XMUT 101
Engineering Technology

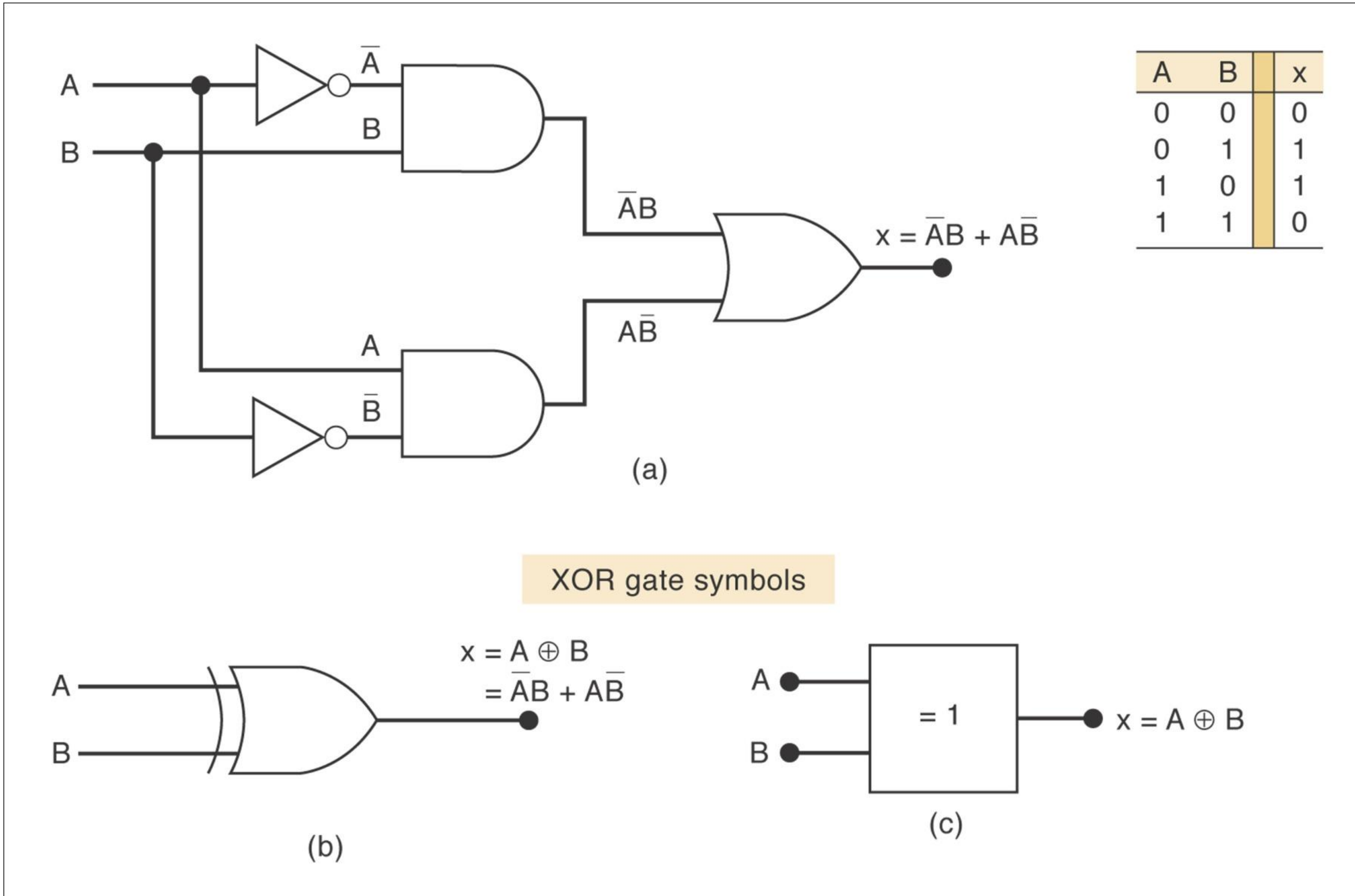
A/Prof. Pawel Dmochowski

School of Engineering and Computer Science
Victoria University of Wellington

Exclusive OR and Exclusive NOR Circuits

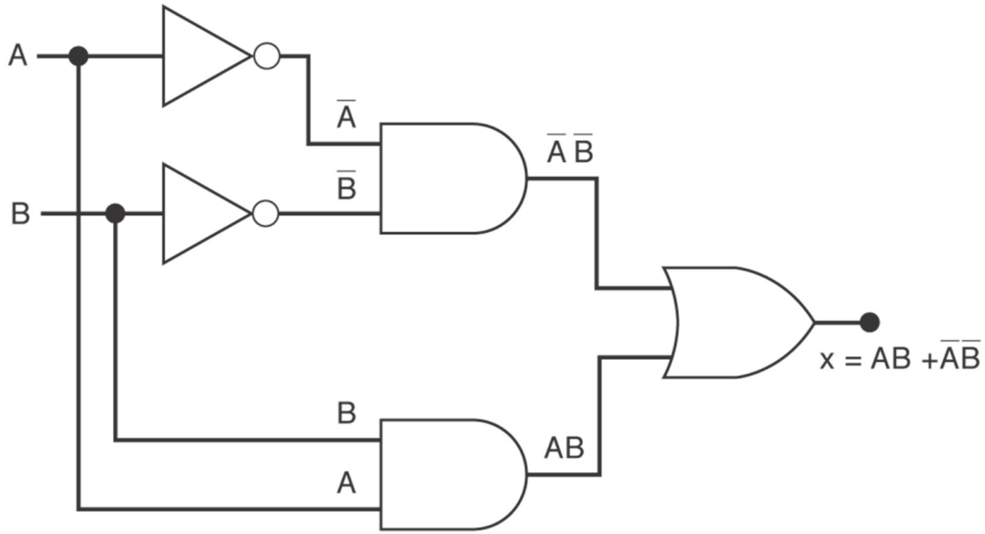
- The exclusive OR, abbreviated XOR produces a HIGH output whenever the two inputs are at opposite levels.
- The exclusive NOR, abbreviated XNOR produces a HIGH output whenever the two inputs are at the same level.
- XOR and XNOR outputs are opposite.

FIGURE 4-20 (a) Exclusive-OR circuit and truth table; (b) traditional XOR gate symbol; (c) IEEE/ANSI symbol for XOR gate.



A	B	x
0	0	0
0	1	1
1	0	1
1	1	0

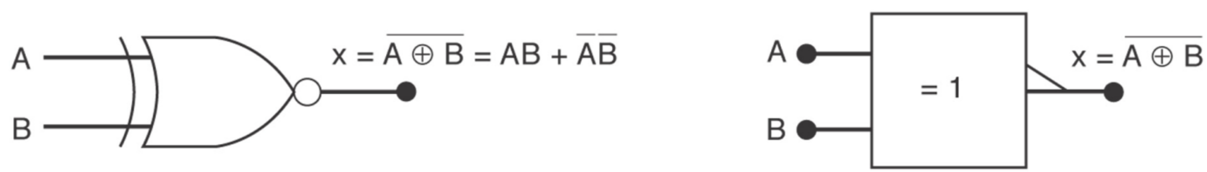
FIGURE 4-21 (a) Exclusive-NOR circuit; (b) traditional symbol for XNOR gate; (c) IEEE/ANSI symbol.



(a)

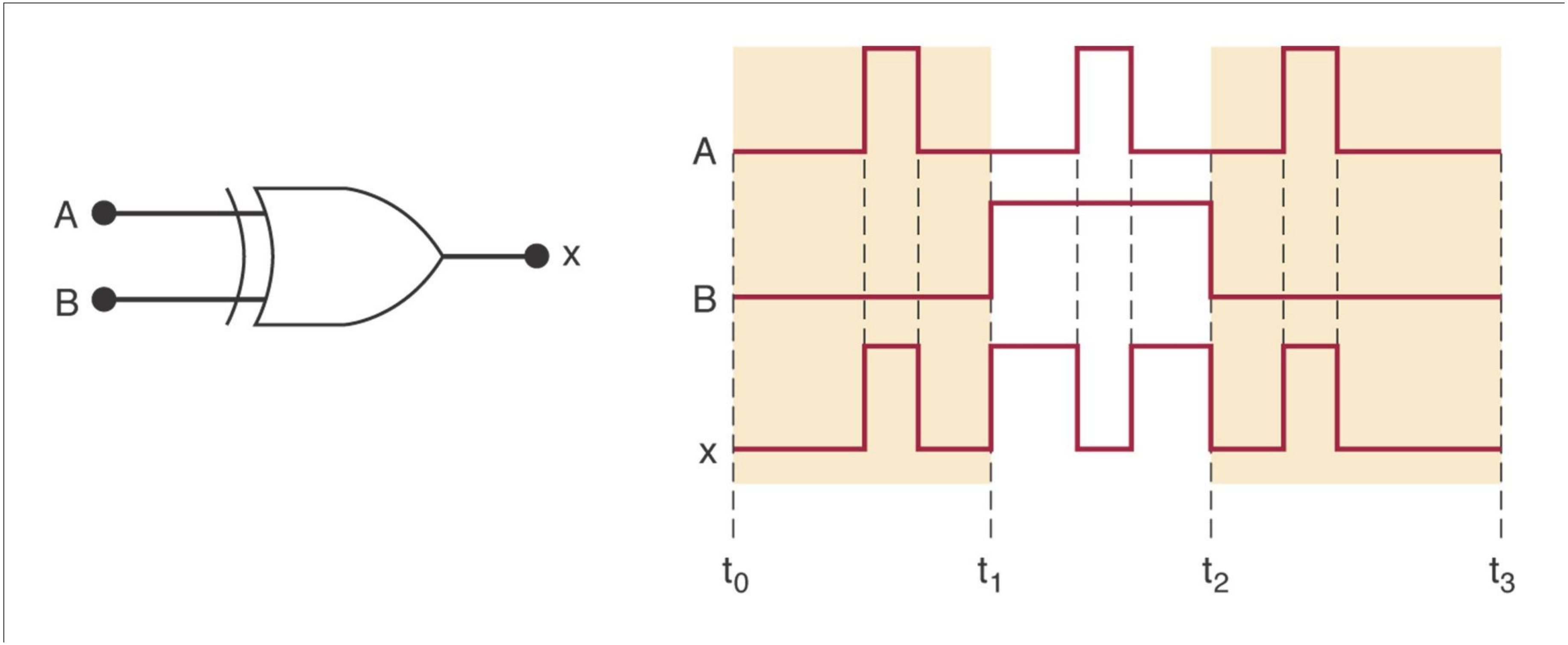
A	B	x
0	0	1
0	1	0
1	0	0
1	1	1

XNOR gate symbols



(b) (c)

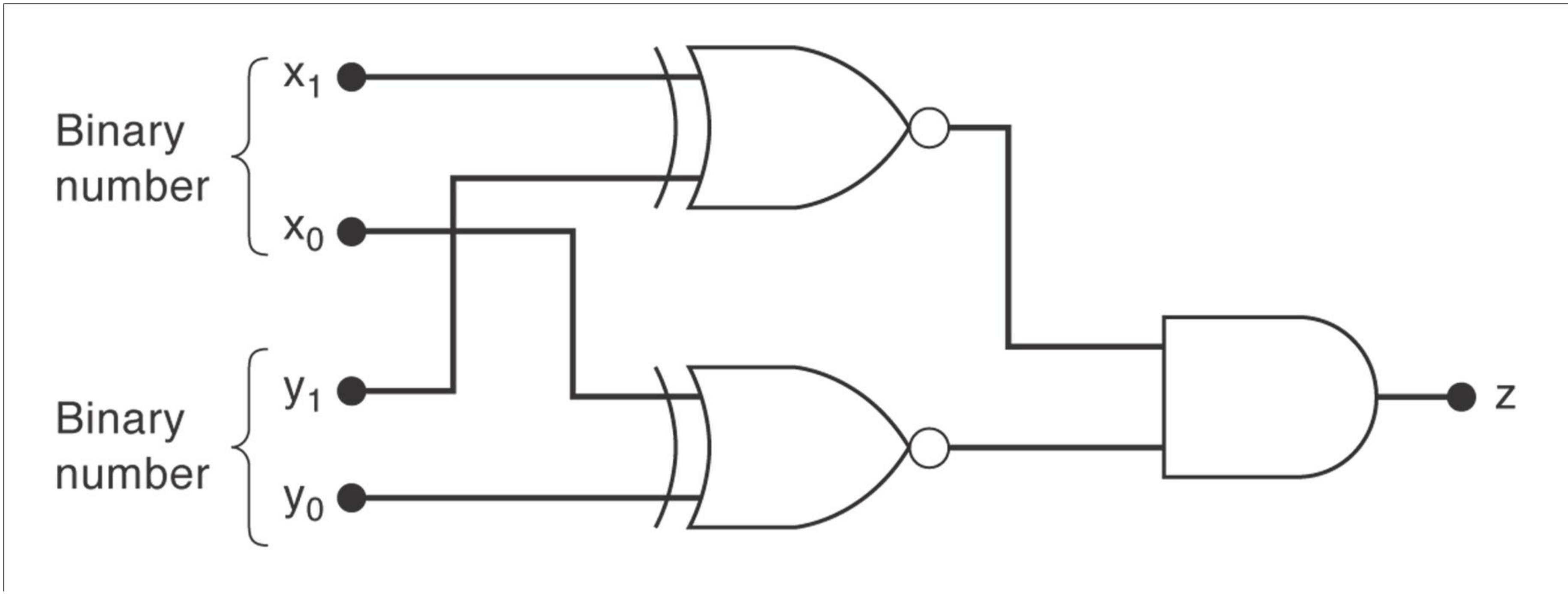
Determine the O/P waveform of the circuit below:



O/P Hi when I/P at different levels

Design a circuit so that the O/P will only be HI when the combination of two sets of two bit binary numbers are equal.

x_1	x_0	y_1	y_0	z (Output)
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1



Binary number
x_1
x_0
Binary number
y_1
y_0
z

Parity Generator and Checker

- Parity bit: extra bit added to data to make the number of 1's even (for even parity) or odd (for odd parity)
- It is used to detect error in transmission
- Example: if we use an even parity system:
  - Data: 1 1 1 0 – we add a parity bit 1
  - Data: 1 1 0 0 – we add parity bit 0
  - Data: 0 0 0 0 – we add a parity bit 0

- A Parity Checker will return a TRUE error bit if the number of 1's is odd (for even parity) and if the number of 1's is even (for odd parity)

Parity Generator

$$A B \oplus A B = \bar{A} B + A \bar{B} \qquad \overline{A B \oplus A B} = \bar{A} \bar{B} + A B$$

- How to construct an even parity generator (3 bits input)?
- Truth table:

A	B	C	P
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Parity Generator

$$AB \oplus AB = \bar{A}B + A\bar{B} \qquad \overline{AB \oplus AB} = \bar{A}\bar{B} + AB$$

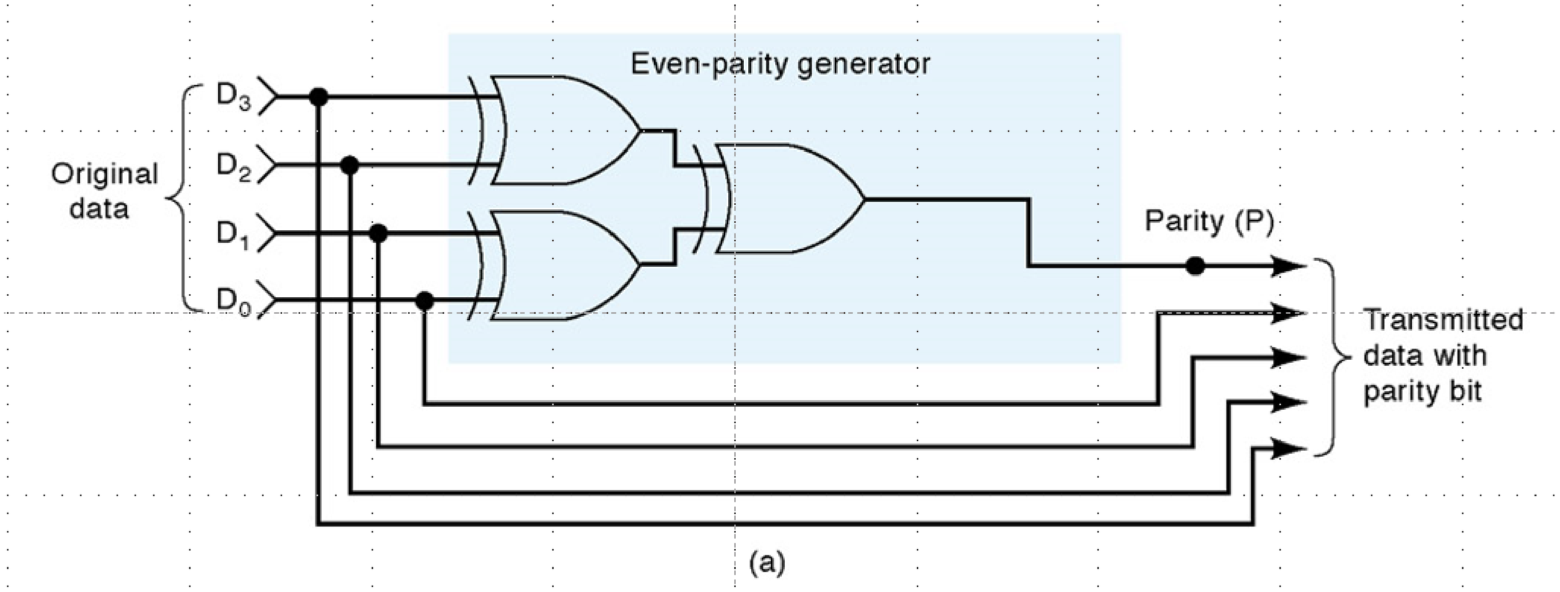
- How to construct an even parity generator (3 bits input)?
- Truth table:

A	B	C	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

$$\begin{aligned} P &= \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC \\ &= \bar{A}(\bar{B}C + B\bar{C}) + A(\bar{B}\bar{C} + BC) \\ &= \bar{A}(BC \oplus BC) + A(\overline{BC \oplus BC}) \\ &= \bar{A}X + A\bar{X} \\ &= A \oplus B \oplus C \end{aligned}$$

Parity Generator and Checker

Similarly for 4 bits (even parity):



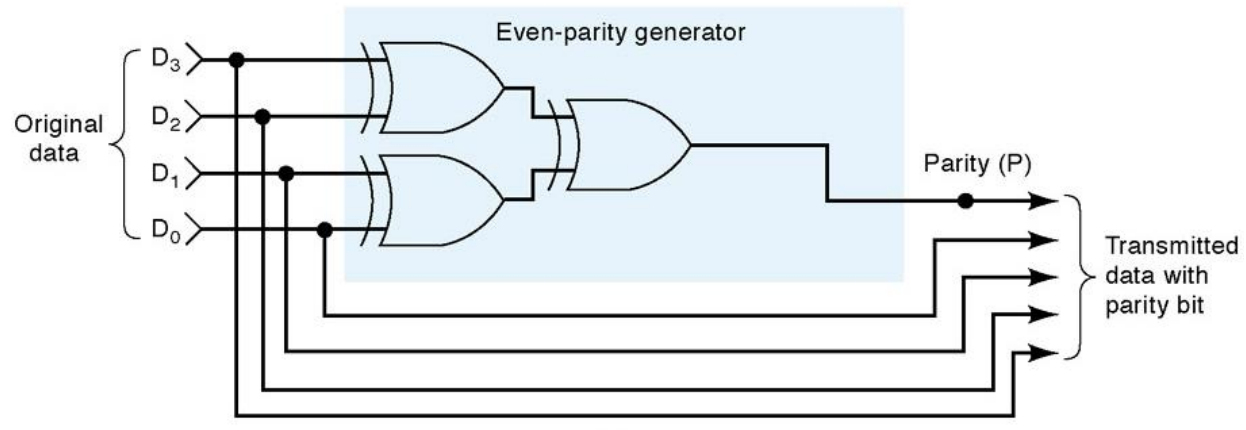
Parity Checker

- Exercise:
  - Design an even parity **checker** (2 data bits) using a truth table
  - Express it using XOR or XNOR gates

A	B	P	E
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

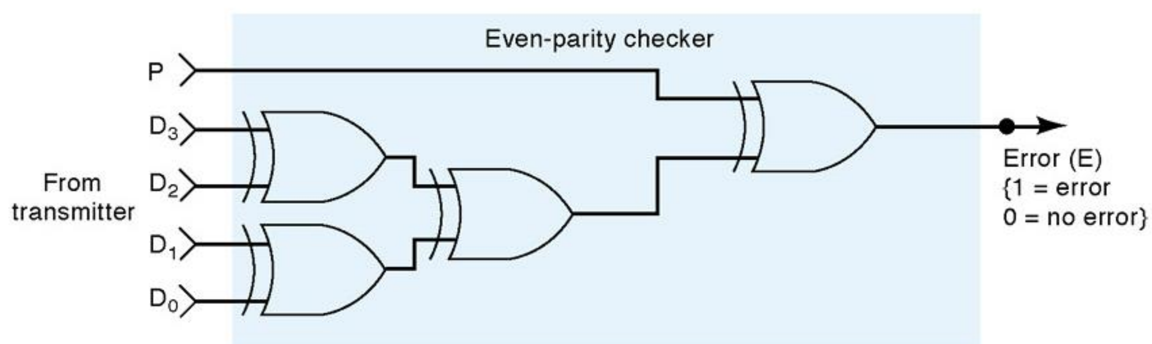
Parity Generator and Checker

Similarly for 4 bits:



(a)

Even Parity:



(b)

Karnaugh Map (K-Map)

- An alternate approach to representing Boolean functions
- Can be used to minimize Boolean functions
- Easy conversion from truth table to K-Map to minimized SOP representation.
- Simple rules (steps) used to perform minimization
- Leads to minimized SOP representation.
  - Much faster and more efficient than previous minimization techniques with Boolean algebra.

Karnaugh Map (K-Map) Method

- The truth table values are placed in the K map as shown below.

B / A	B'	B
A'	1	1
A	0	0

C / AB	C'	C
A'B'		
A'B		
AB		
AB'		

Karnaugh Map (K-Map) Method

- The truth table values are placed in the K map.
- Adjacent K map square differ in only one variable both horizontally and vertically.

Karnaugh Map (K-Map) Method

- The truth table values are placed in the K map.
- Adjacent K map square differ in only one variable both horizontally and vertically.
- The pattern from top to bottom and left to right must be in the form $\overline{A}\,\overline{B}, \overline{A}B, AB, A\overline{B}$

2 inputs:

B / A	B'	B
A'	1	1
A	0	0

4 inputs:

CD / AB	C'D'	C'D	CD	CD'
A'B'				
A'B				
AB				
AB'				

Karnaugh Map (K-Map) Method

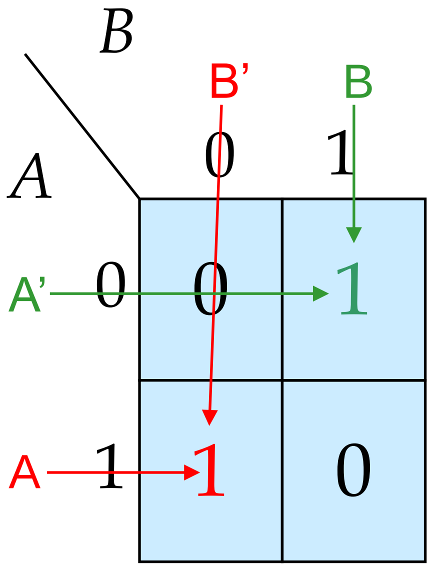
- The truth table values are placed in the K map as shown in on next slide.
- Adjacent K map square differ in only one variable both horizontally and vertically.
- The pattern from top to bottom and left to right must be in the form $\overline{A}\,\overline{B}, \overline{A}B, AB, A\overline{B}$

- A SOP expression can be obtained by OR-ing all squares that contain a 1.

Karnaugh Maps

- A Karnaugh map is a graphical tool for assisting in the general simplification procedure.
- **Two variable maps.**

Example 1:

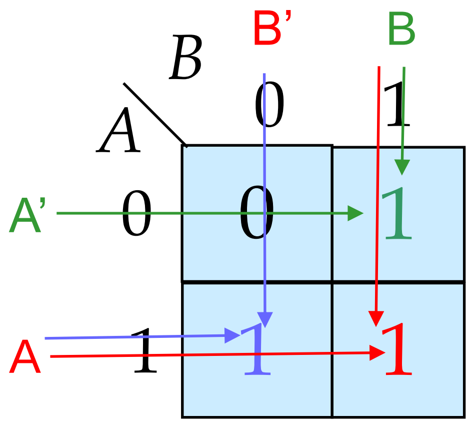


$$F = AB' + A'B$$

Karnaugh Maps

- A Karnaugh map is a graphical tool for assisting in the general simplification procedure.
- **Two variable maps.**

Example 2:



$$F = AB + A'B + AB'$$

Karnaugh Maps

- **Three variable maps.**

Inputs

Output

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

$$F = A'B'C + A'BC' + AB'C' + AB'C + ABC' + ABC$$

Sum of Products expression (SOP)

Karnaugh Maps

- **Three variable maps.**

A \ BC	B′C′	B′C	BC	BC′
A′	0	1	0	1
A	1	1	1	1

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

$$F = A'B'C + A'BC' + AB'C' + AB'C + ABC' + ABC$$

Rules for K-Maps

- We can reduce functions by circling 1's in the K-map
- Each circle represents min-term reduction
- Following circling, we can deduce minimized and-or form.

Rules to consider

- Every cell containing a 1 must be included at least once.
- The largest possible "power of 2 rectangle" must be enclosed.
- The 1's must be enclosed in the smallest possible number of rectangles.

K-Maps and truth tables for (a) two variables.

A	B	X
0	0	1 → $\bar{A}\bar{B}$
0	1	0
1	0	0
1	1	1 → AB

$$\left\{ x = \bar{A}\bar{B} + AB \right\}$$

	$\bar{B}$	B
$\bar{A}$	1	0
A	0	1

(a)

K-Maps and truth tables for (b) three variables.

A	B	C	X	
0	0	0	1	$\rightarrow \overline{A}\,\overline{B}\,\overline{C}$
0	0	1	1	$\rightarrow \overline{A}\,\overline{B}C$
0	1	0	1	$\rightarrow \overline{A}B\overline{C}$
0	1	1	0	
1	0	0	0	
1	0	1	0	
1	1	0	1	$\rightarrow AB\overline{C}$
1	1	1	0	

$$\left\{ \begin{aligned} X = {} & \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C \\ & + \overline{A}B\overline{C} + AB\overline{C} \end{aligned} \right\}$$

	$\overline{C}$	C
$\overline{A}\,\overline{B}$	1	1
$\overline{A}B$	1	0
AB	1	0
$A\overline{B}$	0	0

(b)

K-Maps and truth tables for (c) four variables.

A	B	C	D	X	
0	0	0	0	0	
0	0	0	1	1	$\rightarrow \bar{A}\bar{B}\bar{C}D$
0	0	1	0	0	
0	0	1	1	0	
0	1	0	0	0	
0	1	0	1	1	$\rightarrow \bar{A}B\bar{C}D$
0	1	1	0	0	
0	1	1	1	0	
1	0	0	0	0	
1	0	0	1	0	
1	0	1	0	0	
1	0	1	1	0	
1	1	0	0	0	
1	1	0	1	1	$\rightarrow AB\bar{C}D$
1	1	1	0	0	
1	1	1	1	1	$\rightarrow ABCD$

$$\left\{ \begin{aligned} X = {} & \bar{A}\bar{B}\bar{C}D + \bar{A}B\bar{C}D \\ & + AB\bar{C}D + ABCD \end{aligned} \right\}$$

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	1	0	0
$\bar{A}B$	0	1	0	0
AB	0	1	1	0
$A\bar{B}$	0	0	0	0

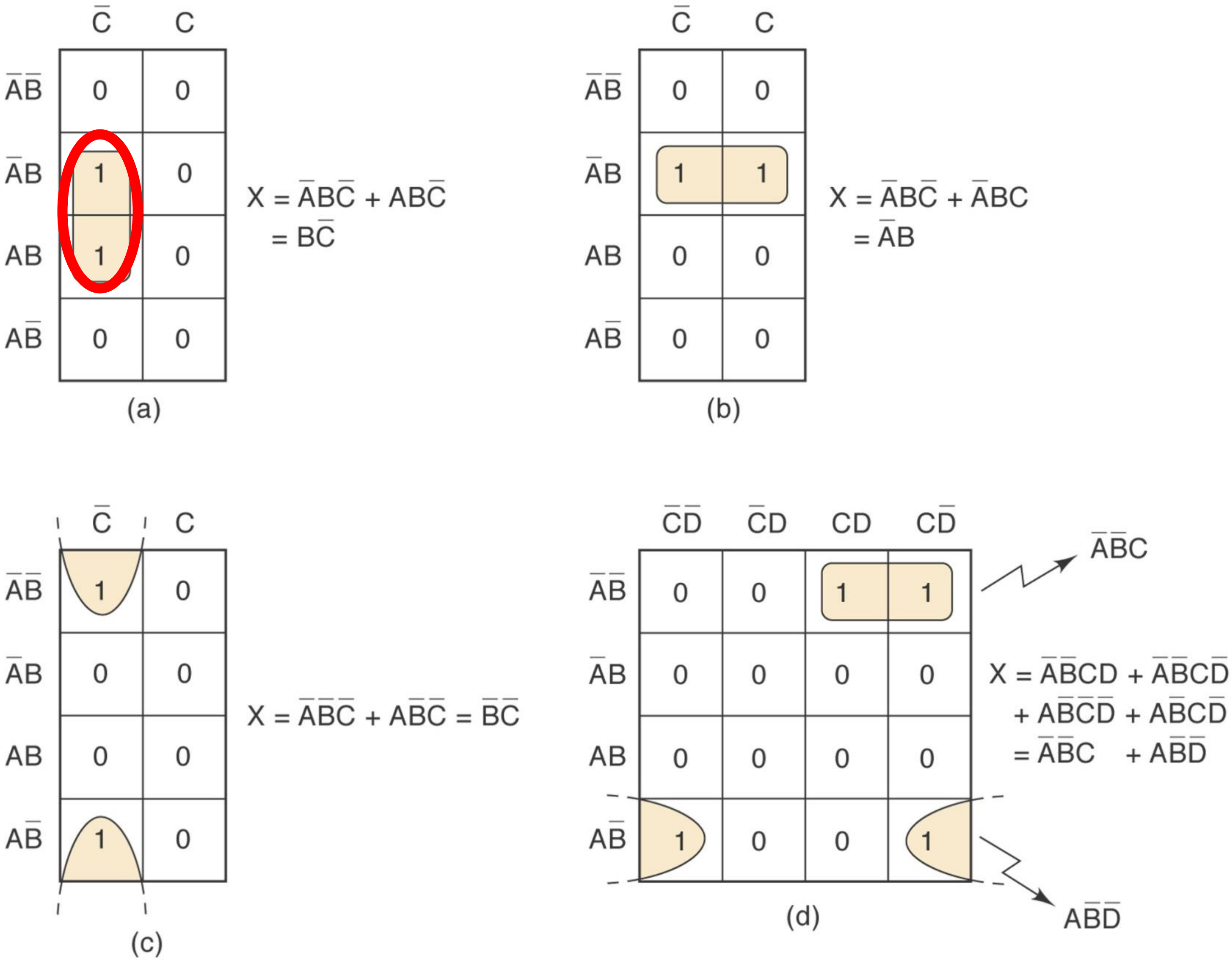
Karnaugh Map Method

- Loop adjacent groups of 2, 4, or 8 that contain 1's will result in further simplification.

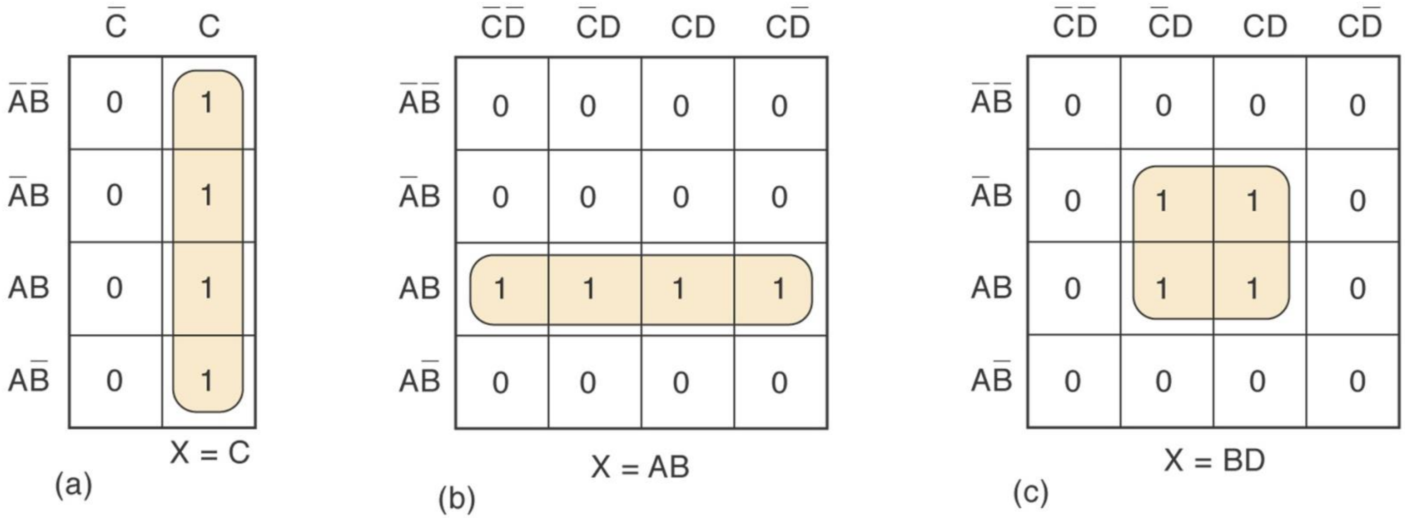
- When the largest possible groups have been looped, only the common terms are placed in the final expression.

- Looping may also be wrapped between top, bottom, and sides.

Looping pairs of adjacent 1's – One variable is eliminated.



Looping groups of four adjacent 1's – Two variables are eliminated.



Complete K-Map simplification process

1. Construct the K map, place 1s as per the truth table.
2. Loop 1s that are not adjacent to any other 1s.
3. Loop 1s that are in pairs *and cannot be looped into quads or octets.*
4. Loop 1s in octets (8) even if they have already been looped.
5. Loop quads (4) that have one or more 1s not already looped.
6. Loop any pairs (2) necessary to include 1s not already looped.
7. Form the OR sum of terms generated by each loop.

Example 1: K-Map simplification

Simplify the following Boolean expression:

$\overline{A}\,\overline{B}C\overline{D} + \overline{A}B\overline{C}D + \overline{A}BCD + AB\overline{C}D + ABCD + A\overline{B}CD$

Example 1: K-Map simplification

Simplify the following Boolean expression:

$$\bar{A}\bar{B}C\bar{D} + \bar{A}B\bar{C}D + \bar{A}BCD + AB\bar{C}D + ABCD + A\bar{B}CD$$

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$				
$\bar{A}B$				
AB				
$A\bar{B}$				

1. Construct the K map, place 1s as per the truth table.
2. Loop 1s that are not adjacent to any other 1s.
3. Loop 1s that are in pairs *and cannot be looped into quads or octets.*
4. Loop 1s in octets (8) *even if they have already been looped.*
5. Loop quads (4) that have one or more 1s not already looped.
6. Loop any pairs (2) necessary to *include 1s not already looped.*
7. Form the OR sum of terms generated by each loop.

Example 1: K-Map simplification

Simplify the following Boolean expression:

$$\bar{A}\bar{B}C\bar{D} + \bar{A}B\bar{C}D + \bar{A}BCD + AB\bar{C}D + ABCD + A\bar{B}CD$$

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$				1
$\bar{A}B$		1	1	
AB		1	1	
$A\bar{B}$			1	

1. Construct the K map, place 1s as per the truth table.
2. Loop 1s that are not adjacent to any other 1s.
3. Loop 1s that are in pairs *and cannot be looped into quads or octets.*
4. Loop 1s in octets (8) *even if they have already been looped.*
5. Loop quads (4) that have one or more 1s not already looped.
6. Loop any pairs (2) necessary to *include 1s not already looped.*
7. Form the OR sum of terms generated by each loop.

Example 1: K-Map simplification

Simplify the following Boolean expression:

$$\bar{A}\bar{B}C\bar{D} + \bar{A}B\bar{C}D + \bar{A}BCD + AB\bar{C}D + ABCD + A\bar{B}CD$$

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$				1
$\bar{A}B$		1	1	
AB		1	1	
$A\bar{B}$			1	

1. Construct the K map, place 1s as per the truth table.
2. Loop 1s that are not adjacent to any other 1s.
3. Loop 1s that are in pairs *and cannot be looped into quads or octets.*
4. Loop 1s in octets (8) *even if they have already been looped.*
5. Loop quads (4) that have one or more 1s not already looped.
6. Loop any pairs (2) necessary to *include 1s not already looped.*
7. Form the OR sum of terms generated by each loop.

Example 1: K-Map simplification

Simplify the following Boolean expression:

$$\bar{A}\bar{B}C\bar{D} + \bar{A}B\bar{C}D + \bar{A}BCD + AB\bar{C}D + ABCD + A\bar{B}CD$$

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$				1
$\bar{A}B$		1	1	
AB		1	1	
$A\bar{B}$			1	

1. Construct the K map, place 1s as per the truth table.
2. Loop 1s that are not adjacent to any other 1s.
3. Loop 1s that are in pairs *and cannot be looped into quads or octets.*
4. Loop 1s in octets (8) *even if they have already been looped.*
5. Loop quads (4) that have one or more 1s not already looped.
6. Loop any pairs (2) necessary to *include 1s not already looped.*
7. Form the OR sum of terms generated by each loop.

Example 1: K-Map simplification

Simplify the following Boolean expression:

$$\bar{A}\bar{B}C\bar{D} + \bar{A}B\bar{C}D + \bar{A}BCD + AB\bar{C}D + ABCD + A\bar{B}CD$$

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$				1
$\bar{A}B$		1	1	
AB		1	1	
$A\bar{B}$			1	

1. Construct the K map, place 1s as per the truth table.
2. Loop 1s that are not adjacent to any other 1s.
3. Loop 1s that are in pairs *and cannot be looped into quads or octets.*
4. Loop 1s in octets (8) *even if they have already been looped. (none here)*
5. Loop quads (4) that have one or more 1s not already looped.
6. Loop any pairs (2) necessary to *include 1s not already looped.*
7. Form the OR sum of terms generated by each loop.

Example (a) K-Map simplification

Simplify the following Boolean expression:

$$\bar{A}\bar{B}C\bar{D} + \bar{A}B\bar{C}D + \bar{A}BCD + AB\bar{C}D + ABCD + A\bar{B}CD$$

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$				1
$\bar{A}B$		1	1	
AB		1	1	
$A\bar{B}$			1	

1. Construct the K map, place 1s as per the truth table.
2. Loop 1s that are not adjacent to any other 1s.
3. Loop 1s that are in pairs *and cannot be looped into quads or octets.*
4. Loop 1s in octets (8) *even if they have already been looped.*
5. Loop quads (4) that have one or more 1s not already looped.
6. Loop any pairs (2) necessary to *include 1s not already looped.*
7. Form the OR sum of terms generated by each loop.

Example 1: K-Map simplification

Simplify the following Boolean expression:

$$\bar{A}\bar{B}C\bar{D} + \bar{A}B\bar{C}D + \bar{A}BCD + AB\bar{C}D + ABCD + A\bar{B}CD$$

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$				1
$\bar{A}B$		1	1	
AB		1	1	
$A\bar{B}$			1	

1. Construct the K map, place 1s as per the truth table.
2. Loop 1s that are not adjacent to any other 1s.
3. Loop 1s that are in pairs *and cannot be looped into quads or octets.*
4. Loop 1s in octets (8) *even if they have already been looped.*
5. Loop quads (4) that have one or more 1s not already looped.
6. Loop any pairs (2) necessary to *include 1s not already looped. (none here)*
7. Form the OR sum of terms generated by each loop.

Example 1: K-Map simplification

Simplify the following Boolean expression:

$$\bar{A}\bar{B}C\bar{D} + \bar{A}B\bar{C}D + \bar{A}BCD + AB\bar{C}D + ABCD + A\bar{B}CD$$

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$				1
$\bar{A}B$		1	1	
AB		1	1	
$A\bar{B}$			1	

1. Construct the K map, place 1s as per the truth table.
2. Loop 1s that are not adjacent to any other 1s.
3. Loop 1s that are in pairs *and cannot be looped into quads or octets.*
4. Loop 1s in octets (8) *even if they have already been looped.*
5. Loop quads (4) that have one or more 1s not already looped.
6. Loop any pairs (2) necessary to *include 1s not already looped.*
7. Form the OR sum of terms generated by each loop.

$$BD + ACD + \bar{A}\bar{B}C\bar{D}$$

Example 2:

Use a K-map to simplify:

$$y = \overline{C}(\overline{A}\,\overline{B}\,\overline{D} + D) + A\overline{B}C + \overline{D}$$

$$y = \overline{C}(\overline{A}\,\overline{B}\,\overline{D} + D) + A\overline{B}C + \overline{D}$$

$$y = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{C}D + A\overline{B}C + \overline{D}$$ = A’B’C’D’+C’D+AB’C+D’

$$y = \overline{C}(\overline{A}\,\overline{B}\,\overline{D} + D) + A\overline{B}C + \overline{D}$$

$$y = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{C}D + A\overline{B}C + \overline{D}$$ = A’B’C’D’+C’D+AB’C+D’

Step 1:

Draw Kmap

CD AB	C’D’	C’D	CD	CD’
A’B’				
A’B				
AB				
AB’				

$$y = \overline{C}(\overline{A}\,\overline{B}\,\overline{D} + D) + A\overline{B}C + \overline{D}$$

$$y = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{C}D + A\overline{B}C + \overline{D}$$ = A’B’C’D’+C’D+AB’C+D’

Step 2:
Insert 1 according to each term in the Boolean expression

CD / AB	**C’D’**	**C’D**	**CD**	**CD’**
A’B’	1			
A’B				
AB				
AB’				

$$y = \overline{C}(\overline{A}\,\overline{B}\,\overline{D} + D) + A\overline{B}C + \overline{D}$$

$$y = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{C}D + A\overline{B}C + \overline{D}$$ = A'B'C'D'+C'D+AB'C+D'

Step 2:
Insert 1 according to each term in the Boolean expression

CD AB	C'D'	C'D	CD	CD'
A'B'	1	1		
A'B		1		
AB		1		
AB'		1		

$$y = \overline{C}(\overline{A}\overline{B}\overline{D} + D) + A\overline{B}C + \overline{D}$$

$$y = \overline{A}\overline{B}\overline{C}\overline{D} + \overline{C}D + A\overline{B}C + \overline{D}$$ = A’B’C’D’+C’D+AB’C+D’

Step 2:
Insert 1 according to each term in the Boolean expression

CD / AB	C’D’	C’D	CD	CD’
A’B’	1	1		
A’B		1		
AB		1		
AB’		1	**1**	**1**

$$y = \overline{C}(\overline{A}\overline{B}\overline{D} + D) + A\overline{B}C + \overline{D}$$

$$y = \overline{A}\overline{B}\overline{C}\overline{D} + \overline{C}D + A\overline{B}C + \overline{D}$$ = A'B'C'D'+C'D+AB'C+D'

Step 2:
Insert 1 according to each term in the Boolean expression

CD / AB	C'D'	C'D	CD	CD'
A'B'	1	1		1
A'B	1	1		1
AB	1	1		1
AB'	1	1	1	1

$$y = \overline{C}(\overline{A}\overline{B}\overline{D} + D) + A\overline{B}C + \overline{D}$$

$$y = \overline{A}\overline{B}\overline{C}\overline{D} + \overline{C}D + A\overline{B}C + \overline{D}$$ = A’B’C’D’+C’D+AB’C+D’

Step 3:
Loop 1s in
a pair,
or in a quad,
or in an octet

CD / AB	C’D’	C’D	CD	CD’
A’B’	1	1		1
A’B	1	1		1
AB	1	1		1
AB’	1	1	1	1

C’

$$y = \overline{C}(\overline{A}\overline{B}\overline{D} + D) + A\overline{B}C + \overline{D}$$

$$y = \overline{A}\overline{B}\overline{C}\overline{D} + \overline{C}D + A\overline{B}C + \overline{D}$$ = A'B'C'D'+C'D+AB'C+D'

<u>Step 3:</u>

Loop 1s in
a pair,
or in a quad,
or in an octet

CD / AB	C'D'	C'D	CD	CD'
A'B'	1	1		1
A'B	1	1		1
AB	1	1		1
AB'	1	1	1	1

C' D'

$$y = \overline{C}(\overline{A}\,\overline{B}\,\overline{D} + D) + A\overline{B}C + \overline{D}$$

$$y = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{C}D + A\overline{B}C + \overline{D}$$ = A’B’C’D’+C’D+AB’C+D’

<u>Step 3:</u>
Loop 1s in
a pair,
or in a quad,
or in an octet

CD / AB	C’D’	C’D	CD	CD’
A’B’	1	1		1
A’B	1	1		1
AB	1	1		1
AB’	1	1	1	1

AB’

C’ D’

$$y = \overline{C}(\overline{A}\overline{B}\overline{D} + D) + A\overline{B}C + \overline{D}$$

$$y = \overline{A}\overline{B}\overline{C}\overline{D} + \overline{C}D + A\overline{B}C + \overline{D}$$ = A'B'C'D'+C'D+AB'C+D'

= AB' + C' + D'

<u>Step 4:</u>
Write the OR sum of terms

CD / AB	C'D'	C'D	CD	CD'
A'B'	1	1		1
A'B	1	1		1
AB	1	1		1
AB'	1	1	1	1

AB'

C'

D'

- Simpify the following expressions
  - using Boolean algebra and
  - using K-maps

$$\text{(a)}\quad x = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}BC + ABC + A\overline{B}\,\overline{C} + A\overline{B}C$$

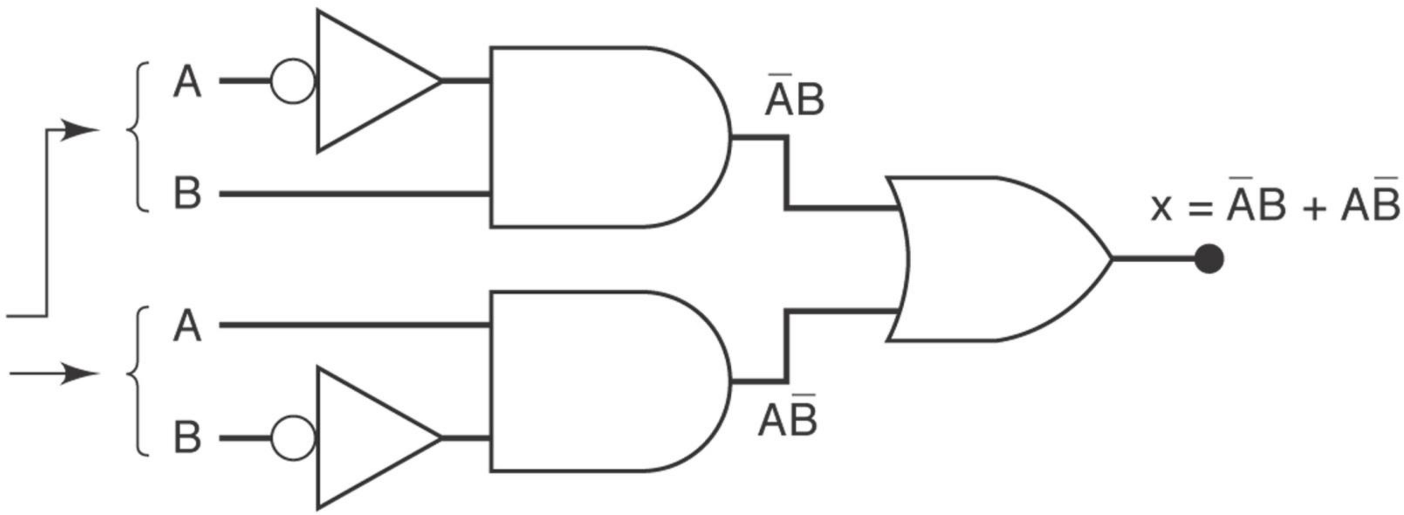
$$\text{(b)}\quad x = \overline{C + D} + \bar{A}C\bar{D} + A\bar{B}\bar{C} + \bar{A}\bar{B}CD + AC\bar{D}$$

Designing Combinational Logic Circuits

If we know the design conditions {(a) truth table} we want to design the logic circuit and then (b) implement the circuit with AND, OR and NOT gates.

A	B	x
0	0	0
0	1	1
1	0	1
1	1	0

(a)



(b)

Designing Combinational Logic Circuits

Design Procedure:

1. Set up truth table
2. Write AND term for each case where the output is HI
3. Write the SOP expression for the output
4. Simplify the expression
5. Implement the circuit

Designing Combinational Logic Circuits

Example 1:

Design a logic circuit that has three inputs, *A*, *B*, and *C*, whose output will be HIGH only when a majority of the inputs are HIGH.

1) Set up truth table

Design a logic circuit that has <u>three inputs</u>, *A*, *B*, and *C*, whose output will be HIGH only when a majority of the inputs are HIGH.

3 columns for the inputs

Output column

$2^3 = 8$ possible combinations

A	B	C	x
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

1) Set up truth table

- Design a logic circuit that has three inputs, *A*, *B*, and *C*, whose output will be HIGH only when a majority of the inputs are HIGH.

A	*B*	*C*	*x*
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

1) Set up truth table

2) Write AND term for each case where the output is HI

A	B	C	x	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\rightarrow \overline{A}BC$
1	0	0	0	
1	0	1	1	$\rightarrow A\overline{B}C$
1	1	0	1	$\rightarrow AB\overline{C}$
1	1	1	1	$\rightarrow ABC$

3) Write the SOP expression for the output

$$x = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

3) Write the SOP expression for the output

4) Simplify the expression – using Boolean Algebra Laws

$$x = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

$$x = \overline{A}BC + ABC + A\overline{B}C + ABC + AB\overline{C} + ABC$$

$$x = BC(\overline{A} + A) + AC(\overline{B} + B) + AB(\overline{C} + C)$$

$$\boxed{x = BC + AC + AB}$$

3) Write the SOP expression for the output

4) Simplify the expression – using K-Map method

$$X = A'BC + AB'C + ABC' + ABC$$

	C'	C
A'B'		
A'B		1
AB	1	1
AB'		1

3) Write the SOP expression for the output

4) Simplify the expression – using K-Map method

$$X = A'BC + AB'C + ABC' + ABC$$

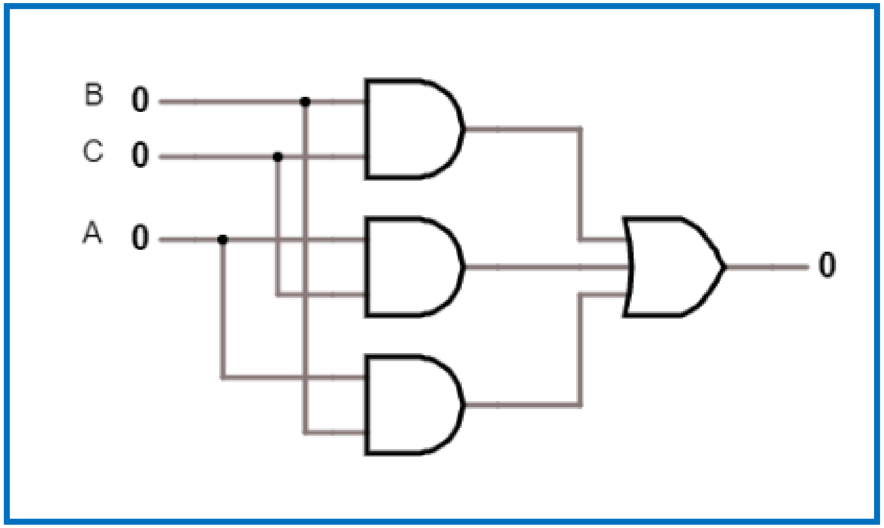
	C'	C
A'B'		
A'B		1
AB	1	1
AB'		1

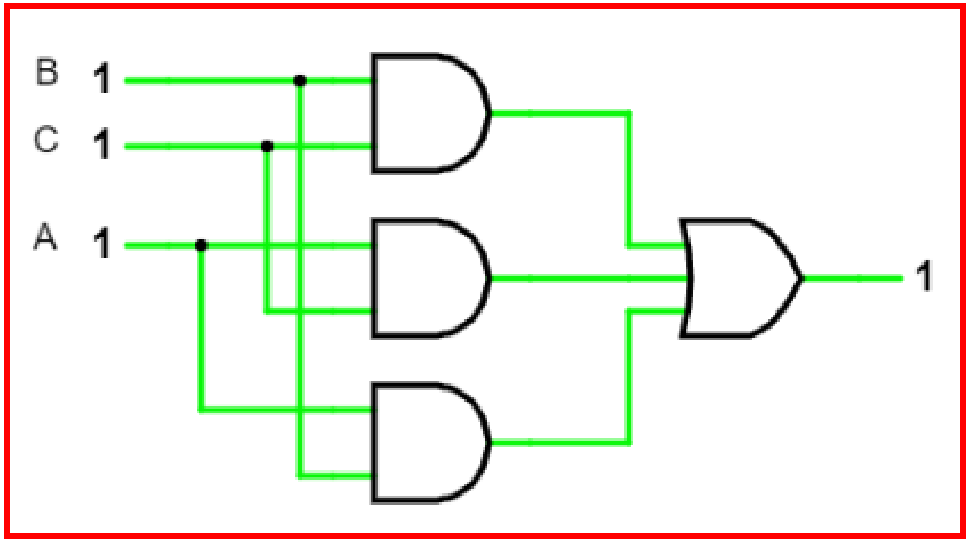
Simplified expression is AB + BC + AC

3) Write the SOP expression for the output

4) Simplified the expression: $x = BC + AC + AB$

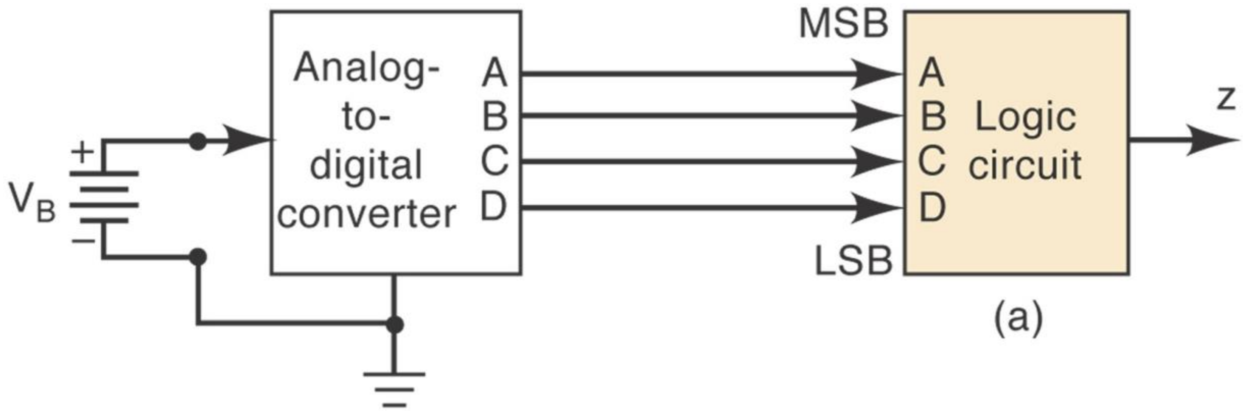
5) Implement circuit:





Example 3:

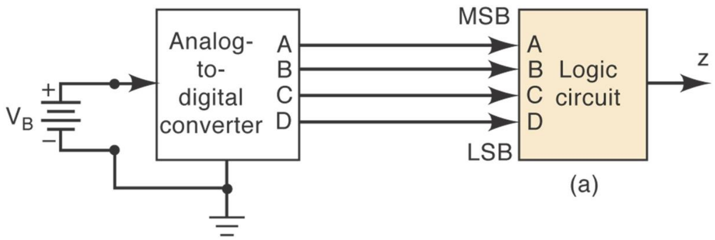
Design a battery monitor that will produce a HI (ie 1) so long as the battery is higher than 6 V (= 0110 in binary) on the output of the ADC (Analog to Digital Converter)



(a)

Example 3: Design a battery monitor that will produce a HI as long as the battery is higher than 6 V = 0110 on the output of the ADC (Analog to Digital Converter)

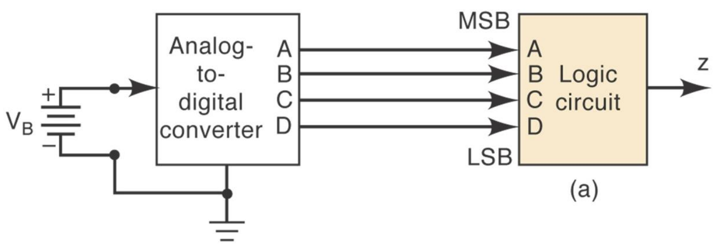
Step 1: Set up the Truth Table



	A	B	C	D	z
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	
10	1	0	1	0	
11	1	0	1	1	
12	1	1	0	0	
13	1	1	0	1	
14	1	1	1	0	
15	1	1	1	1	

Example 3: Design a battery monitor that will produce a HI so long as the battery is higher than 6 V = 0110 on the output of the ADC (Analog to Digital Converter)

Step 1: Set up the Truth Table

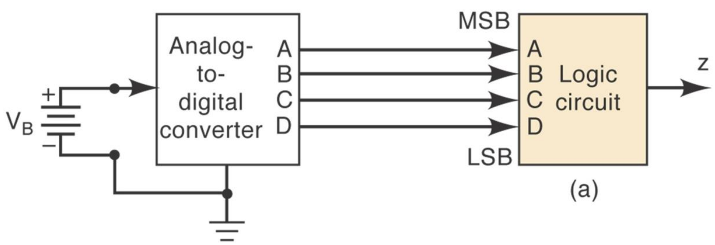


(a)

	A	B	C	D	z
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	1
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	1

Example 3: Design a battery monitor that will produce a HI as long as the battery is higher than 6 V = 0110 on the output of the ADC (Analog to Digital Converter)

Step 1: Set up the Truth Table

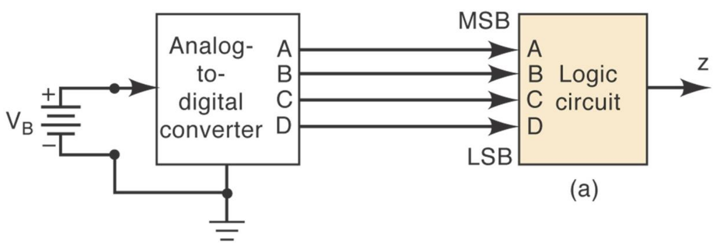


	A	B	C	D	z
0	0	0	0	0	
1	0	0	0	1	
2	0	0	1	0	
3	0	0	1	1	
4	0	1	0	0	
5	0	1	0	1	
6	0	1	1	0	
7	0	1	1	1	1 A'BCD
8	1	0	0	0	1 AB'C'D'
9	1	0	0	1	1 AB'C'D
10	1	0	1	0	1 AB'CD'
11	1	0	1	1	1 AB'CD
12	1	1	0	0	1 ABC'D'
13	1	1	0	1	1 ABC'D
14	1	1	1	0	1 ABCD'
15	1	1	1	1	1 ABCD

Step 2: Write AND term for each case where the output is HI

Example 3: Design a battery monitor that will produce a HI as long as the battery is higher than 6 V = 0110 on the output of the ADC (Analog to Digital Converter)

Step 1: Set up the Truth Table



Step 2: Write AND term for each case where the output is HI

Step 3: Write the SOP for the output

$$z = A'BCD + AB'C'D' + AB'C'D + AB'CD' + AB'CD + ABC'D' + ABC'D + ABCD' + ABCD$$

	A	B	C	D	z	
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	1	A'BCD
8	1	0	0	0	1	AB'C'D'
9	1	0	0	1	1	AB'C'D
10	1	0	1	0	1	AB'CD'
11	1	0	1	1	1	AB'CD
12	1	1	0	0	1	ABC'D'
13	1	1	0	1	1	ABC'D
14	1	1	1	0	1	ABCD'
15	1	1	1	1	1	ABCD

Example 3: Design a battery monitor that will produce a HI as long as the battery is higher than 6 V = 0110 on the output of the ADC (Analog to Digital Converter)

Step 3: Write the SOP for the output

$$z = A'BCD + AB'C'D' + AB'C'D + AB'CD' + AB'CD + ABC'D' + ABC'D + ABCD' + ABCD$$

Step 4: Simplify the expression

	C'D'	C'D	CD	CD'
A'B'				
A'B			1	
AB				
AB'				

	A	B	C	D	z	
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	1	A'BCD
8	1	0	0	0	1	AB'C'D'
9	1	0	0	1	1	AB'C'D
10	1	0	1	0	1	AB'CD'
11	1	0	1	1	1	AB'CD
12	1	1	0	0	1	ABC'D'
13	1	1	0	1	1	ABC'D
14	1	1	1	0	1	ABCD'
15	1	1	1	1	1	ABCD

Example 3: Design a battery monitor that will produce a HI as long as the battery is higher than 6 V = 0110 on the output of the ADC (Analog to Digital Converter)

Step 4: Simplify the expression

	C'D'	C'D	CD	CD'
A'B'				
A'B			1	
AB				
AB'	1			

A'BCD
AB'C'D'
AB'C'D
AB'CD'
AB'CD
ABC'D'
ABC'D
ABCD'
ABCD

Example 3: Design a battery monitor that will produce a HI as long as the battery is higher than 6 V = 0110 on the output of the ADC (Analog to Digital Converter)

Step 4: Simplify the expression

	C'D'	C'D	CD	CD'
A'B'				
A'B			1	
AB				
AB'	1	1		

A'BCD
AB'C'D'
AB'C'D
AB'CD'
AB'CD
ABC'D'
ABC'D
ABCD'
ABCD

Example 3: Design a battery monitor that will produce a HI as long as the battery is higher than 6 V = 0110 on the output of the ADC (Analog to Digital Converter)

Step 4: Simplify the expression

	C'D'	C'D	CD	CD'
A'B'				
A'B			1	
AB				
AB'	1	1	1	

A'BCD
AB'C'D'
AB'C'D
AB'CD'
AB'CD
ABC'D'
ABC'D
ABCD'
ABCD

Example 3: Design a battery monitor that will produce a HI as long as the battery is higher than 6 V = 0110 on the output of the ADC (Analog to Digital Converter)

Step 4: Simplify the expression

	C'D'	C'D	CD	CD'
A'B'				
A'B			1	
AB	1	1	1	1
AB'	1	1	1	1

A'BCD
AB'C'D'
AB'C'D
AB'CD'
AB'CD
ABC'D'
ABC'D
ABCD'
ABCD

Example 3: Design a battery monitor that will produce a HI as long as the battery is higher than 6 V = 0110 on the output of the ADC (Analog to Digital Converter)

Step 4: Simplify the expression

	C'D'	C'D	CD	CD'
A'B'				
A'B			1	
AB	1	1	1	1
AB'	1	1	1	1

BCD

A

A'BCD
AB'C'D'
AB'C'D
AB'CD'
AB'CD
ABC'D'
ABC'D
ABCD'
ABCD

Simplified expression is BCD + A

Example 3: Design a battery monitor that will produce a HI as long as the battery is higher than 6 V = 0110 on the output of the ADC (Analog to Digital Converter)

Step 4: Simplify the expression

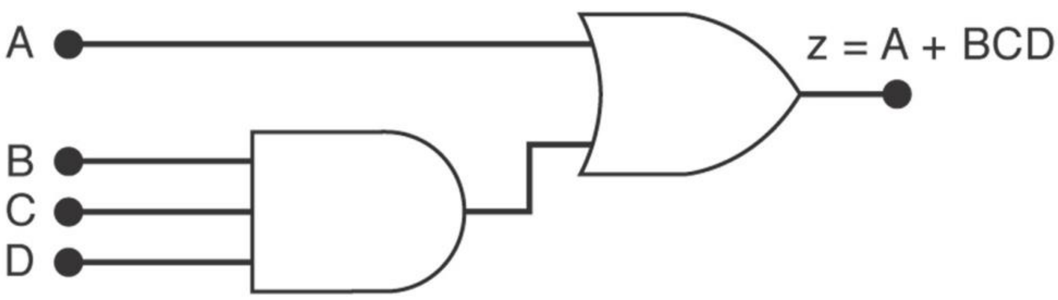
	C'D'	C'D	CD	CD'
A'B'				
A'B			1	
AB	1	1	1	1
AB'	1	1	1	1

BCD

A

A'BCD
AB'C'D'
AB'C'D
AB'CD'
AB'CD
ABC'D'
ABC'D
ABCD'
ABCD

Step 5: Implement the logic circuit



Don’t care Output Conditions

In some cases, we may encounter output states that are ‘impossible’ – that is the corresponding input combination is not possible in practice

Eg: A digital display (0-9) that is driven by a binary input (4 bits): binary values 1010 -> 1111 never occur.

Don’t care Output Conditions

Can be changed 0/1 so that the simplest expression can be obtained from the K-map. Typically occur when we know certain input conditions are impossible.

A	B	C	z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	x
1	0	0	x
1	0	1	1
1	1	0	1
1	1	1	1

"don't care"

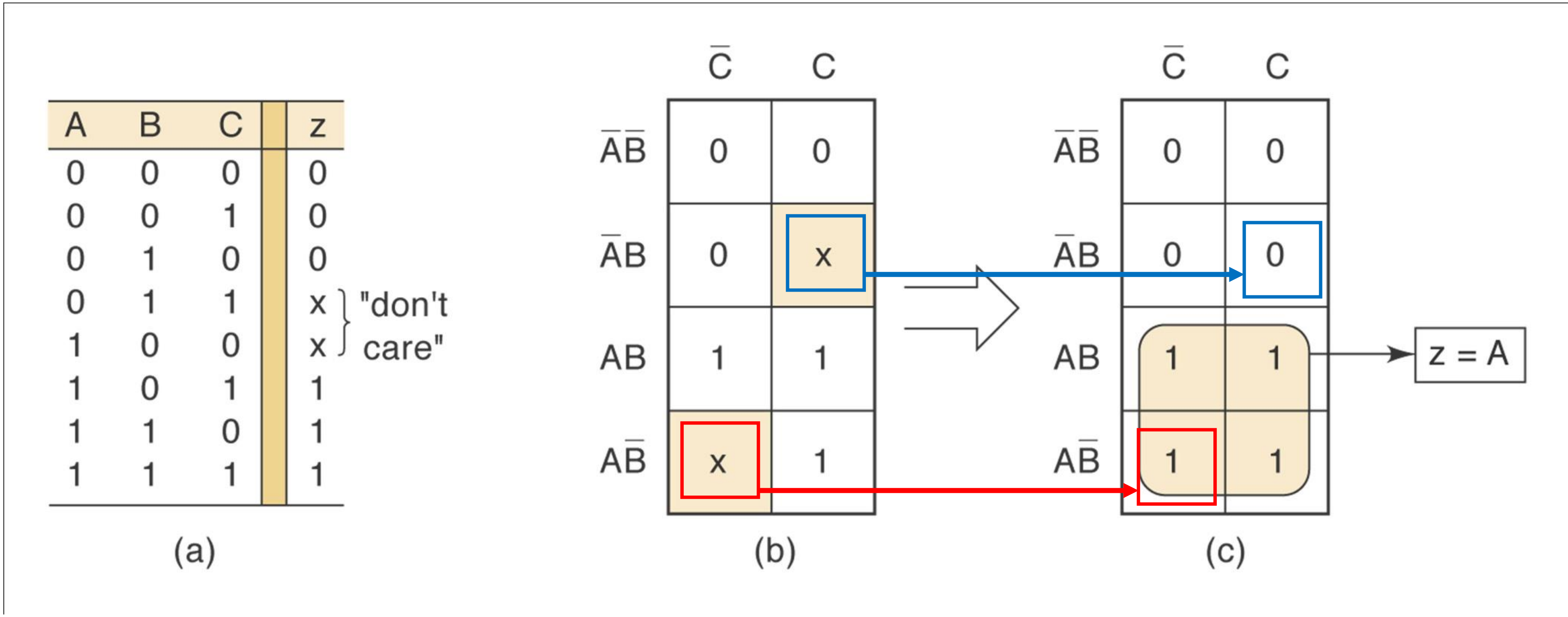
(a)

	$\overline{C}$	C
$\overline{A}\,\overline{B}$	0	0
$\overline{A}B$	0	x
AB	1	1
$A\overline{B}$	x	1

(b)

Don’t care Output Conditions

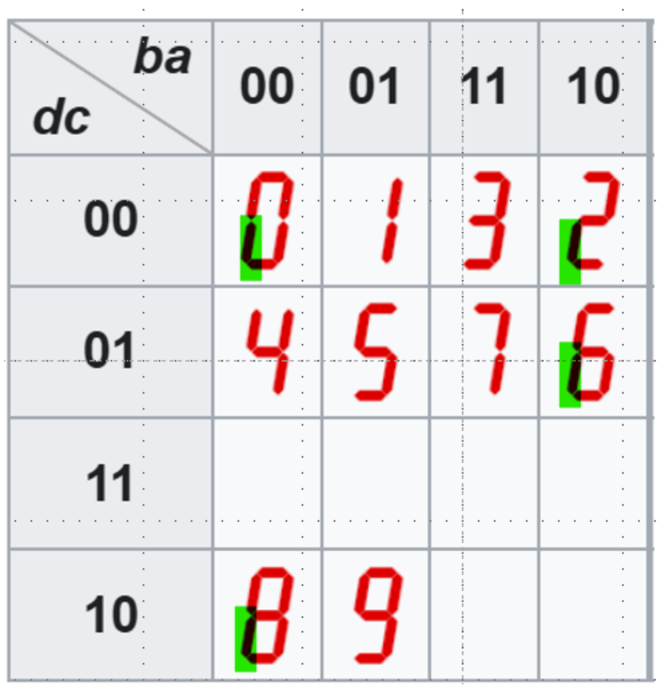
Can be changed 0/1 so that the simplest expression can be obtained from the K-map. Typically occur when we know certain input conditions are impossible.

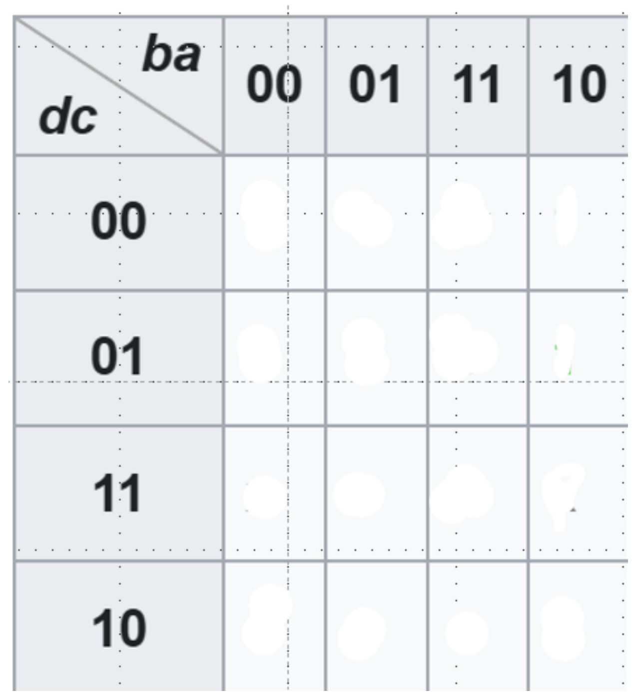


(a) (b) (c)

Example: Binary Coded Decimal

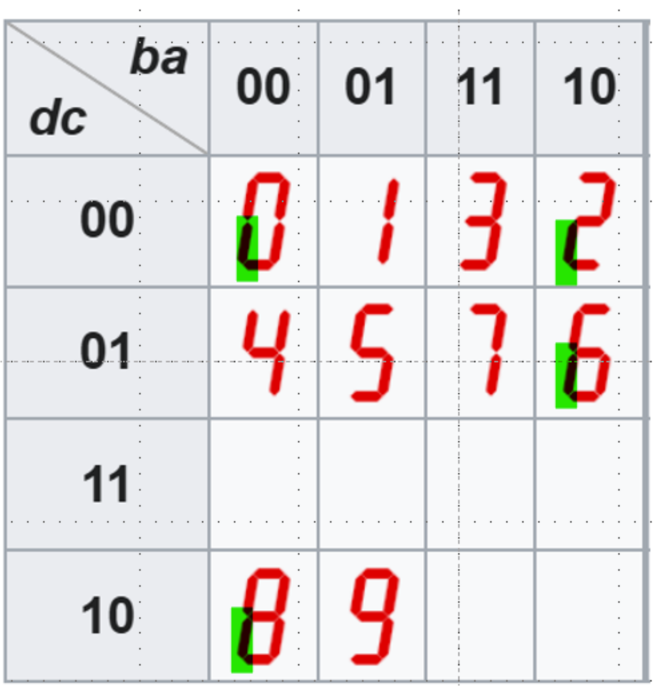
Consider again the 4-bit binary number abcd, driving a 7-segment LED display. Simplify the expression that will light up the bottom left led on a 7-segment display.





Example: Binary Coded Decimal

Consider again the 4-bit binary number abcd, driving a 7-segment LED display. Simplify the expression that will light up the bottom left led on a 7-segment display.

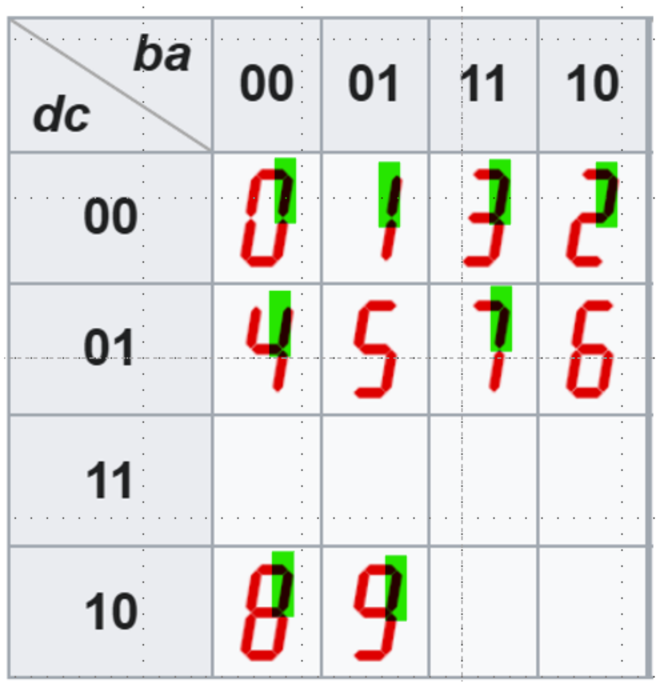


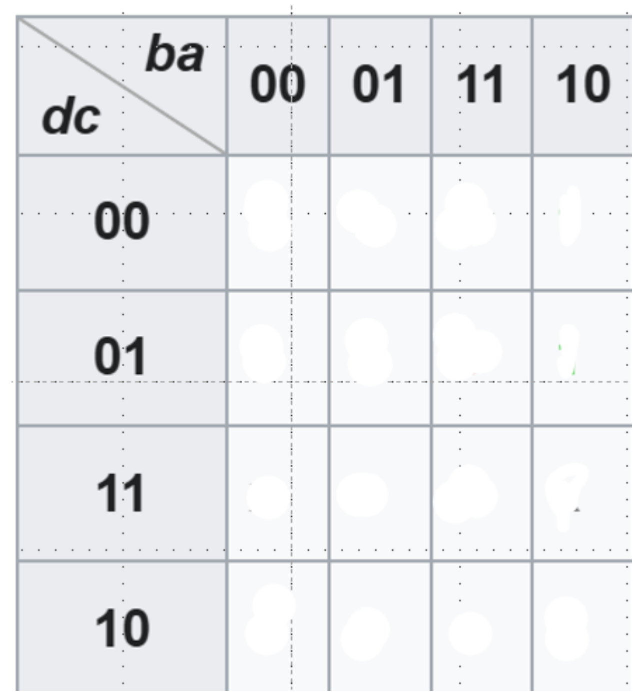
dc \ ba	00	01	11	10
00	1	0	0	1
01	0	0	0	1
11	x	x	x	x
10	1	0	x	x

dc \ ba	00	01	11	10
00	1	0	0	1
01	0	0	0	1
11	0	0	0	1
10	1	0	0	1

Exercise: Binary Coded Decimal

Consider again the 4-bit binary number abcd, driving a 7-segment LED display. Simplify the expression that will light up the top right led on a 7-segment display.





Determine the minimum expression for each of the K maps shown below.

	$\overline{C}\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\overline{B}$	1	1	1	1
$\overline{A}B$	1	1	0	0
AB	0	0	0	1
$A\overline{B}$	0	0	1	1

(a)

	$\overline{C}\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\overline{B}$	1	0	1	1
$\overline{A}B$	1	0	0	1
AB	0	0	0	0
$A\overline{B}$	1	0	1	1

(b)

	$\overline{C}$	C
$\overline{A}\overline{B}$	0	1
$\overline{A}B$	0	0
AB	1	0
$A\overline{B}$	1	X

(c)

Determine the input conditions needed to produce a HI output in the circuit below