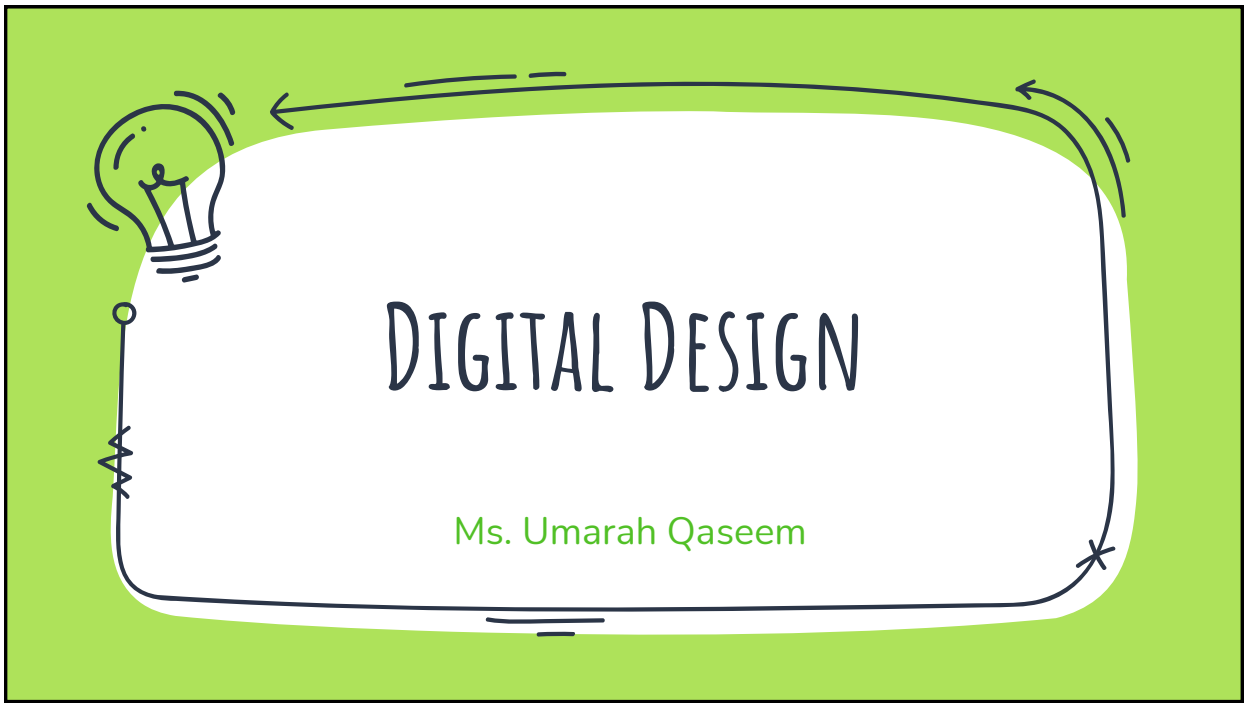
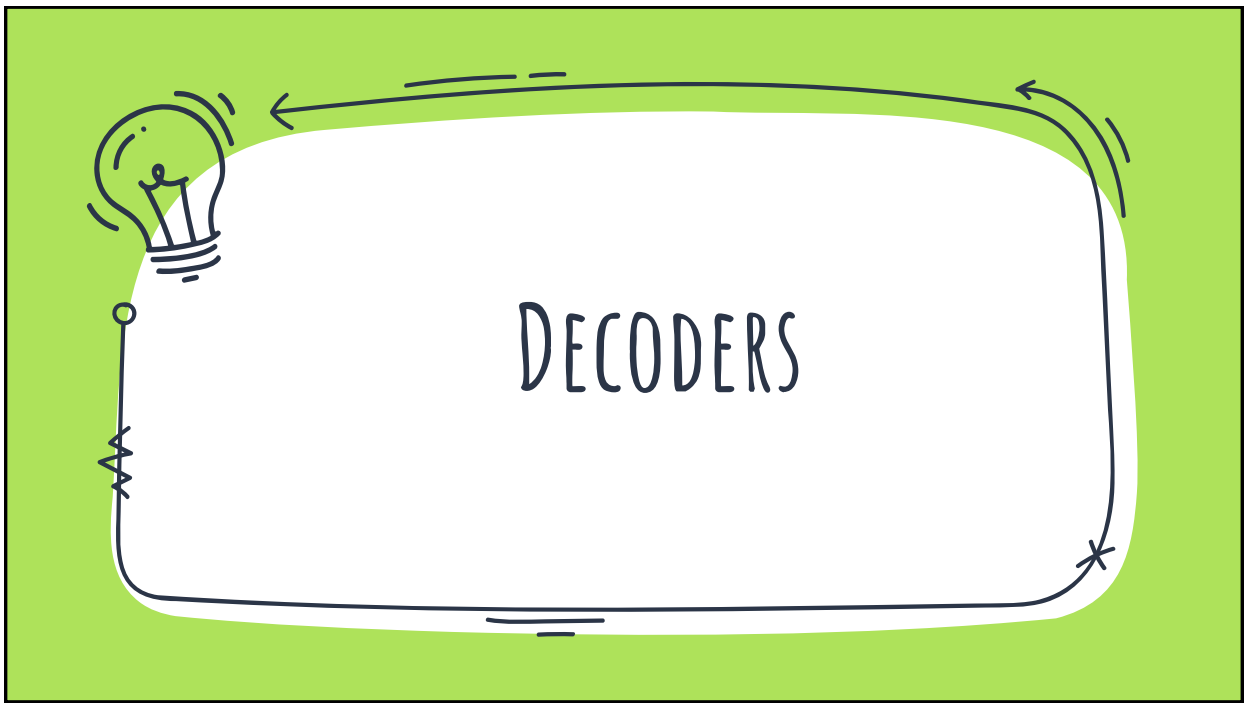


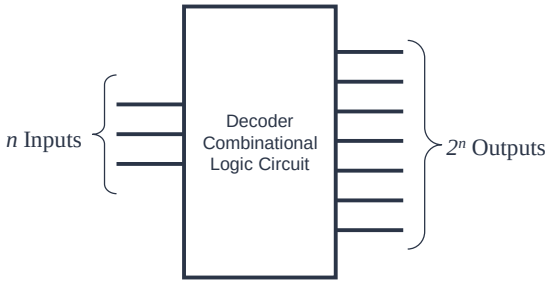
1



4

Decoders

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

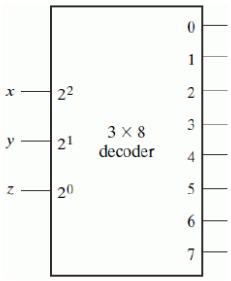


5

5

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



6

6

Decoder Examples

3-to-8-Line Decoder

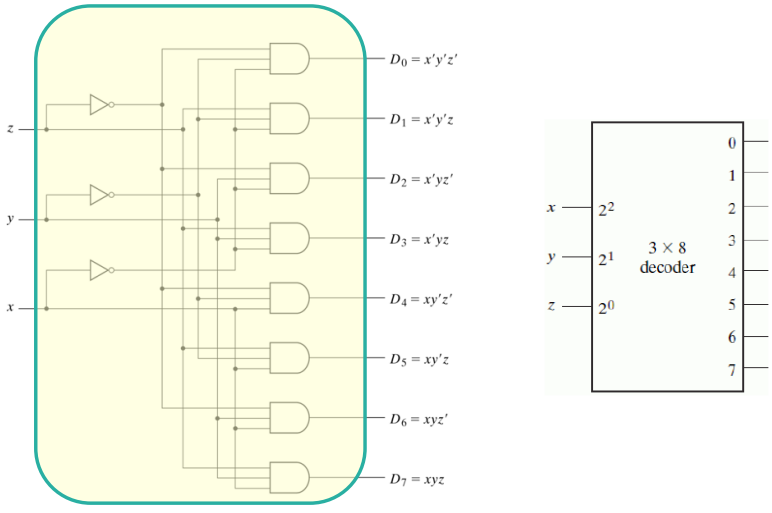
Binary Inputs			Outputs							
			D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

- 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

7

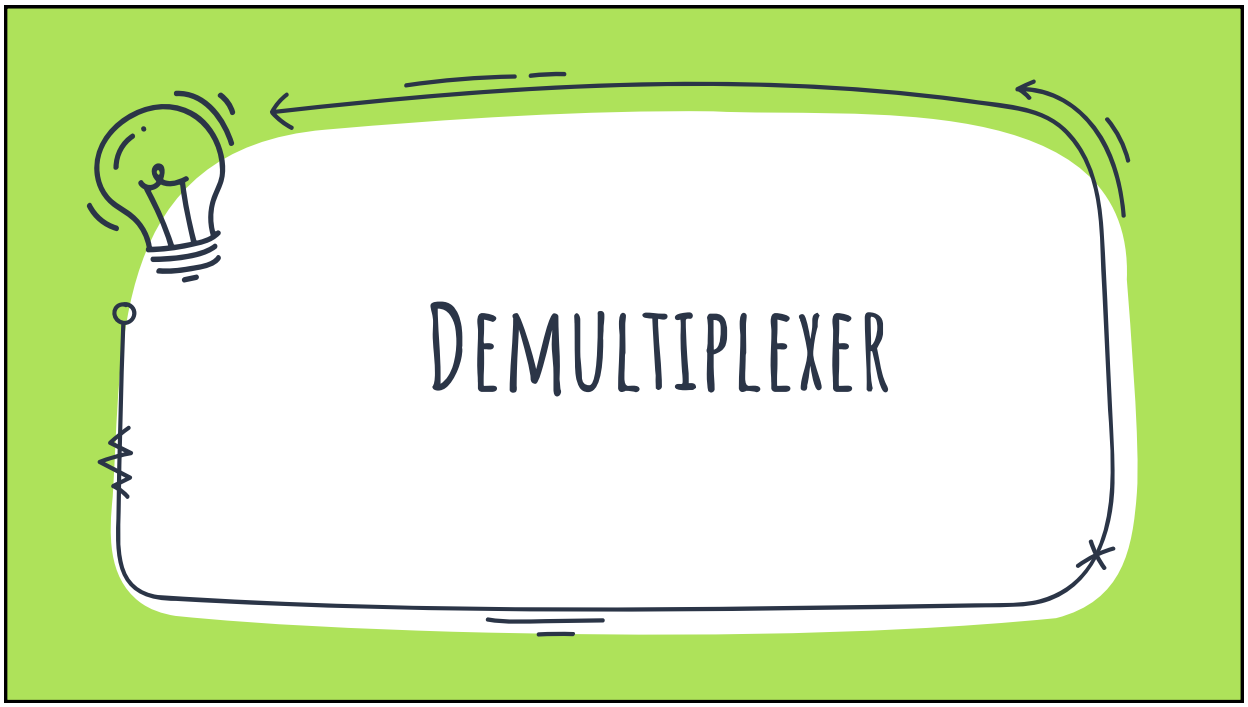
7

3-to-8-Line Decoder



8

8



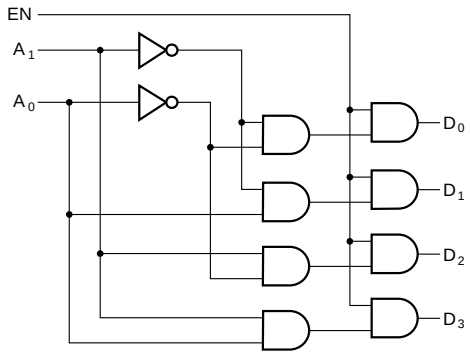
9

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 … 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)

10

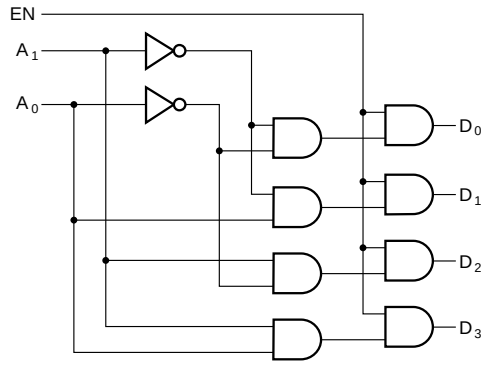
10

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)

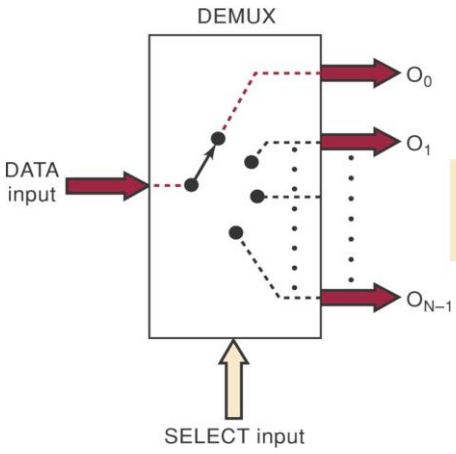


(b)

11

11

Demultiplexer



DATA input is transmitted to only one of the outputs as determined by select input code.

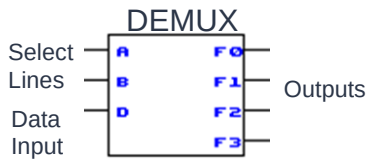
12

12

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.

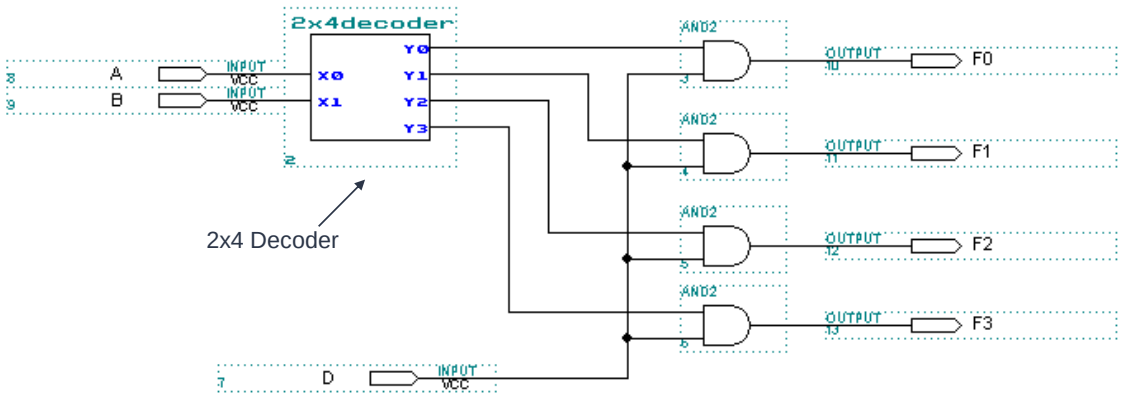
D	A	B	F3	F2	F1	F0
D	0	0	0	0	0	D
D	0	1	0	0	D	0
D	1	0	0	D	0	0
D	1	1	D	0	0	0



13

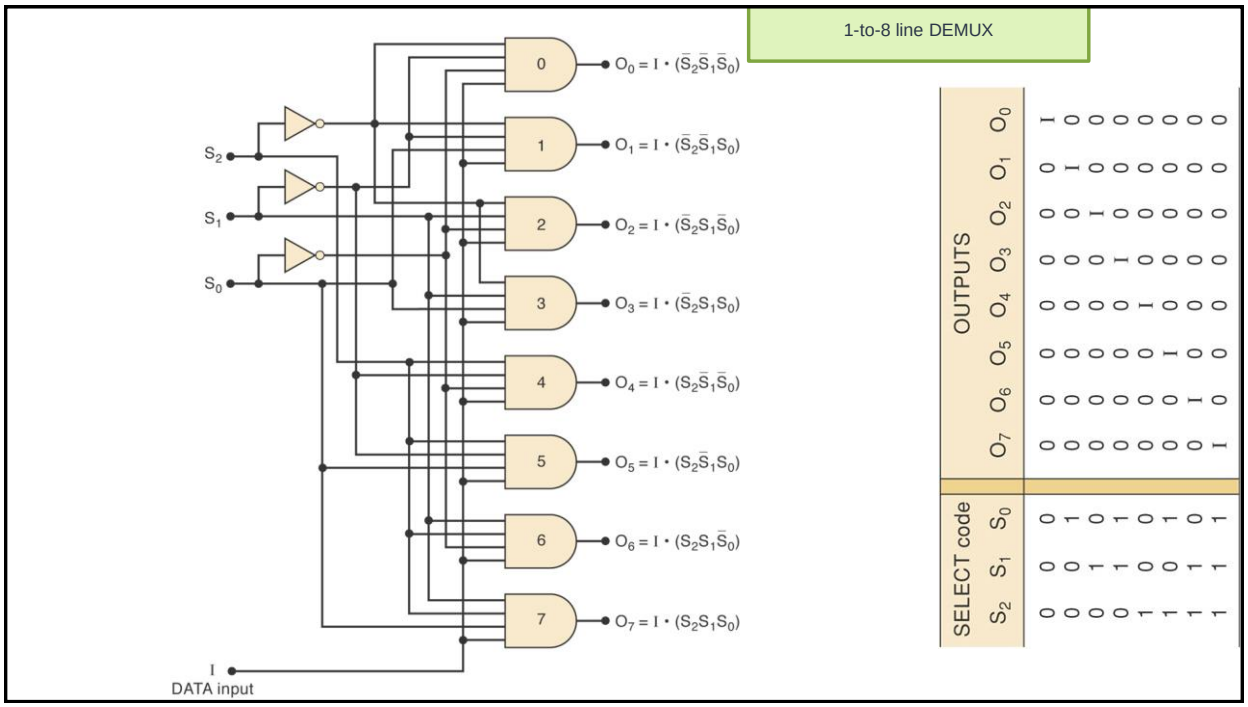
13

1 to 4 Demux



14

14



15

Decoder with Enable

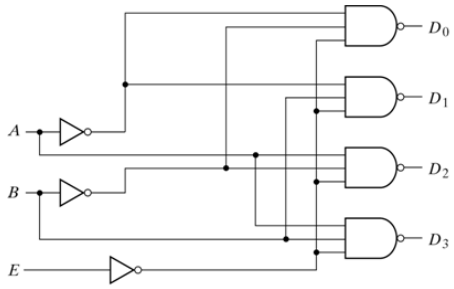
- EN is called a Control Signal
- Control Signals can be
  - Active High Signal
    - EN = 1 – Turns "ON" Decoder
  - Active Low Signal
    - EN=0 – Turns "ON" Decoder

16

16

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

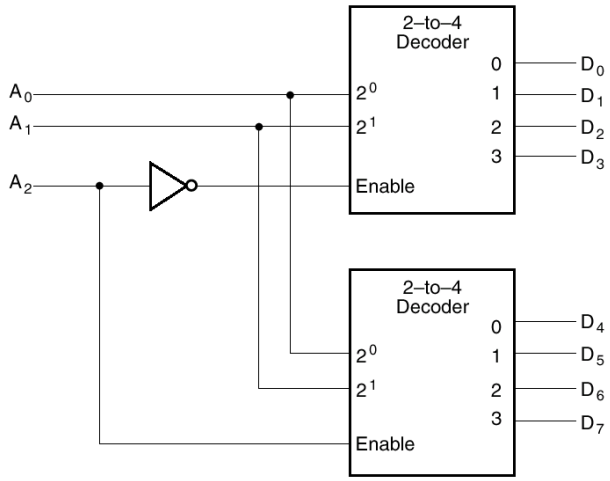
E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

(b) Truth table

17

17

Using Enable Input for Expansion

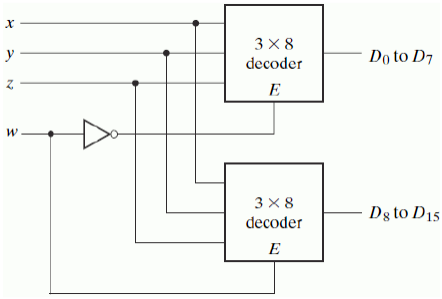


18

18

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



19

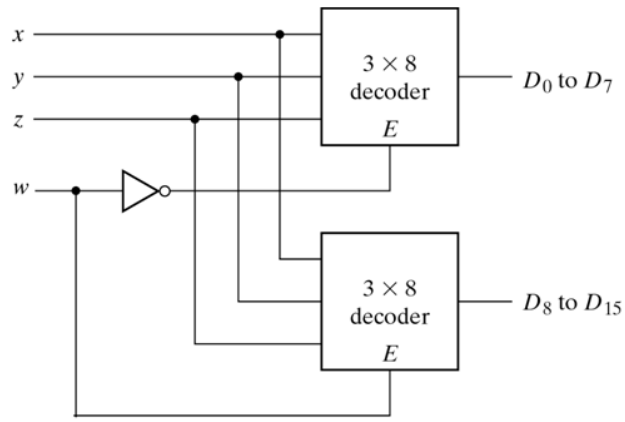
A 4x16 Decoder

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.



20

20

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

21

21

Example: Full Adder with Decoder

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

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

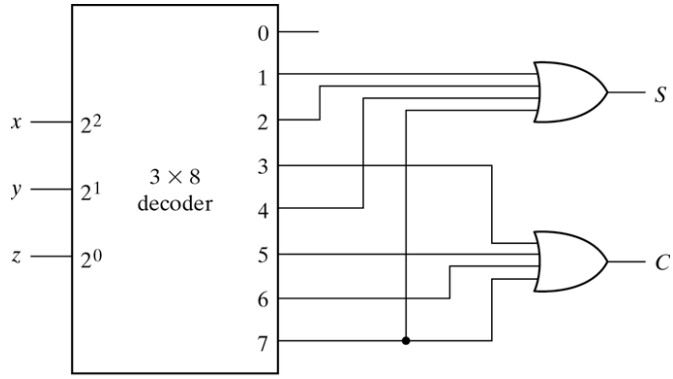
A	B	C_{in}	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

22

22

Example: Full Adder with Decoder

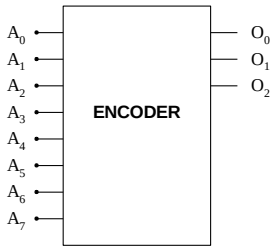
$$S(x, y, z) = \Sigma(1, 2, 4, 7)$$
$$C(x, y, z) = \Sigma(3, 5, 6, 7)$$

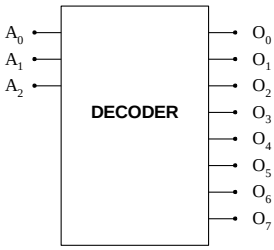


23

23

ENCODERS AND DECODERS



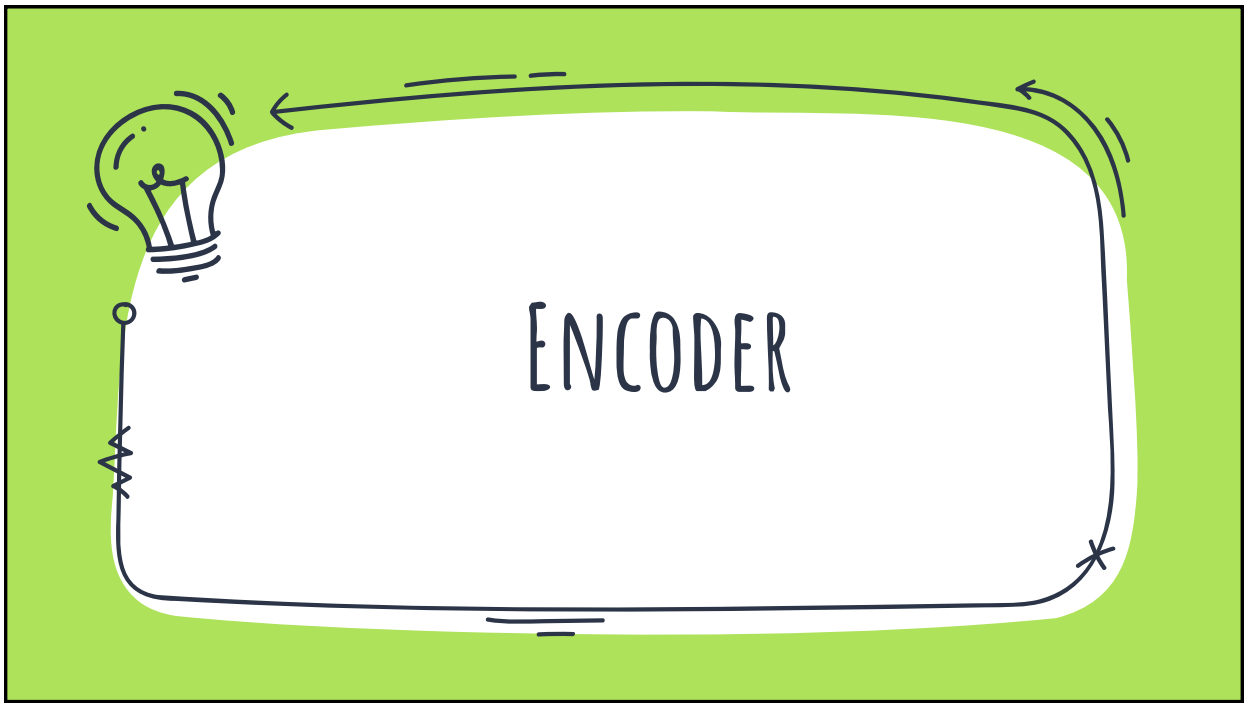


ONLY ONE INPUT ACTIVATED AT A TIME
BINARY CODE OUTPUT

BINARY CODE INPUT
ONLY ONE OUTPUT ACTIVATED AT A TIME

24

24



25

ENCODERS

- Encoding - the opposite of decoding
- Circuits that perform encoding are called *encoders*
- An encoder has 2^n (or fewer) input lines and n output lines which generate the binary code corresponding to the input values
- Typically, an encoder converts a code containing exactly one bit that is 1, to a binary code corresponding to the position in which the 1 appears.

26

26

27

27

Encoders

- *Encoder: translates 2^n input lines into n output lines*
  - Input: *2^n lines*
  - Output: *n bits*
  - Output is binary coding of input that is 1

28

28

Encoder – Truth Table

Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

29

29

Encoders

- Inputs are Minterms
- Can OR them together appropriately
- $A_0 = D_1 + D_3 + D_5 + D_7$

Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

30

30

Encoders - Activity

- Find A_1 and A_2

Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

31

31

Encoders

- $A_0 = D_1 + D_3 + D_5 + D_7$
- $A_1 = D_2 + D_3 + D_6 + D_7$
- $A_2 = D_4 + D_5 + D_6 + D_7$

Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

32

32

Encoders

Can you see any problem here?

- $A_0 = D_1 + D_3 + D_5 + D_7$
- $A_1 = D_2 + D_3 + D_6 + D_7$
- $A_2 = D_4 + D_5 + D_6 + D_7$

Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

33

33

Encoders: Problems

- Only one input can be active at a time
  - Simultaneous active inputs result in undefined output
- When all inputs are zero
  - Equal to the case when D_0 is 1
- Example
  - If D3 and D6 are active simultaneously, what is the output?
  - 111
- How can we solve this problem?
  - What should the output be if multiple lines are active?
  - Different solutions:
    - Anyone (random)
    - Give priority to lower or higher lines
    - Indicate invalid input (requires extra bit, *valid bit V*)

x = D4 + D5 + D6 + D7
y = D2 + D3 + D6 + D7
z = D1 + D3 + D5 + D7

34

Priority Encoder

- Chooses one with highest priority
  - Largest number, usually
  - To solve problem of multiple inputs

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

What if all inputs are zero?

35

35

Priority Encoder

- Add another output, why?
- To solve the problem of all 0s.

Inputs				Outputs		
D_3	D_2	D_1	D_0	A_1	A_0	V
0	0	0	0	X	X	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1

36

36

Priority Encoder – what did we learn so far?

- Simple encoder, with additional functionality
  - If multiple inputs are 1, give priority to one of them
- Example: 4-to-2 priority encoder with priority given to one bit

Valid bit

Inputs				Outputs		
D_0	D_1	D_2	D_3	x	y	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

- Which bit has the highest priority?
  - D_3

37

Priority Encoder

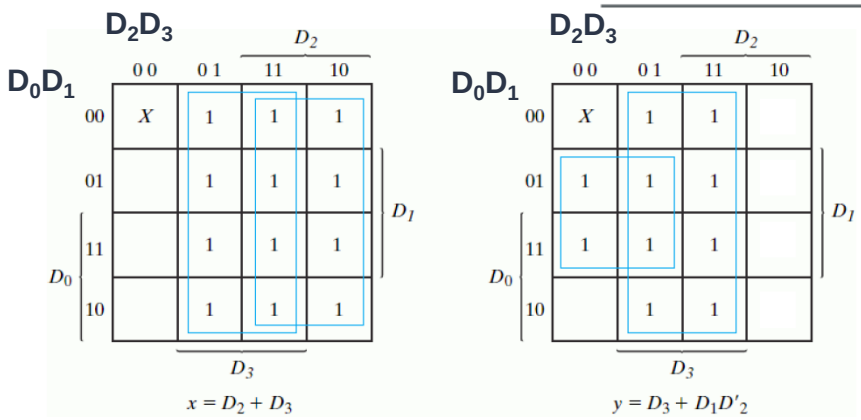
Inputs				Outputs		
D_0	D_1	D_2	D_3	x	y	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

Activity
Design it yourself

38

Priority Encoder

Inputs				Outputs		
D_0	D_1	D_2	D_3	x	y	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1



Valid bit $V = D_0 + D_1 + D_2 + D_3$

39

Priority Encoder

Inputs				Outputs		
D_3	D_2	D_1	D_0	A_1	A_0	V
0	0	0	0	X	X	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1

Activity
Design it yourself

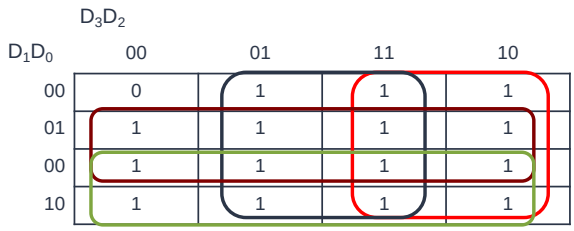
$V=D_0+D_1+D_2+D_3$

40

40

Priority Encoder

Inputs				Outputs		
D_3	D_2	D_1	D_0	A_1	A_0	V
0	0	0	0	X	X	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1



For V

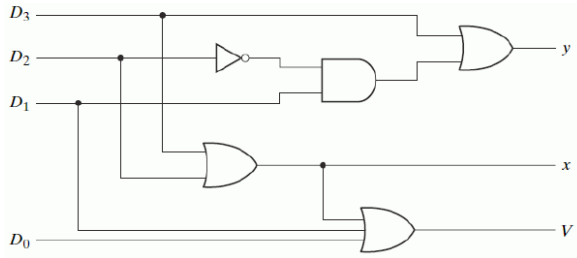
$V=D_0+D_1+D_2+D_3$

41

41

Priority Encoder

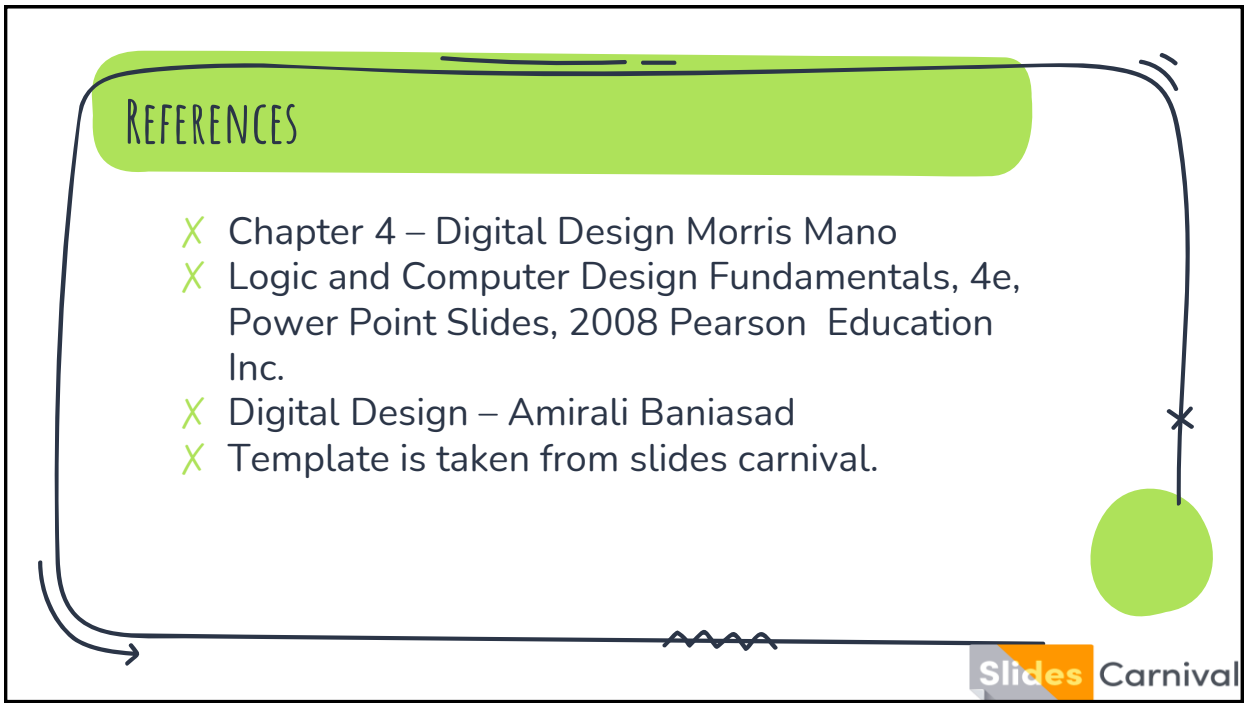
- Circuit Diagram



x = D2 + D3
y = D3 + D1 D2'
V = D0 + D1 + D2 + D3

42

42


References
Chapter 4 – Digital Design Morris Mano
Logic and Computer Design Fundamentals, 4e, Power Point Slides, 2008 Pearson Education Inc.
Digital Design – Amirali Baniasad
Template is taken from slides carnival.
Slides Carnival

46