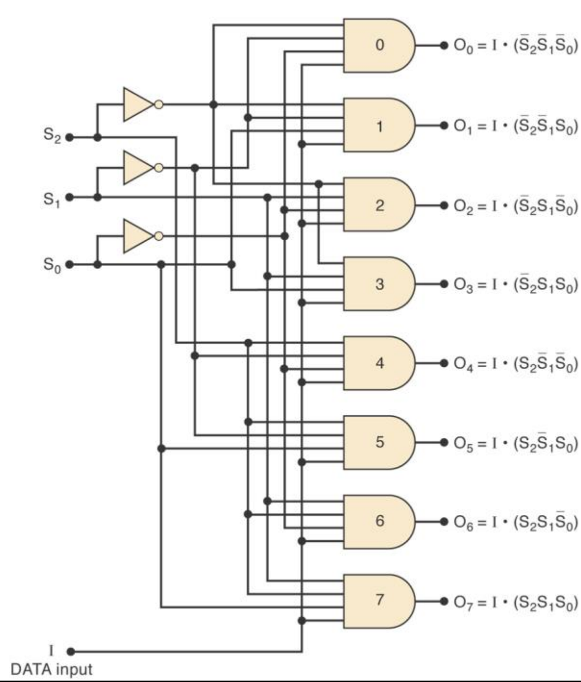
Only 1 AND gate will be "ON"

Only one F will be active

46

46

1-to-8 line DEMUX



SELECT code S_2	S_1	S_0	OUTPUTS O_7	O_6	O_5	O_4	O_3	O_2	O_1	O_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

47

Decoder with Enable

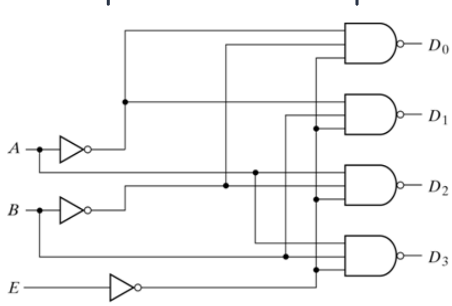
- EN is called a Control Signal
- Control Signals can be
  - Active High Signal
    - EN = 1 – Turns "ON" Decoder
  - Active Low Signal
    - EN=0 – Turns "ON" Decoder

48

48

Decoder with Enable Input - NAND

- 2-to-4 line decoder with *Enable* - NAND implementation
- Circuit generates output only if *Enable* is selected (E=0)
- If disabled (E=1), no output line is picked
- Truth table for NAND decoder
- Complemented outputs and *Enable*



(a) Logic diagram

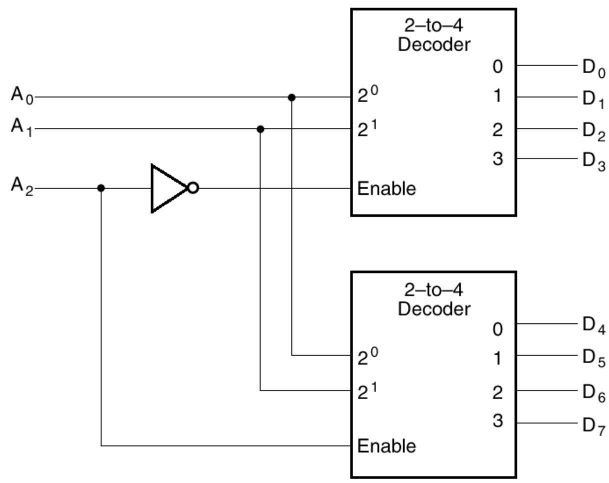
E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

(b) Truth table

49

49

Using Enable Input for Expansion

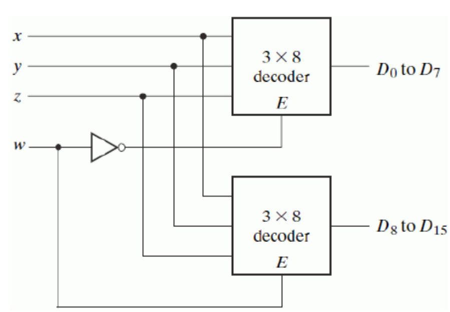


50

50

Advanced Decoder

- Enable bit allows construction of large decoders using smaller ones
- Example: Construct a 4-to-16 decoder only using 3-to-8 decoders

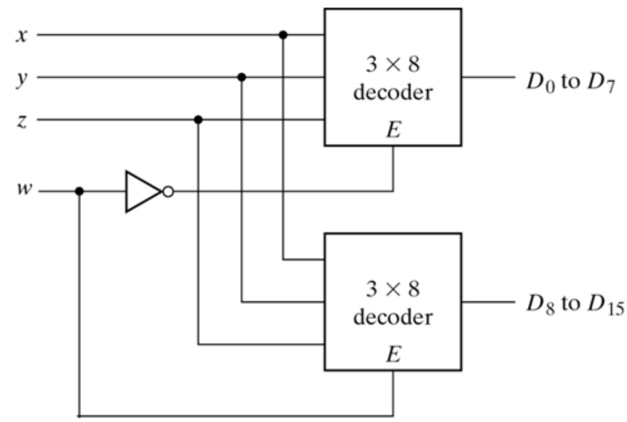


51

A 4x16 Decoder

When $w = 1$, the top decoder is disabled and the bottom is enabled.

Bottom decoder generates 8 minterms 1000 to 1111, while the top decoder outputs are 0's.



When $w = 0$, the top decoder is enabled and the bottom is disabled.

Top decoder generates 8 minterms 0000 to 0111, while the bottom decoder outputs are 0's.

52

52

Implementing Functions with Decoders

- Implement *m* functions of *n* variables with:
  - Sum-of-minterms expressions
  - One *n*-to-2^n-line decoder
  - *m* OR gates, one for each output
- Approach
  - Find the minterms for each output function
  - OR the minterms together

53

53

Example: Full Adder with Decoder

- The sum and carry outputs of a full adder are given by:

$$S(x, y, z) = \Sigma(1, 2, 4, 7)$$
$$C(x, y, z) = \Sigma(3, 5, 6, 7)$$

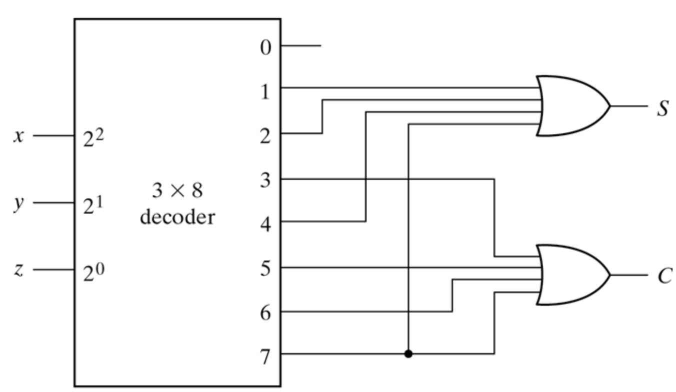
A	B	C_{in}	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

54

54

Example: Full Adder with Decoder

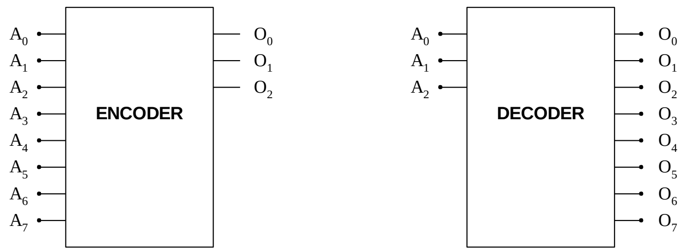
$$S(x, y, z) = \Sigma(1, 2, 4, 7)$$
$$C(x, y, z) = \Sigma(3, 5, 6, 7)$$



55

55

ENCODERS AND DECODERS



ONLY ONE INPUT ACTIVATED AT A TIME

BINARY CODE OUTPUT

BINARY CODE INPUT

ONLY ONE OUTPUT ACTIVATED AT A TIME

56

56

References

- Chapter 3 – Digital Design Morris Mano
- Template is taken from slides carnival.

57