Chapter3: Gate-Level Minimization

Lecture5- Non-degenerate Forms, Exclusive-OR Function and it's Applications in Digital Design

Engr. Arshad Nazir, Asst Prof
Dept of Electrical Engineering
SEECS

Objectives

- Understand Non-degenerate Forms and Implement Functions using these Forms at Two Levels
- Study Exclusive-OR function, and Applications in Digital Design
- Odd and Even Functions
- Parity Generation and Checking Circuits

Non-degenerate Forms

- Consider that we have four types of gates:

  AND, OR, NAND, and NOR.
- In a two-level circuit, we can have as many as 16 combinations of two-level forms:
  - Eight of these combinations are called degenerate forms because they degenerate to a single operation
    - For example, an AND-AND circuit is simple an AND of all inputs
- The eight non-degenerate forms are:

Combine '1s' on map	Combine '0s" on map
AND-OR	**OR-AND**
NAND-NAND	**NOR-NOR**
NOR-OR	**NAND-AND**
OR-NAND	**AND-NOR**

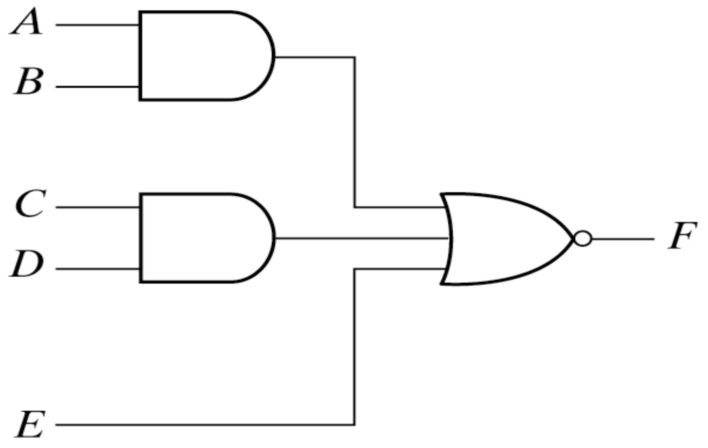
AND-OR-INVERT Implementation

- The two forms NAND-AND and AND-NOR are equivalent and can be treated together.
- Both perform the AND-OR-INVERT function
- AND-NOR resembles the AND-OR except for the inversion done by the bubble in the output of NOR gate.
- Therefore, if the complement of the function is simplified into sum-of-products form (by combining 0’s on the map, it will be possible to implement F’ with the AND-OR part of the function.
- In the figure (next slide) the function implemented is

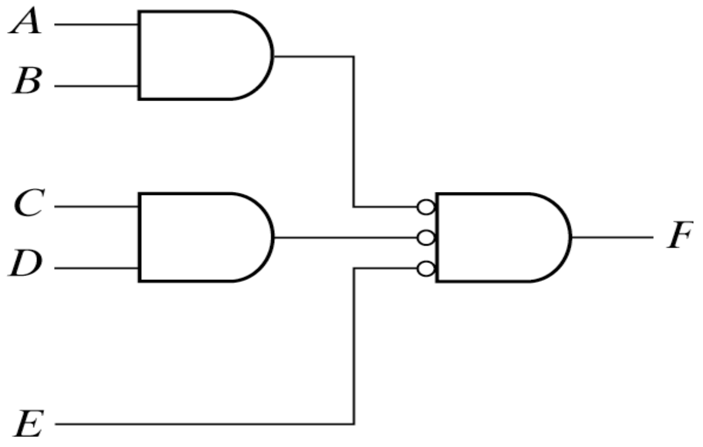
  $F = (AB+CD+E)'$

  $F' = AB+CD+E$ (sum of products for F’)

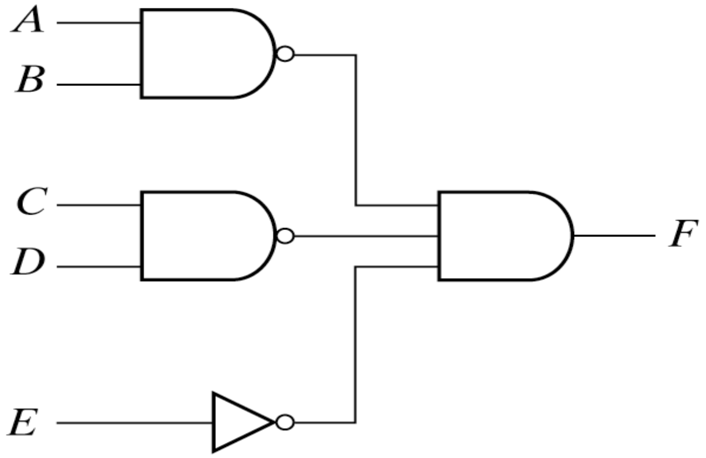
AND-OR-INVERT Implementation



(a) AND-NOR



(b) AND-NOR



(c) NAND-AND

Fig. 3-29 AND-OR-INVERT Circuits; $F = (AB + CD + E)'$

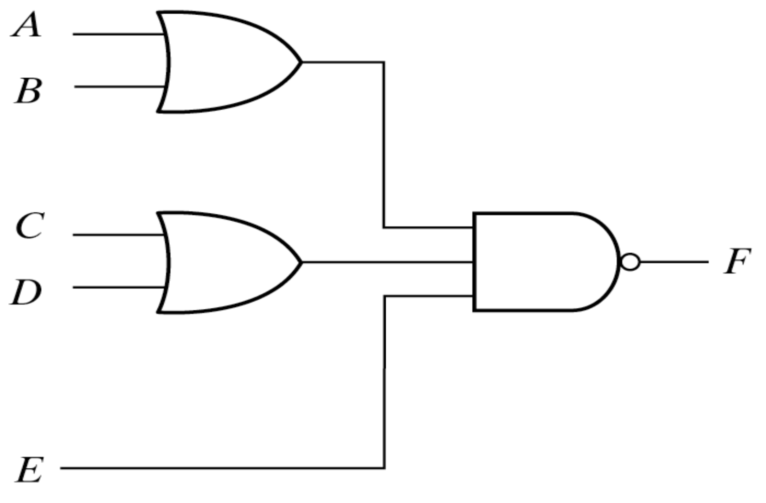
OR-AND-INVERT Implementation

- The OR-NAND and NOR-OR are equivalent and can be treated together.
- Both forms perform the OR-AND-INVERT function
- OR-NAND resembles the OR-AND except for the inversion done by the bubble in the output of NAND gate.
- Therefore, if the complement of the function is simplified into product-of-sums form (by combining 1's on map), it will be possible to implement F' with the OR-AND part of the function.
- In the figure (next slide) the function implemented is

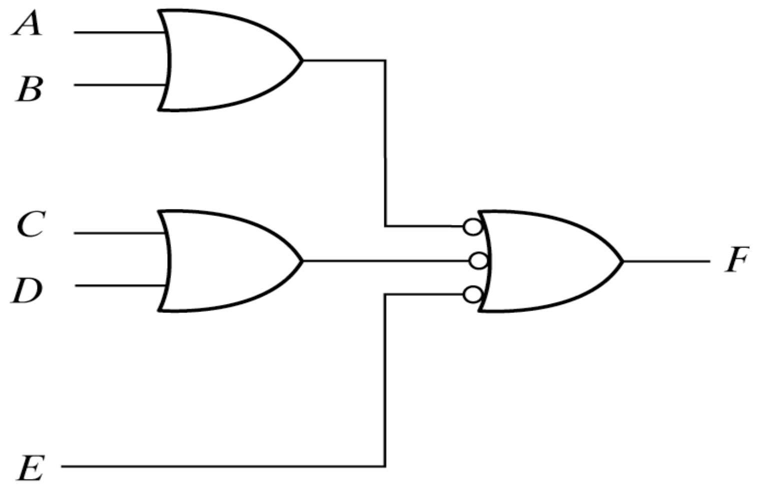
$F = [\ (A+B)(C+D)E\ ]'$

$F' = (A+B)(C+D)E$ (product of sums for F')

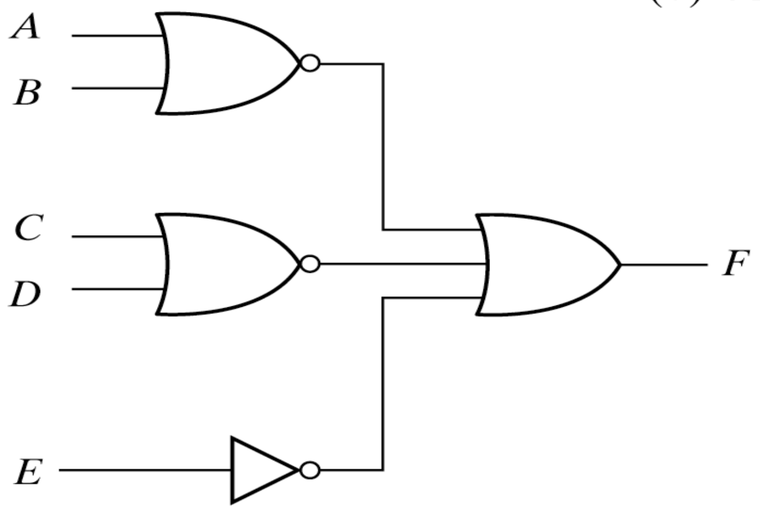
OR-AND-INVERT Implementation



(a) OR-NAND



(b) OR-NAND



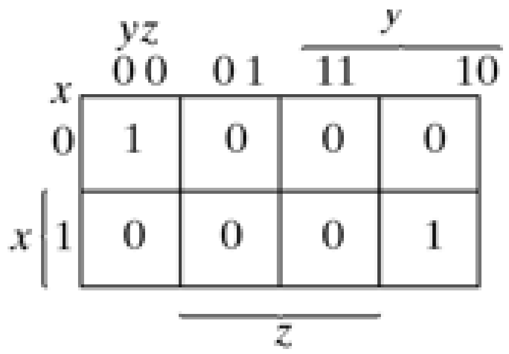
(c) NOR-OR

Fig. 3-30 OR-AND-INVERT Circuits; $F = [(A + B)(C + D)E]'$

Non-degenerate Forms

Example: Implement the following Boolean function F, using the Two-Level form of logic (a) AND-NOR (b) NAND-AND (c) OR-NAND and NOR-OR

$F(x,y,z)=\sum(1,6)$



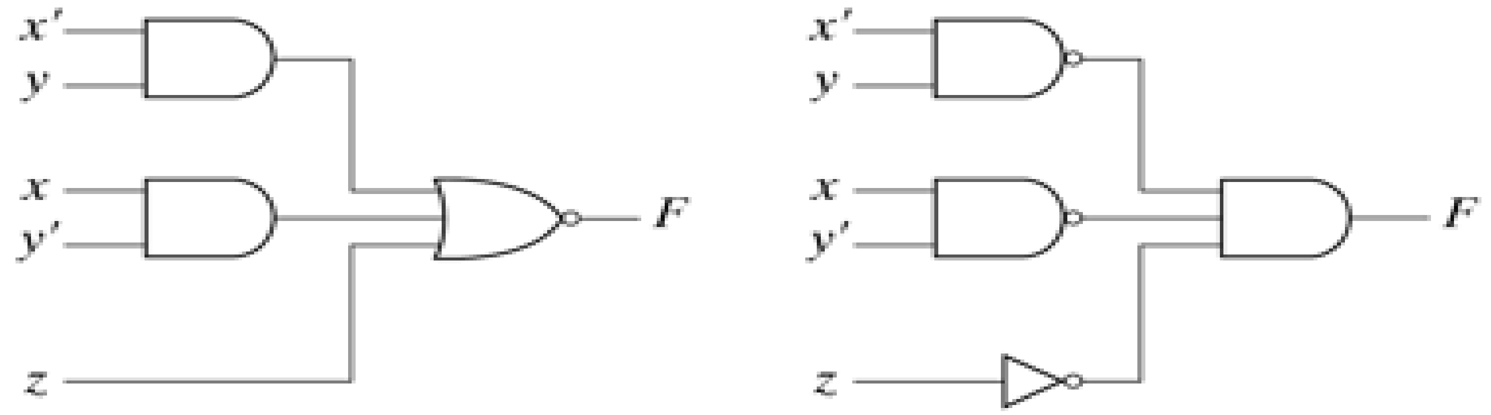
x \ yz	00	01	11	10
0	1	0	0	0
1	0	0	0	1

$F = x'y'z' + xyz'$

$F' = x'y + xy' + z$

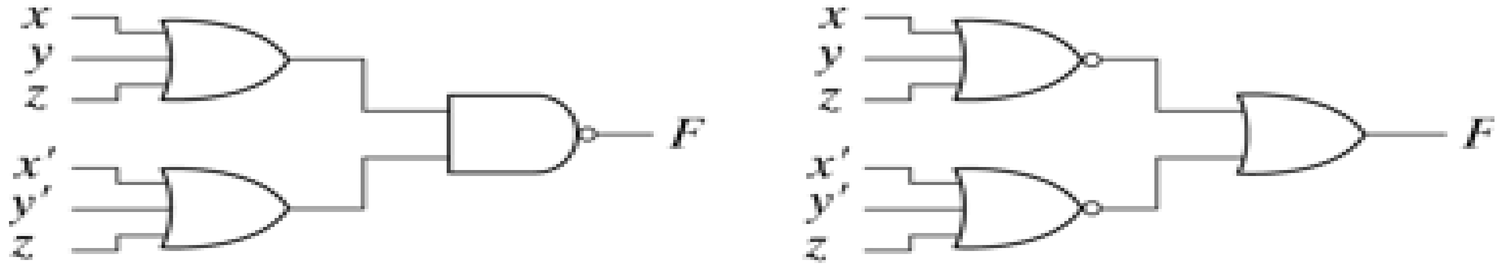
(a) Map simplification in sum of products.

Non-degenerate Forms



AND-NOR NAND-AND

(b) $F = (x'y + xy' + z)'$



OR-NAND NOR-OR

(c) $F = [(x + y + z)(x' + y' + z)]'$

Fig. 3-31 Other Two-level Implementations

Exclusive-OR Function

- Exclusive-OR (XOR) performs the following function

  $x \oplus y = xy' + x'y$

- This function is equal to one only if one of x or y is equal to one but not both.
- Exclusive NOR (XNOR) can be generated by taking the complement of an XOR operation

  $(x \oplus y)' = xy + x'y'$

- XNOR is also known as equivalence
- The following identities apply to XOR

  $x \oplus 0 = x$

  $x \oplus 1 = x'$

  $x \oplus x = 0$

  $x \oplus x' = 1$

  $x \oplus y' = x' \oplus y = (x \oplus y)'$

- XOR is also commutative and associative

  $A \oplus B = B \oplus A$

  $(A \oplus B) \oplus C = A \oplus (B \oplus C) = A \oplus B \oplus C$

Exclusive-NOR Function

$x \oplus y = xy' + x'y$

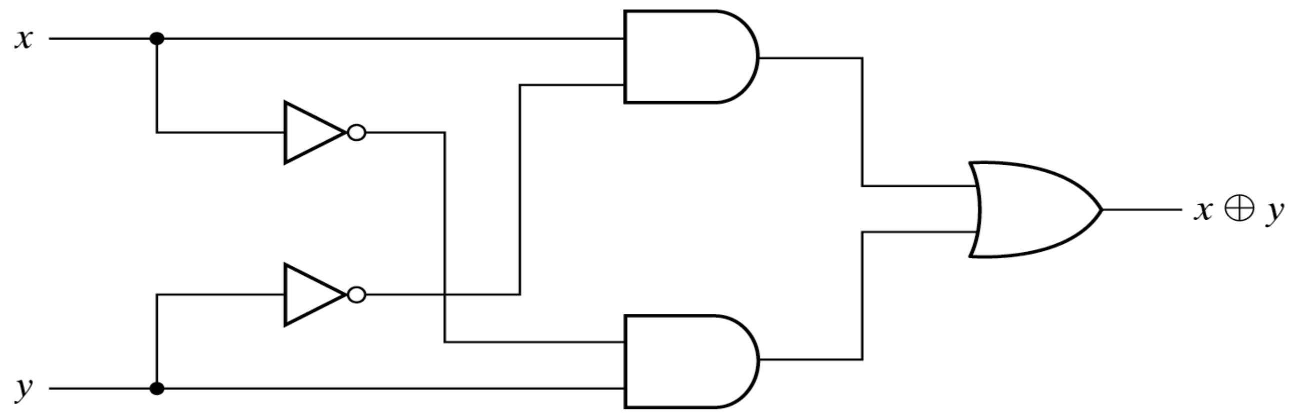
x	y	F
0	0	0
0	1	1
1	0	1
1	1	0

The complement of XOR is Exclusive-NOR:

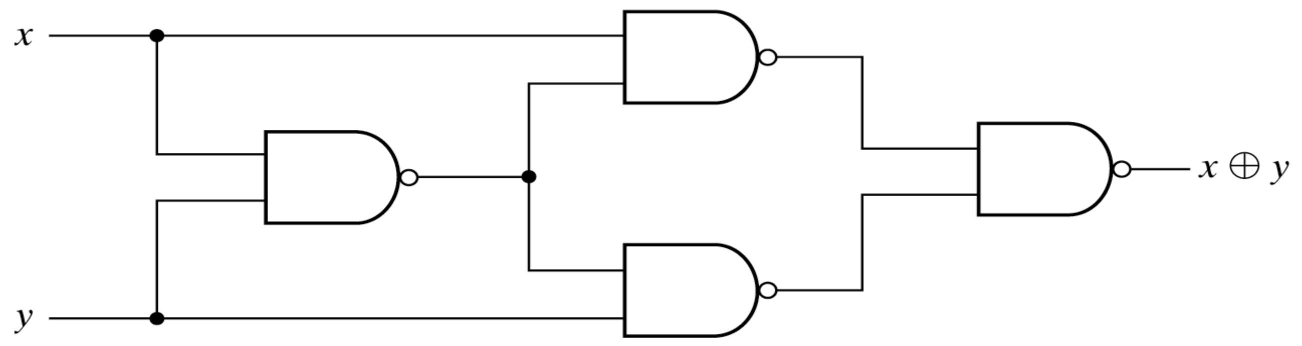
$(x \oplus y)' = xy + x'y'$

x	y	F
0	0	1
0	1	0
1	0	0
1	1	1

Exclusive-OR Implementation using NAND gates



(a) With AND-OR-NOT gates



(b) With NAND gates

Fig. 3-32 Exclusive-OR Implementations

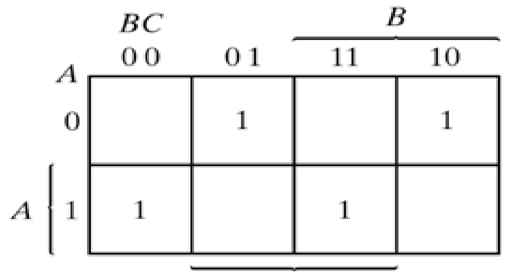
Odd and Even Functions

- The XOR operation with three or more variables can be converted into an ordinary Boolean function by replacing the $\oplus$ with its equivalent Boolean expression

  $A \oplus B \oplus C = (A \oplus B) . C' + (A \oplus B)' . C = (AB' + A'B)C' + (AB + A'B')C$ [As $(A \oplus B)' = AB + A'B'$]

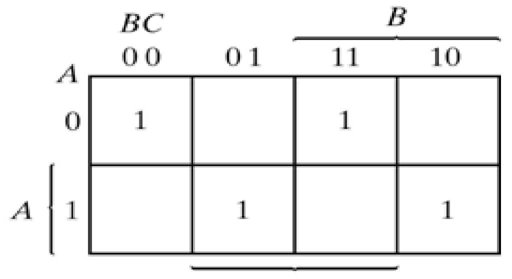
  $= AB'C' + A'BC' + ABC + A'B'C = \sum(1, 2, 4, 7)$

- This function is equal to 1 only if one variable is equal to 1 or if all three variables are equal to 1.
  - ➢ This implies that an odd number of variables must be one. This is defined as an odd function.
- The complement of an odd function is an even function.



A \ BC	00	01	11	10
0		1		1
1	1		1	

(a) Odd function
$F = A \oplus B \oplus C$



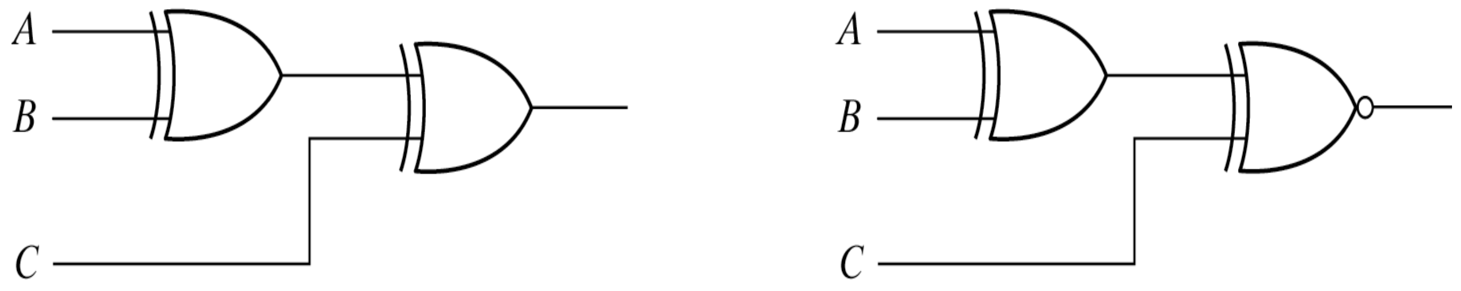
A \ BC	00	01	11	10
0	1		1	
1		1		1

(a) Even function
$F = (A \oplus B \oplus C)'$

Fig. 3-33 Map for a Three-variable Exclusive-OR Function

Odd and Even Functions Cont…

- The three input odd function is implemented by means of 2-input exclusive- OR gates
- The complement of an odd function (i.e., even function) is obtained by replacing the output gate with an exclusive- NOR gate



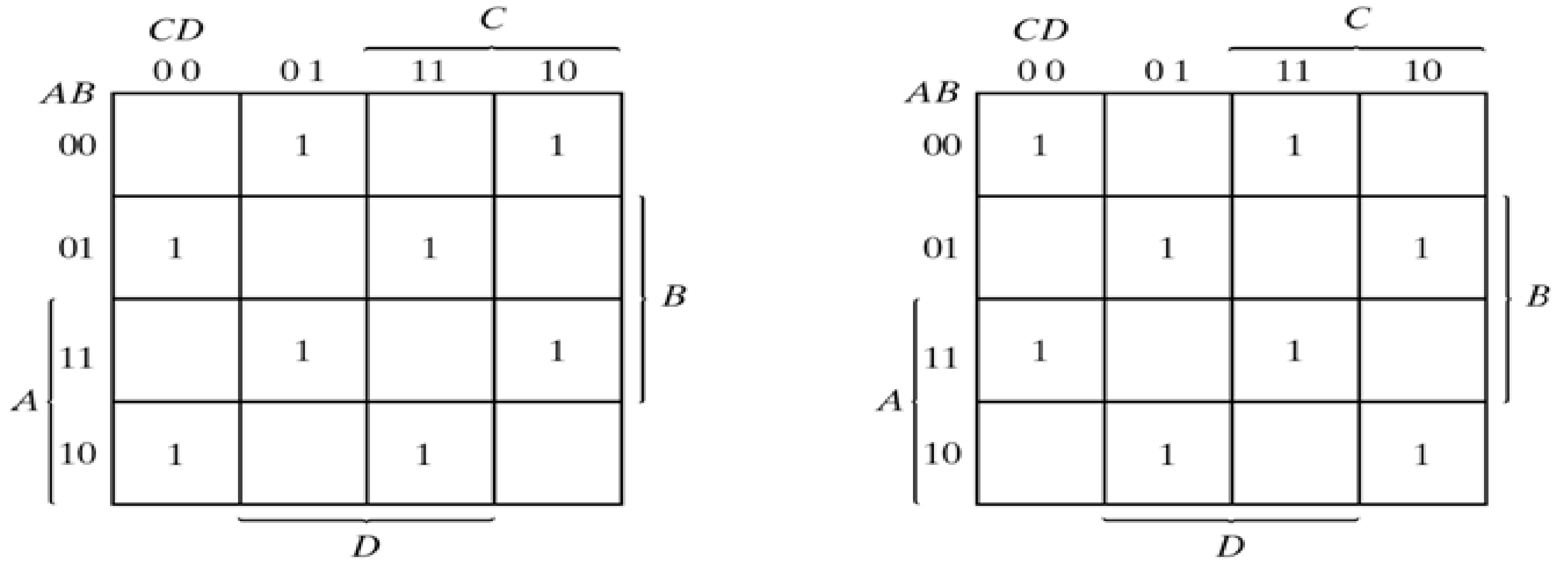
(a) 3-input odd function

(b) 3-input even function

Fig. 3-34 Logic Diagram of Odd and Even Functions

Odd and Even Functions Cont…

• The definition of Odd and Even functions can be extended to functions with more than three variables as shown in the following maps.



AB \ CD	00	01	11	10
00		1		1
01	1		1	
11		1		1
10	1		1	

(a) Odd function
$F = A \oplus B \oplus C \oplus D$

AB \ CD	00	01	11	10
00	1		1	
01		1		1
11	1		1	
10		1		1

(b) Even function
$F = (A \oplus B \oplus C \oplus D)'$

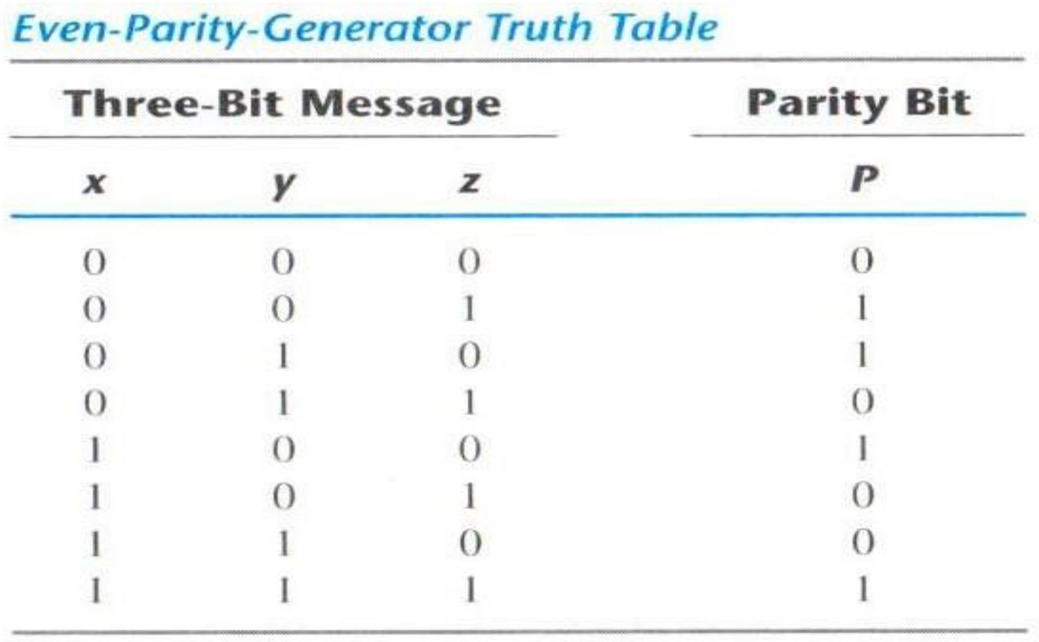
Fig. 3-35 Map for a Four-variable Exclusive-OR Function

Parity Generation and Checking

- Exclusive-OR (XOR) functions are very useful in systems requiring error-detection and correction codes.
- Parity bit is for the purpose of detecting errors. It is an extra bit added to make the total number of 1's either odd or even
- The message including the parity bit is transmitted and then checked at the receiving end for errors
- An error is detected if the checked parity does not correspond with the one transmitted
  - ➢ A circuit that generates a parity bit is called a parity generator.
  - ➢ The circuit that checks the parity is called a parity checker.

Parity Generation and Checking Cont…

- Following is the truth table for even parity generator

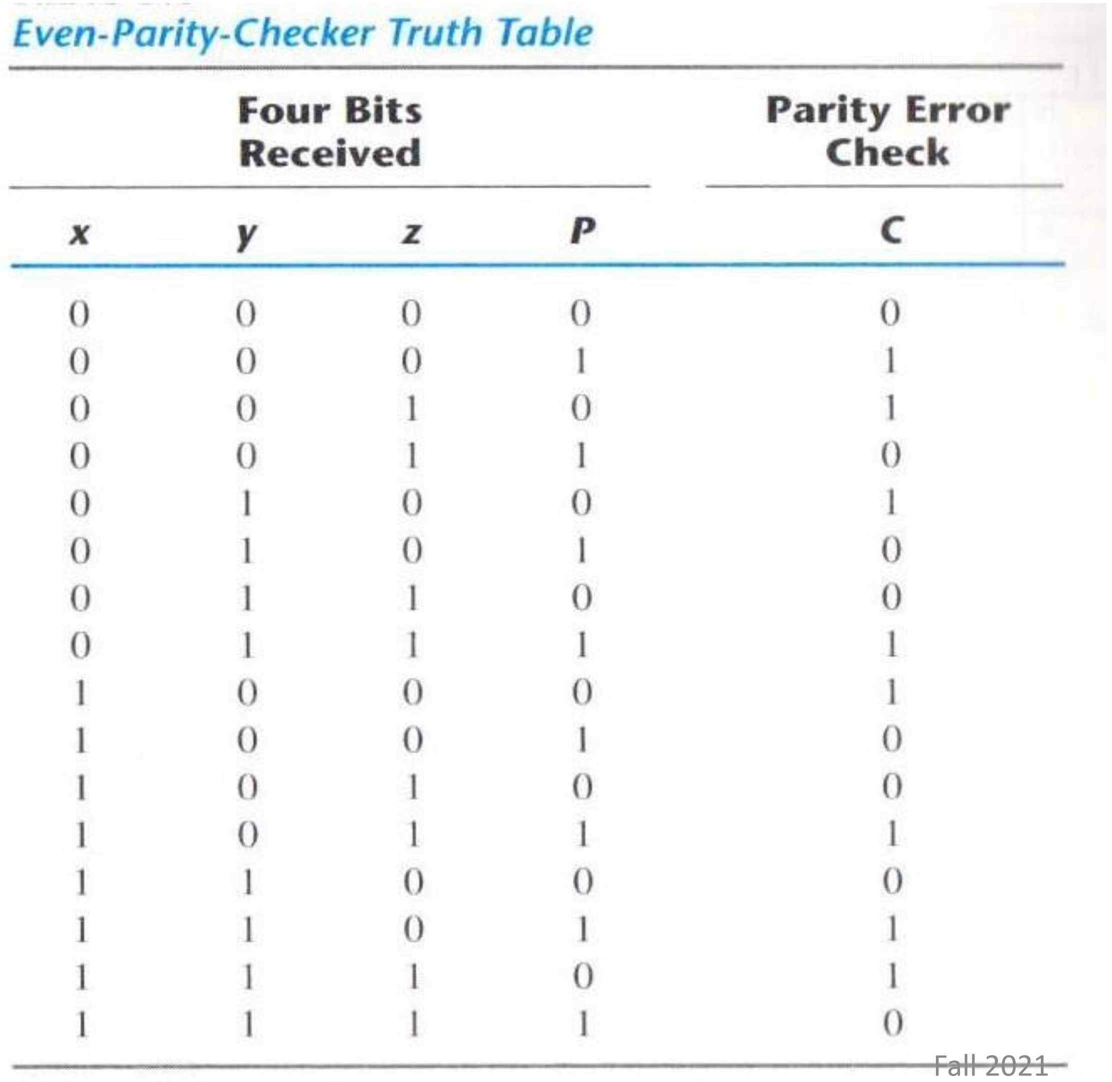


Even-Parity-Generator Truth Table

Three-Bit Message			Parity Bit
x	*y*	*z*	*P*
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

- The three bits x, y and z constitute the message and are inputs. The parity bit P is the output
- P must be generated to make the total number of 1's even. So P constitutes the odd function (three variable exclusive- OR function)

Parity Generation and Checking Cont…



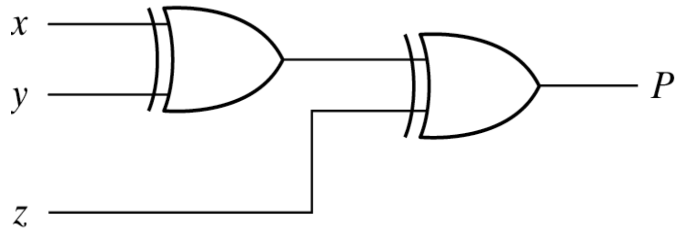
Even-Parity-Checker Truth Table

Four Bits Received				Parity Error Check
x	*y*	*z*	*P*	*C*
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

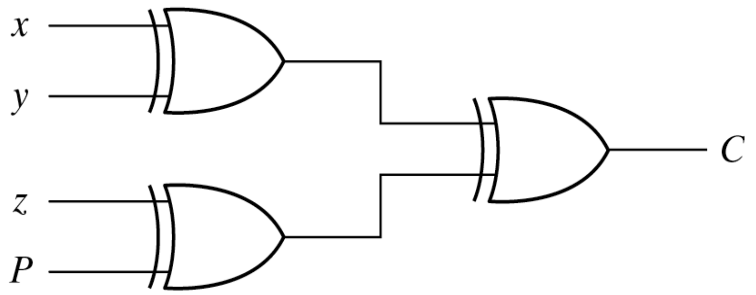
•The four bits are received by the parity checker circuit.
•The same parity technique is used both at transmitting and receiving terminals.
•The parity checker circuit will output 1 whenever, the odd number of input variables are 1 i.e even parity.
•Truth Table for even parity checker is listed to the left

Parity Generation and Checking cont…

- The parity checker can also be implemented with exclusive- OR gates
- parity check: C = x ⊕ y Å z Å P
  - ➢ C=1: an odd number of data bit error
  - ➢ C=0: correct or an even number of data bit error



(a) 3-bit even parity generator



(a) 4-bit even parity checker

Fig. 3-36 Logic Diagram of a Parity Generator and Checker

The End