Chapter 2 – Combinational Digital Circuits

Part 2 – Circuit Optimization

Overview

- **Part 1 – Gate Circuits and Boolean Equations**
  - **Binary Logic and Gates**
  - **Boolean Algebra**
  - **Standard Forms**
- **Part 2 – Circuit Optimization**
  - **Two-Level Optimization**
  - **Map Manipulation**
  - **Practical Optimization (Espresso)**
  - **Multi-Level Circuit Optimization**
- **Part 3 – Additional Gates and Circuits**
  - **Other Gate Types**
  - **Exclusive-OR Operator and Gates**
  - **High-Impedance Outputs**

Circuit Optimization

- Goal: To obtain the simplest implementation for a given function
- Optimization is a more formal approach to simplification that is performed using a specific procedure or algorithm
- Optimization requires a cost criterion to measure the simplicity of a circuit
- Distinct cost criteria we will use:
  - Literal cost (L)
  - Gate input cost (G)
  - Gate input cost with NOTs (GN)

Literal Cost

- Literal – a variable or its complement
- Literal cost – the number of literal appearances in a Boolean expression corresponding to the logic circuit diagram
- Examples:
  - $F = BD + A\overline{B}C + A\overline{C}\,\overline{D}$ $L = 8$
  - $F = BD + A\overline{B}C + A\overline{B}\,\overline{D} + AB\overline{C}$ $L =$
  - $F = (A + B)(A + D)(B + C + \overline{D})(\overline{B} + \overline{C} + D)$ $L =$
  - Which solution is the best?

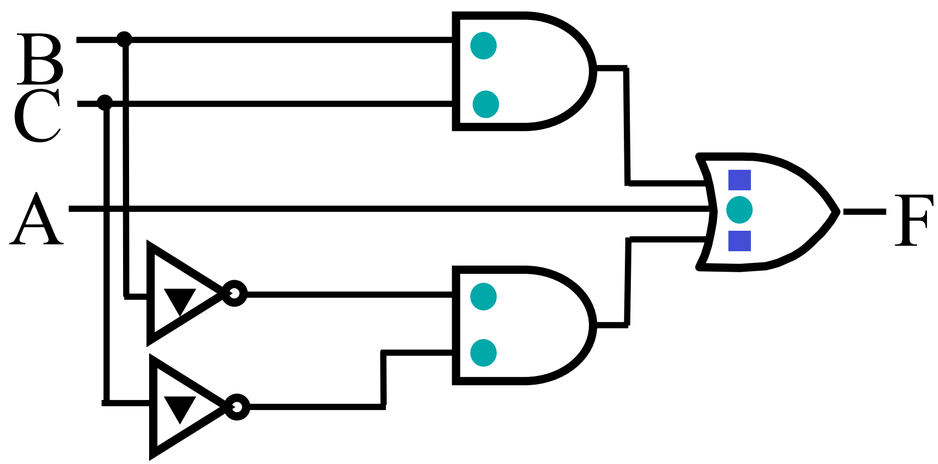
Gate Input Cost

- **Gate input costs - the number of inputs to the gates in the implementation corresponding exactly to the given equation or equations.** (G - inverters not counted, GN - inverters counted)
- **For SOP and POS equations, it can be found from the equation(s) by finding the sum of:**
  - all literal appearances
  - the number of terms excluding single literal terms,(G) and
  - optionally, the number of distinct complemented single literals (GN).
- **Example:**
  - $F = BD + A\overline{B}C + A\overline{C}\,\overline{D}$ G = 8, GN = 11
  - $F = BD + A\overline{B}C + A\overline{B}\,\overline{D} + AB\overline{C}$ G = , GN =
  - $F = (A + \overline{B})(A + D)(B + C + \overline{D})(\overline{B} + \overline{C} + D)$ G = , GN =
  - Which solution is the best?

Cost Criteria (continued)

- Example 1: $GN = G + 2 = 9$
- $F = A + B\,C + \overline{B}\,\overline{C}$ $L = 5$

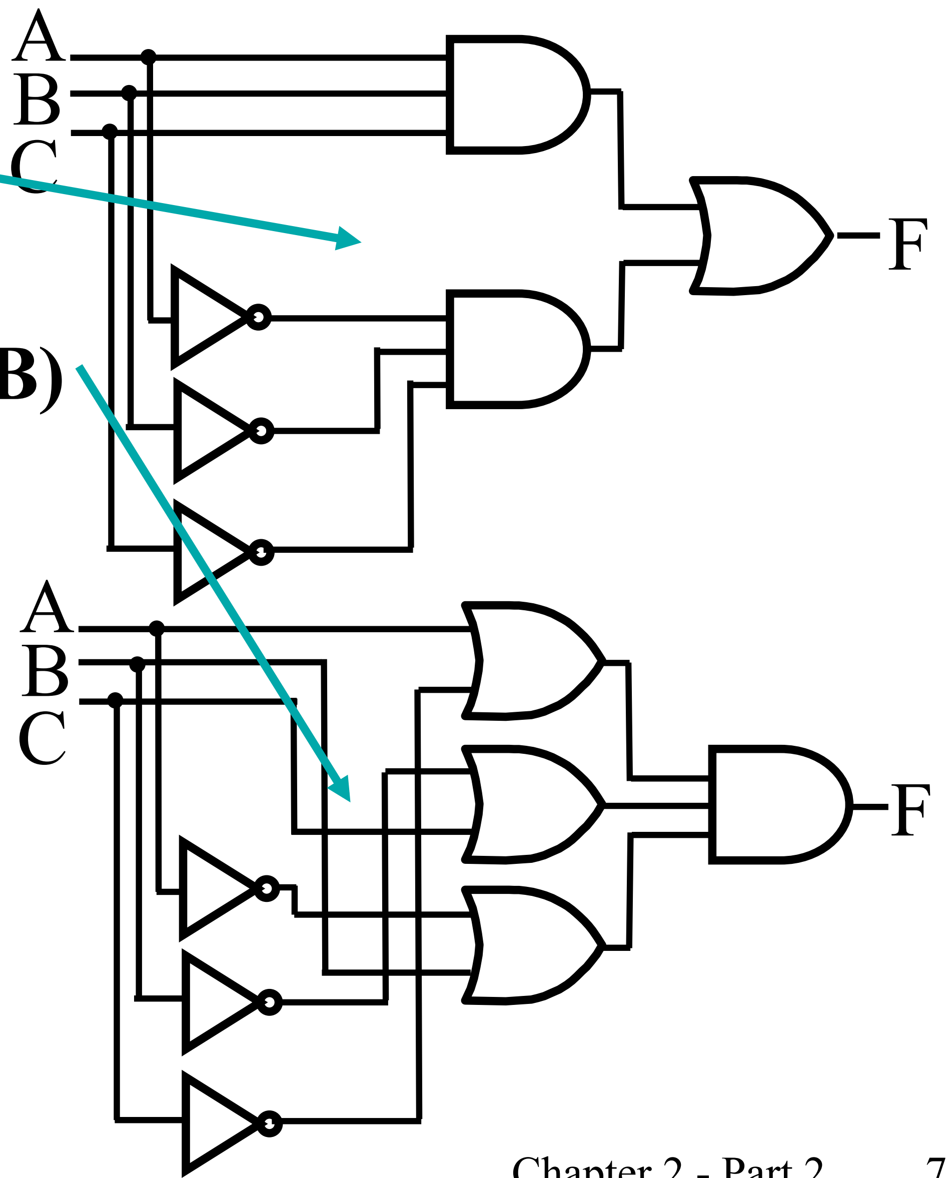
$G = L + 2 = 7$



- L (literal count) counts the AND inputs and the single literal OR input.
- G (gate input count) adds the remaining OR gate inputs
- GN(gate input count with NOTs) adds the inverter inputs

Cost Criteria (continued)

- Example 2:
- $F = A\,B\,C + \bar{A}\,\bar{B}\,\bar{C}$
- $L = 6 \quad G = 8 \; GN = 11$
- $F = (A + \bar{C})(\bar{B} + C)(\bar{A} + B)$
- $L = 6 \quad G = 9 \; GN = 12$
- <u>Same</u> function and <u>same</u> literal cost
- But first circuit has <u>better</u> gate input count and <u>better</u> gate input count with NOTs
- Select it!



Boolean Function Optimization

- **Minimizing the gate input (or literal) cost of a (a set of) Boolean equation(s) reduces circuit cost.**
- **We choose gate input cost.**
- **Boolean Algebra and graphical techniques are tools to minimize cost criteria values.**
- **Some important questions:**
  - **When do we stop trying to reduce the cost?**
  - **Do we know when we have a minimum cost?**
- **Methods**
  - Quine-McCluskey
  - Karnaugh (K-) diyagramı
  - Espresso

Quine-McCluskey Method

- The Boolean function is represented in SOP or POS.
- 1st Phase: Finding the prime implicants of f.
- 2nd Phase: Finding optimal representation of f by taking some of the prime implicants.

Optimization Example

$$f(x_1,x_2,x_3,x_4)=\sum_m(1,3,5,6,7,8,12,14,15)$$

$$f(x_1,x_2,x_3,x_4)=x_1'x_2'x_3'x_4+x_1'x_2'x_3x_4+ x_1'x_2x_3'x_4+x_1'x_2x_3x_4'+x_1'x_2x_3x_4+x_1x_2'x_3'x_4' +x_1x_2x_3'x_4+x_1x_2x_3x_4'+x_1x_2x_3x_4$$

$$f(x_1,x_2,x_3,x_4)= x_1'x_2'x_4(x_3'+x_3)+x_1'x_3'x_4(x_2'+x_2)+x_1'x_3x_4(x_2'+x_2) +x_1'x_2x_4(x_3'+x_3)+x_1'x_2x_3(x_4'+x_4)+x_2x_3x_4'(x_1'+x_1) +x_2x_3x_4(x_1'+x_1)+x_1x_3'x_4'(x_2'+x_2)+x_1x_2x_4'(x_3'+x_3)+x_1x_2x_3(x_4'+x_4)$$

$$f(x_1,x_2,x_3,x_4)=x_1'x_2'x_4+x_1'x_3'x_4+x_1'x_3x_4+x_1'x_2x_4+x_1'x_2x_3+x_2x_3x_4' +x_2x_3x_4+x_1x_3'x_4'+x_1x_2x_4'+x_1x_2x_3$$

$$f(x_1,x_2,x_3,x_4)= x_1'x_4+x_1'x_4+x_2x_3+x_2x_3+x_1x_3'x_4'+x_1x_2x_4'$$

$$f(x_1,x_2,x_3,x_4)= x_1'x_4+x_2x_3+x_1x_3'x_4'+x_1x_2x_4'$$

Optimization of SOP Representation By Quine-McCluskey Method – 1st Phase : Finding the prime implicants

- Eaxmple: $f(x_1,x_2,x_3,x_4)=\sum_m(1,3,5,6,7,8,12,14,15)$

		x_1	x_2	x_3	x_4
√	1	0	0	0	1
√	8	1	0	0	0
√	3	0	0	1	1
√	5	0	1	0	1
√	6	0	1	1	0
√	12	1	1	0	0
√	7	0	1	1	1
√	14	1	1	1	0
√	15	1	1	1	1

		x_1	x_2	x_3	x_4
1,3	√	0	0	–	1
1,5	√	0	–	0	1
8,12	A	1	–	0	0
3,7	√	0	–	1	1
5,7	√	0	1	–	1
6,7	√	0	1	1	–
6,14	√	–	1	1	0
12,14	B	1	1	–	0
7,15	√	–	1	1	1
14,15	√	1	1	1	–

	x_1	x_2	x_3	x_4	
1,3,5,7	0	–	–	1	C
1,5,3,7	0	–	–	1	
6,7,14,15	–	1	1	–	D
6,14,7,15	–	1	1	–	

Sum of Prime Implicants:

$$f(x_1,x_2,x_3,x_4)=x_1x_3'x_4'+x_1x_2x_4'+x_1'x_4+x_2x_3$$

Optimization of SOP Representation By Quine-McCluskey Method – 2nd Phase : Finding optimal representation

Coverage Table Method

A 8,12

B 12,14

C 1,3,5,7

D 6,7,14,15

	1	3	5	6	7	8	12	14	15	
A						X	X			EPI
B							X	X		
C	X	X	X		X					EPI
D				X	X			X	X	EPI

Optimal Representation:

$$f(x_1,x_2,x_3,x_4)=x_1x_3'x_4'+x_1'x_4+x_2x_3$$

Optimization of SOP Representation By Quine-McCluskey Method – 2nd Phase : Finding optimal representation

Patrick Method

A	8,12
B	12,14
C	1,3,5,7
D	6,7,14,15

P=C C C D (C+D) A (A+B) (B+D) D

C=1, D=1, A=1

P=1

Optimal Representation:

$f(x_1,x_2,x_3,x_4)=x_1x_3'x_4'+x_1'x_4+x_2x_3$

Example: $f(x_1,x_2,x_3,x_4)=\sum_m(1,4,5,7,8,9,11,12,14,15)$

		x_1	x_2	x_3	x_4
√	1	0	0	0	1
√	4	0	1	0	0
√	8	1	0	0	0
√	5	0	1	0	1
√	9	1	0	0	1
√	12	1	1	0	0
√	7	0	1	1	1
√	11	1	0	1	1
√	14	1	1	1	0
√	15	1	1	1	1

	x_1	x_2	x_3	x_4	
1,5	0	–	0	1	A
1,9	–	0	0	1	B
4,5	0	1	0	–	C
4,12	–	1	0	0	D
8,9	1	0	0	–	E
8,12	1	–	0	0	F
5,7	0	1	–	1	G
9,11	1	0	–	1	H
12,14	1	1	–	0	I
7,15	–	1	1	1	J
11,15	1	–	1	1	K
14,15	1	1	1	–	L

Sum of Prime Implicants:

$$f(x_1,x_2,x_3,x_4)=x_1x_3'x_4+x_2'x_3'x_4+x_1'x_2x_3'+x_2x_3'x_4'+x_1x_2'x_3'$$
$$+x_1x_3'x_4'+x_1'x_2x_4+x_1x_2'x_4+x_1x_2x_4'+x_2x_3x_4+x_1x_3x_4+x_1x_2x_3$$

Example: $f(x_1,x_2,x_3,x_4)=\sum_m(1,4,5,7,8,9,11,12,14,15)$

	1	4	5	7	8	9	11	12	14	15
A	X		X							
B	X					X				
C		X	X							
D		X						X		
E					X	X				
F					X			X		
G			X	X						
H						X	X			
I								X	X	
J				X						X
K							X			X
L									X	X

P=(A+B)(C+D)(A+C+G)(G+J)(E+F)

(B+E+H)(H+K)(D+F+I)(I+L)(J+K+L)

There is no essential prime implicant.

None of the rows cover another row.

None of the columns cover another column.

Let A=1

P= (C+D)(G+J)(E+F)(B+E+H)(H+K)

(D+F+I)(I+L)(J+K+L)

There is no essential prime implicant.

E covers B. D covers C. J covers G.

B=0, C=0, G=0

P= D J (E+F)(E+H)(H+K)(D+F+I)(I+L)

(J+K+L)

D and J are essential prime implicants.

D=1, J=1

P= (E+F)(E+H)(H+K)(I+L)

There is no essential prime implicant.

E covers F. H covers K. F=0, K=0

P= E (E+H) H (I+L)

E and H essential prime implicant.

E=1, H=1

There is no essential prime implicant.

No row or column coverage.

I=1 or L=1

Example: $f(x_1,x_2,x_3,x_4)=\sum_m(1,4,5,7,8,9,11,12,14,15)$

Optimal Representation:

$f(x_1,x_2,x_3,x_4)=x_1x_3'x_4+x_2x_3'x_4'+x_2x_3x_4+x_1x_2'x_3'+x_1x_2'x_4+x_1x_2x_4'$
or $x_1x_2x_3$

Optimization of POS Representation By Quine-McCluskey Method

- Example: $f(x_1,x_2,x_3,x_4)=\Pi_M(1,4,5,8,11,12,14)$

		x_1	x_2	x_3	x_4
√	1	0	0	0	1
√	4	0	1	0	0
√	8	1	0	0	0
√	5	0	1	0	1
√	12	1	1	0	0
A	11	1	0	1	1
√	14	1	1	1	0

	x_1	x_2	x_3	x_4	
1,5	0	–	0	1	B
4,5	0	1	0	–	C
4,12	–	1	0	0	D
8,12	1	–	0	0	E
12,14	1	1	–	0	F

	1	4	8	5	12	11	14	
A						X		EPI
B	X			X				EPI
C		X		X				
D		X			X			
E			X		X			EPI
F					X		X	EPI

Product of Prime Implicants:

$f(x_1,x_2,x_3,x_4)=(x_1'+x_2+x_3'+x_4')$

$(x_1+x_3+x_4')(x_1+x_2'+x_3)(x_2'+x_3+x_4)$

$(x_1'+x_3+x_4)(x_1'+x_2'+x_4)$

Optimal Representation:

$f(x_1,x_2,x_3,x_4)=(x_1'+x_2+x_3'+x_4')(x_1+x_3+x_4')$

$(x_1'+x_3+x_4)(x_1'+x_2'+x_4)$

$(x_1+x_2'+x_3)$ or $(x_2'+x_3+x_4)$

Optimization of SOP Representation By Quine–McCluskey Method with Don't Care Conditions

- Example: $f(x_1,x_2,x_3)=\Sigma_m(0,1,5)+d\Sigma_m(2,6)$

		x_1	x_2	x_3
√	0	0	0	0
√	1	0	0	1
√	2	0	1	0
√	5	1	0	1
√	6	1	1	0

	x_1	x_2	x_3	
0,1	0	0	–	A
0,2	0	–	0	B
1,5	–	0	1	C
2,6	–	1	0	D

	0	1	5	
A	X	X		
B	X			
C		X	X	EPI

Sum of Prime Implicants:

$f(x_1,x_2,x_3)=x_1'x_2'+x_1'x_3'+x_2'x_3+x_2x_3'$

Optimal Representation:

$f(x_1,x_2,x_3)=x_2'x_3+x_1'x_2'$ veya $x_1'x_3'$

Karnaugh Maps (K-map)

- **A K-map is a collection of squares**
  - **Each square represents a minterm**
  - **The collection of squares is a graphical representation of a Boolean function**
  - **Adjacent squares differ in the value of one variable**
  - **Alternative algebraic expressions for the same function are derived by recognizing patterns of squares**
- **The K-map can be viewed as**
  - **A reorganized version of the truth table**
  - **A topologically-warped Venn diagram as used to visualize sets in algebra of sets**

Some Uses of K-Maps

- Provide a means for:
  - Finding optimum or near optimum
    - SOP and POS standard forms, and
    - two-level AND/OR and OR/AND circuit implementations

    for functions with small numbers of variables
  - Visualizing concepts related to manipulating Boolean expressions, and
  - Demonstrating concepts used by computer-aided design programs to simplify large circuits

Two Variable Karnaugh Map

- Two variable: x ve y
  - 4 minterms:
    - $m_0 = x'y'$ → 00
    - $m_1 = x'y$ → 01
    - $m_2 = xy'$ → 10
    - $m_3 = xy$ → 11

x \ y	0	1
0	m_0	m_1
1	m_2	m_3

x \ y	0	1
0	$x'y'$	$x'y$
1	xy'	xy

K-Map and Truth Tables

- The K-Map is just a different form of the truth table.
- Example – Two variable function:
  - We choose a,b,c and d from the set {0,1} to implement a particular function, F(x,y).

Function Table

Input Values (x,y)	Function Value F(x,y)
0 0	a
0 1	b
1 0	c
1 1	d

K-Map

	y = 0	y = 1
x = 0	a	b
x = 1	c	d

K-Map Function Representation

- **Example:** F(x,y) = x

x \ y	0	1
0	0	0
1	1	1

- **For function F(x,y), the two adjacent cells containing 1's can be combined using the Minimization Theorem:**

$$F(x, y) = x\,\overline{y} + x\,y = x$$

K-Map Function Representation

- **Example:** G(x,y) = x + y

x \ y	0	1
0	0	1
1	1	1

- **For G(x,y), two pairs of adjacent cells containing 1's can be combined using the Minimization Theorem:**

$$G(x,y) = \left(x\,\overline{y} + x\,y\right) + \left(xy + \overline{x}\,y\right) = x + y$$

Duplicate x y

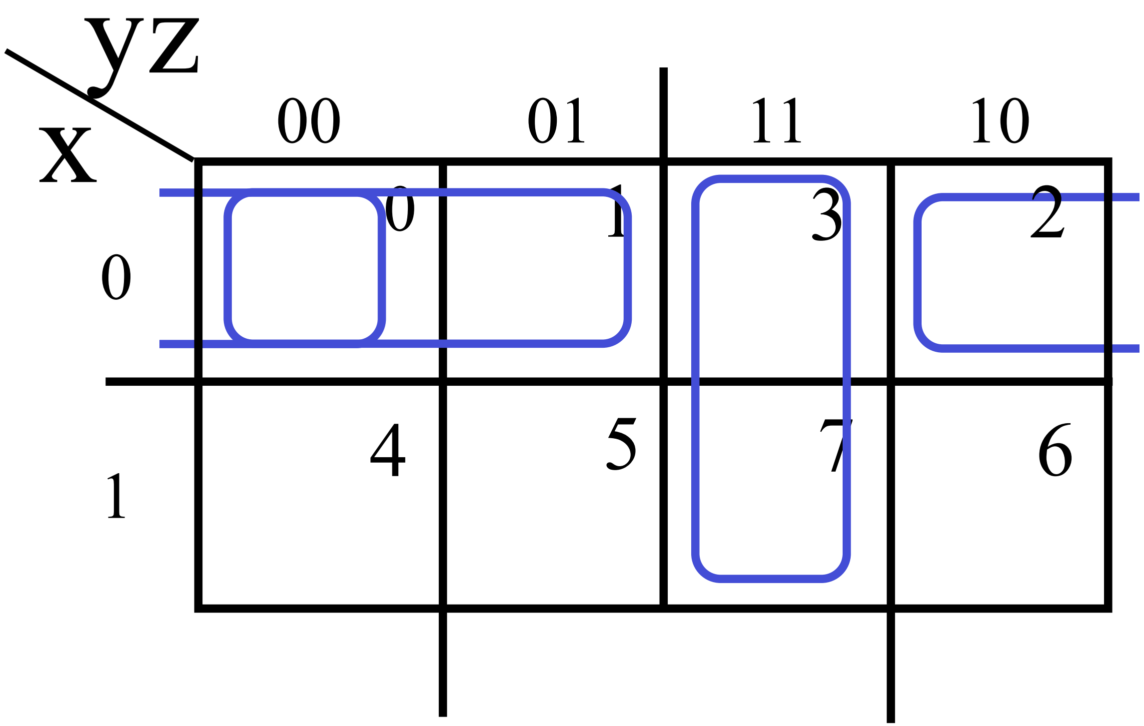
Three Variable Karnaugh Map

x \ yz	00	01	11	10
0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	m_6

- **Note that if the binary value for an index differs in one bit position, the minterms are adjacent on the K-Map**
  - $m_2 \leftrightarrow m_6$, $m_3 \leftrightarrow m_7$
  - $m_2 \leftrightarrow m_0$, $m_6 \leftrightarrow m_4$

Three Variable Karnaugh Map

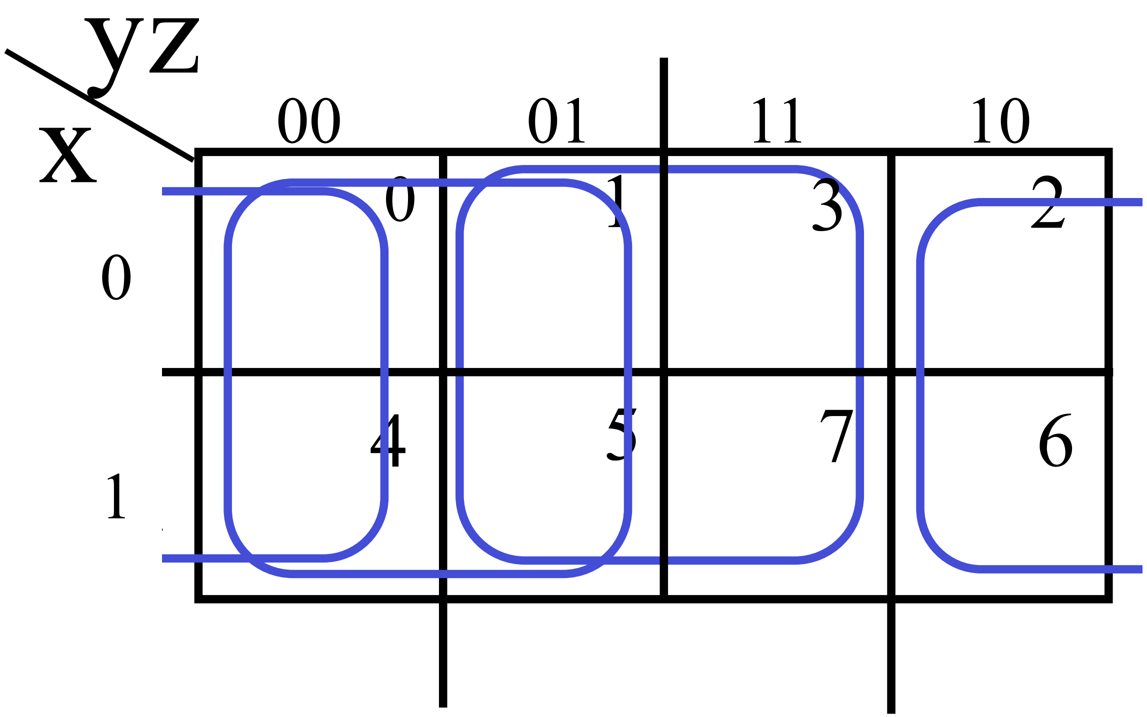
- Rectengular examples with 2 cells



- Read the terms shown by the rectengulars.

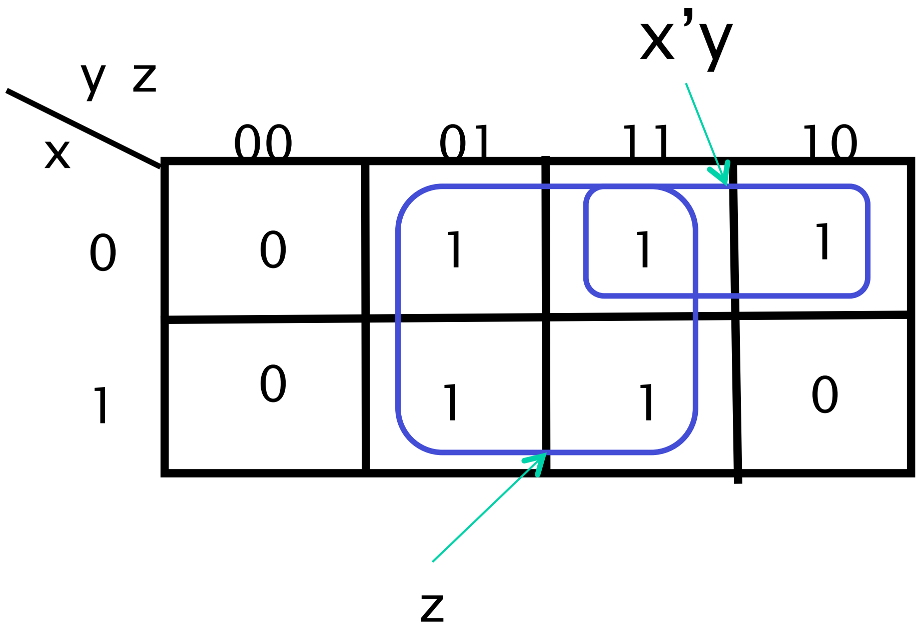
Three Variable Karnaugh Map

- Rectengular examples with 4 cells



- Read the terms shown by the rectengulars.

Example: $F(x,y,z)=\sum_m(1,2,3,5,7)$



1. sütun	2. sütun	3. sütun
x y z	x y z	x y z
√ 0 0 1	√ 0 – 1	B – – 1
√ 0 1 0	√ – 0 1	
√ 0 1 1	A 0 1 –	
√ 1 0 1	√ – 1 1	
√ 1 1 1	√ 1 – 1	

$F=x'y+z$

$P=AB$

$A=B=1$

$A=x'y$

$B=z$

$F=A+B=x'y+z$

Example: Three Variable Karnaugh Map

- $F_1(x, y, z) = \Sigma\,(2, 3, 4, 5)$

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

- $F_1(x, y, z) = xy' + x'y$
- $F_2(x, y, z) = \Sigma\,(3, 4, 6, 7)$

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

- $F_1(x, y, z) = xz' + yz$

Example

- $F_1(x, y, z) = \Sigma\,(0, 2, 4, 5, 6)$

y

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

z

$F_1(x, y, z) = z' + y'x$

Combining Squares

- By combining squares, we reduce number of literals in a product term, reducing the literal cost, thereby reducing the other two cost criteria

- On a 3-variable K-Map:
  - One square represents a minterm with three variables
  - Two adjacent squares represent a product term with two variables
  - Four "adjacent" terms represent a product term with one variable
  - Eight "adjacent" terms is the function of all ones (no variables) = 1.

Three-Variable Maps

- Reduced literal product terms for SOP standard forms correspond to <u>rectangles</u> on K-maps containing cell counts that are powers of 2.
- Rectangles of 2 cells represent 2 adjacent minterms; of 4 cells represent 4 minterms that form a "pairwise adjacent" ring.
- Rectangles can contain non-adjacent cells as illustrated by the "pairwise adjacent" ring above.

Three-Variable Map Simplification

- **Use a K-map to find an optimum SOP equation for $F(X, Y, Z) = \Sigma_m(0,1,2,4,6,7)$**

Four Variable Karnaugh Map

- 4 variables: x, y, z, t
  - 16 minterms

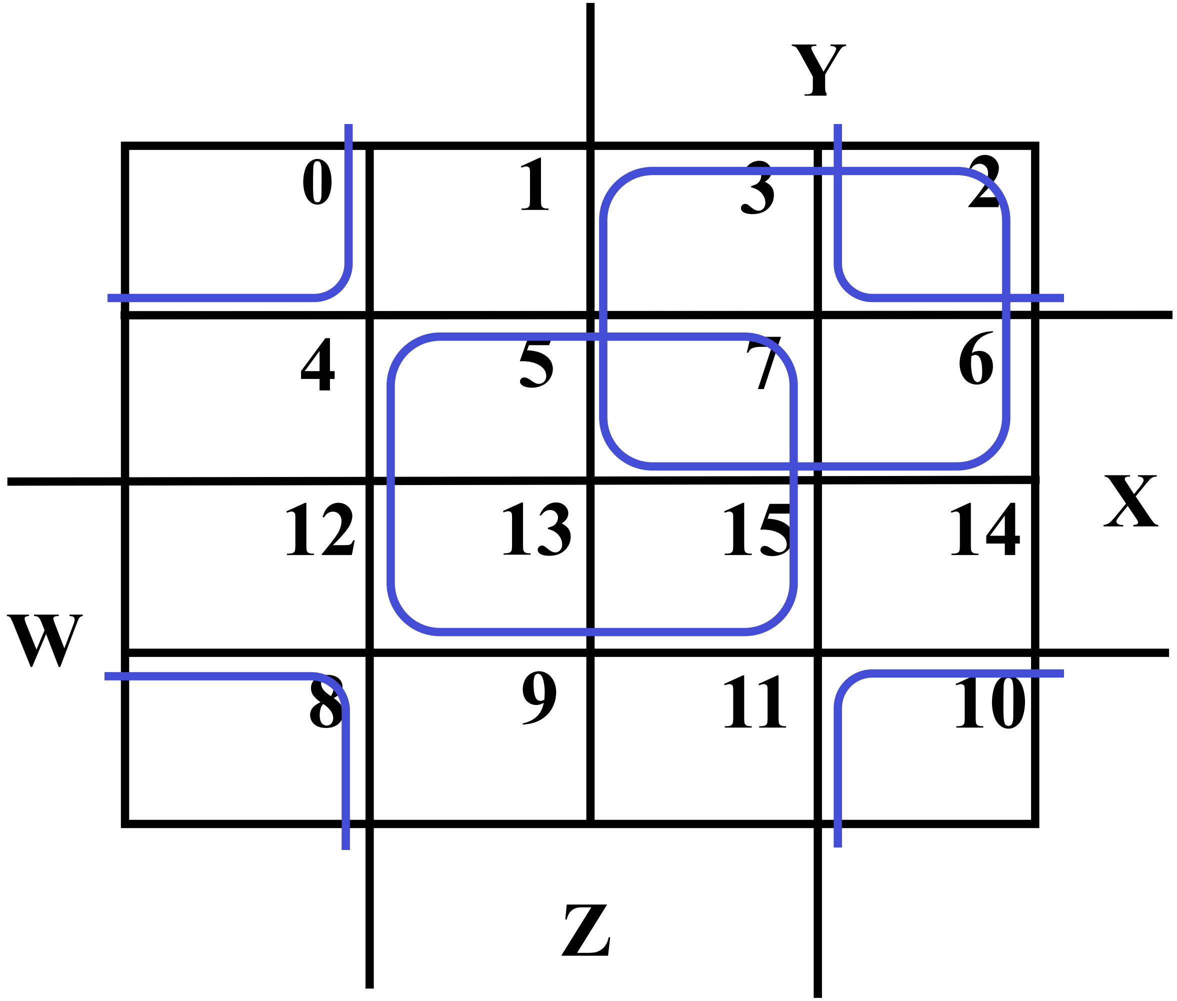
				z	z	
	xy \ zt	00	01	11	10	
	00	m_0	m_1	m_3	m_2	
	01	m_4	m_5	m_7	m_6	y
x	11	m_{12}	m_{13}	m_{15}	m_{14}	y
x	10	m_8	m_9	m_{11}	m_{10}	
			t	t		

Four Variable Terms

- Four variable maps can have rectangles corresponding to:
  - A single 1 = 4 variables, (i.e. Minterm)
  - Two 1s = 3 variables,
  - Four 1s = 2 variables
  - Eight 1s = 1 variable,
  - Sixteen 1s = zero variables (i.e. Constant "1")

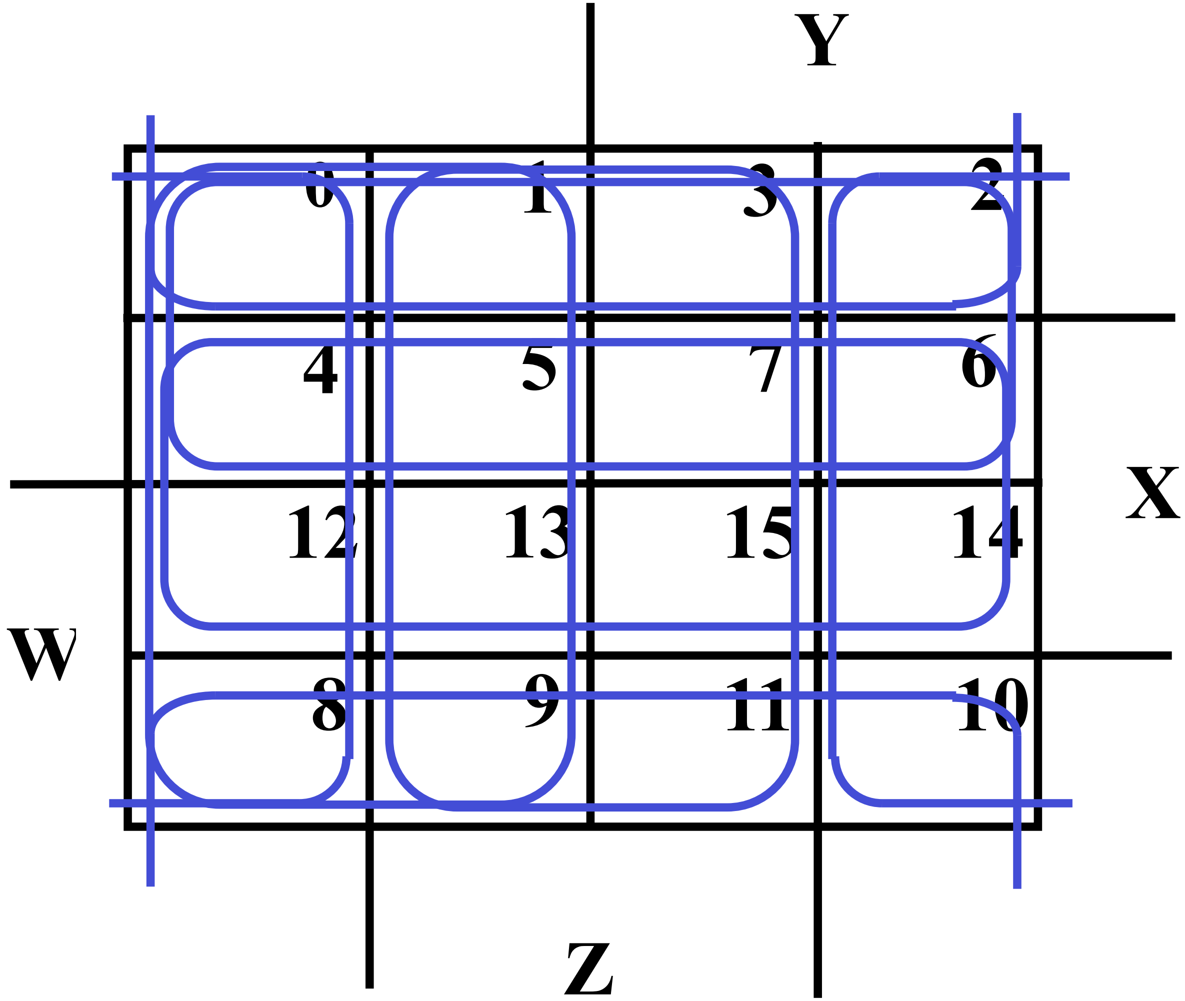
Four-Variable Maps

- **Example Shapes of Rectangles:**



Four-Variable Maps

- Example Shapes of Rectangles:



Example: $F(x,y,z,t) = \Sigma\,(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$

xy \ zt	00	01	11	10
00	1	1	0	1
01	1	1	0	1
11	1	1	0	1
10	1	1	0	0

- $F(x,y,z,t) = z' + x't' + yt'$

Example: F(x,y,z,t)=x'y'z'+y'zt'+x'yzt'+xy'z'

xy \ zt	00	01	11	10
00	1	1	0	1
01	0	0	0	1
11	0	0	0	0
10	1	1	0	1

- F(x,y,z,t) =y't'+z'y'+x'zt'

Optimization of POS Representation with Karnaugh Map

- Example: $f(x,y,z,t)=\Pi_M(1,4,5,8,11,12,14)$

xy \ zt	00	01	11	10
00	1	0	1	1
01	0	0	1	1
11	0	1	1	0
10	0	1	0	1

- $f(x_1,x_2,x_3,x_4)=$ $(x+z+t')$ $(x'+y'+t)$
  $(x'+z+t)$ $(x'+y+z'+t')$
  $(x+y'+z)$ or $(y'+z+t)$

Systematic Simplification

- A *Prime Implicant* is a product term obtained by combining the maximum possible number of adjacent squares in the map into a rectangle with the number of squares a power of 2.
- A prime implicant is called an *Essential Prime Implicant* if it is the <u>only</u> prime implicant that covers (includes) one or more minterms.
- Prime Implicants and Essential Prime Implicants can be determined by inspection of a K-Map.
- A set of prime implicants *"covers all minterms"* if, for each minterm of the function, at least one prime implicant in the set of prime implicants includes the minterm.

Five Variable or More K-Maps

- For five variable problems, we use *two adjacent K-maps*.
- It becomes harder to visualize adjacent minterms for selecting PIs.
- You can extend the problem to six variables by using four K-Maps
- 5 variable → 32 cells

tw \ xyz	000	001	011	010	110	111	101	100
00	m_0	m_4	m_{12}	m_8	m_{24}	m_{28}	m_{20}	m_{16}
01	m_1	m_5	m_{13}	m_9	m_{25}	m_{29}	m_{21}	m_{17}
11	m_3	m_7	m_{15}	m_{11}	m_{27}	m_{31}	m_{23}	m_{19}
10	m_2	m_6	m_{14}	m_{10}	m_{26}	m_{30}	m_{22}	m_{18}

Five Variable K-Maps

- Adjacent:
  - A cell in x = 0 map is adjacent to the same cell in x = 1 map.
  - Example, $m_4 \rightarrow m_{20}$ and $m_{15} \rightarrow m_{31}$ are adjacent.

yz \ tw	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7	m_6
11	m_{12}	m_{13}	m_{15}	m_{14}
10	m_8	m_9	m_{11}	m_{10}

x = 0

yz \ tw	00	01	11	10
00	m_{16}	m_{17}	m_{19}	m_{18}
01	m_{20}	m_{21}	m_{23}	m_{22}
11	m_{28}	m_{29}	m_{31}	m_{30}
10	m_{24}	m_{25}	m_{27}	m_{26}

x = 1

Example: Five Variable K-Maps

- F(x, y, z, t, w) = Σ (0, 2, 4, 6, 9, 13, 21, 23, 25, 29, 31)

yz \ tw	00	01	11	10
00	1			1
01	1			1
11		1		
10		1		

x = 0

yz \ tw	00	01	11	10
00				
01		1	1	
11		1	1	
10		1		

x = 1

- F(x,y,z,t,w) =

Don't Cares in K-Maps

- Sometimes a function table or map contains entries for which it is known:
  - the input values for the minterm will never occur, or
  - The output value for the minterm is not used
- In these cases, the output value need not be defined
- Instead, the output value is defined as a “don't care”
- By placing “don't cares” (an “x” entry) in the function table or map, the cost of the logic circuit may be lowered.
- Example 1: A logic function having the binary codes for the BCD digits as its inputs. Only the codes for 0 through 9 are used. The six codes, 1010 through 1111 <u>never occur</u>, so the output values for these codes are “x” to represent “don’t cares.”

Don't Cares in K-Maps

- **Example 2: A circuit that represents a very common situation that occurs in computer design has two distinct sets of input variables:**
  - **A, B, and C which take on all possible combinations, and**
  - **Y which takes on values 0 or 1.**

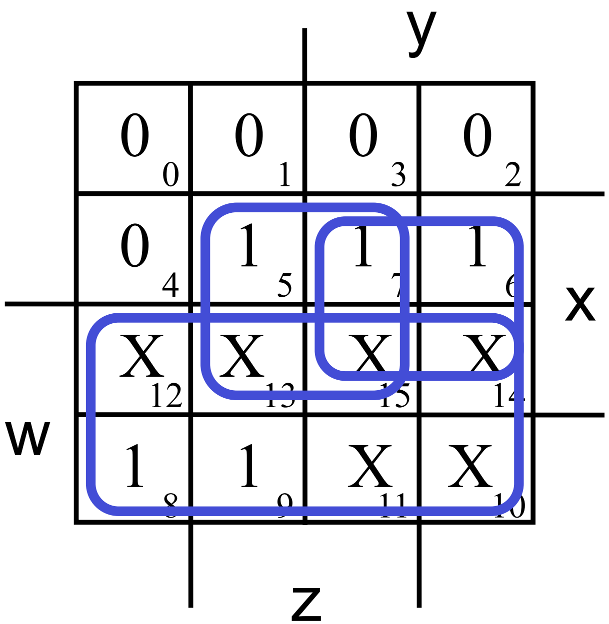
  and a single output Z. The circuit that receives the output Z observes it only for combinations of A, B, and C such A = 1 and B = 1 or C = 0, otherwise ignoring it. Thus, Z is specified only for those combinations, and for all other combinations of A, B, and C, Z is a don't care. Specifically, Z must be specified for $AB + \overline{C} = 1$, and is a don't care for :

$$\overline{AB + \overline{C}} = (\overline{A} + \overline{B})C = \overline{A}C + \overline{B}C = 1$$

- **Ultimately, each don't care "x" entry may take on either a 0 or 1 value in resulting solutions**
- **For example, an "x" may take on value "0" in an SOP solution and value "1" in a POS solution, or vice-versa.**
- **Any minterm with value "x" need not be covered by a prime implicant.**

Example: BCD "5 or More"

- **The map below gives a function F1(w,x,y,z) which is defined as "5 or more" over BCD inputs. With the don't cares used for the 6 non-BCD combinations:**



F1 (w,x,y,z) = w + x z + x y G = 7

- **This is much lower in cost than F2 where the "don't cares" were treated as "0s."**

$$F_2(w, x, y, z) = \overline{w}\,x\,z + \overline{w}\,x\,y + w\,\overline{x}\,\overline{y} \quad G = 12$$

- **For this particular function, cost G for the POS solution for $F_1(w,x,y,z)$ is not changed by using the don't cares.**

Product of Sums Example

- **Find the optimum POS solution:**

  $F(A, B, C, D) = \Sigma_m(3,9,11,12,13,14,15) + \Sigma_d(1,4,6)$

Optimization Algorithm

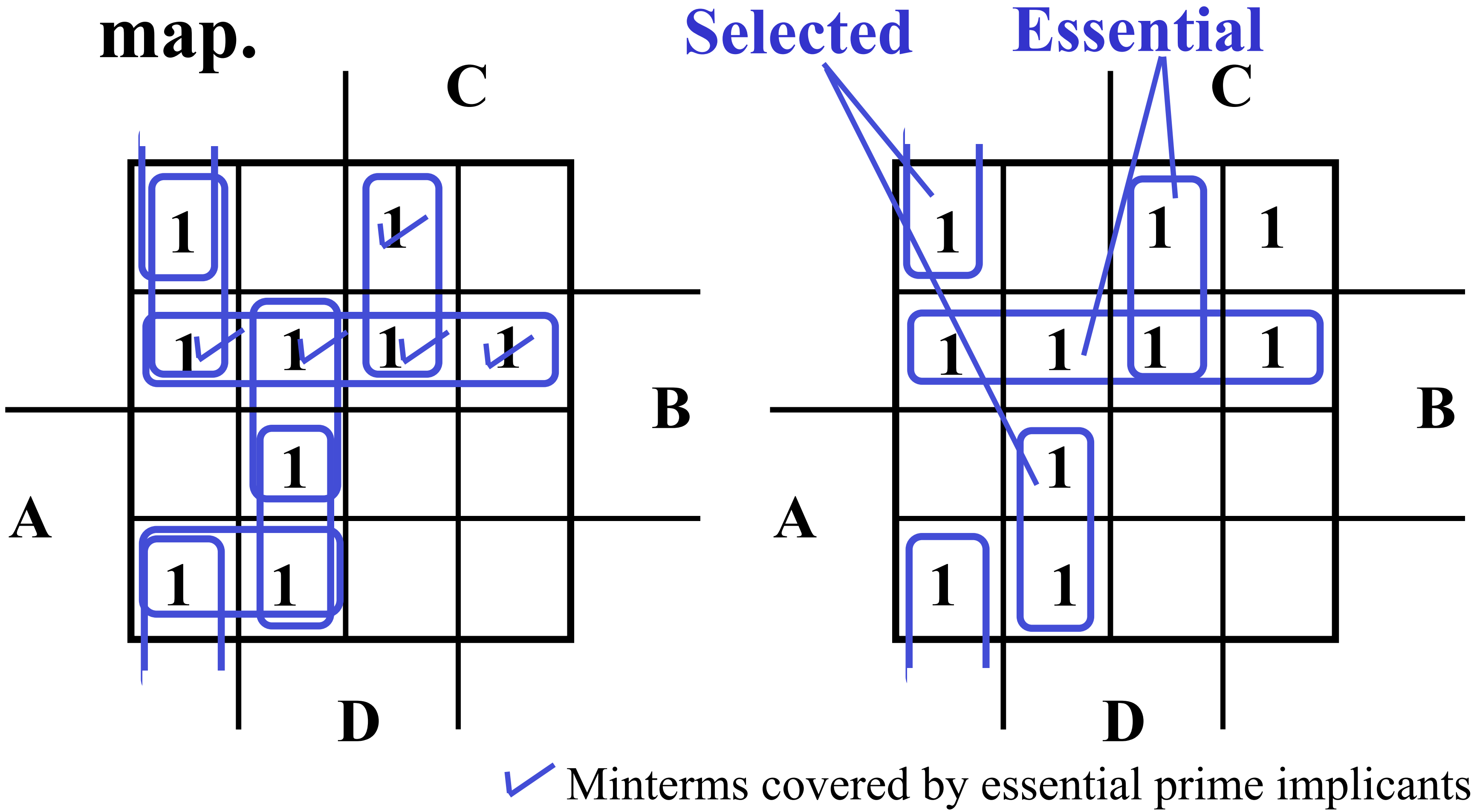
- Find <u>all</u> prime implicants.
- Include <u>all</u> essential prime implicants in the solution
- Select a minimum cost set of non-essential prime implicants to cover all minterms not yet covered:
  - Obtaining an optimum solution: See Reading Supplement - More on Optimization
  - Obtaining a good simplified solution: Use the Selection Rule

Prime Implicant Selection Rule

- Minimize the overlap among prime implicants as much as possible. In particular, in the final solution, make sure that each prime implicant selected includes at least one minterm not included in any other prime implicant selected.

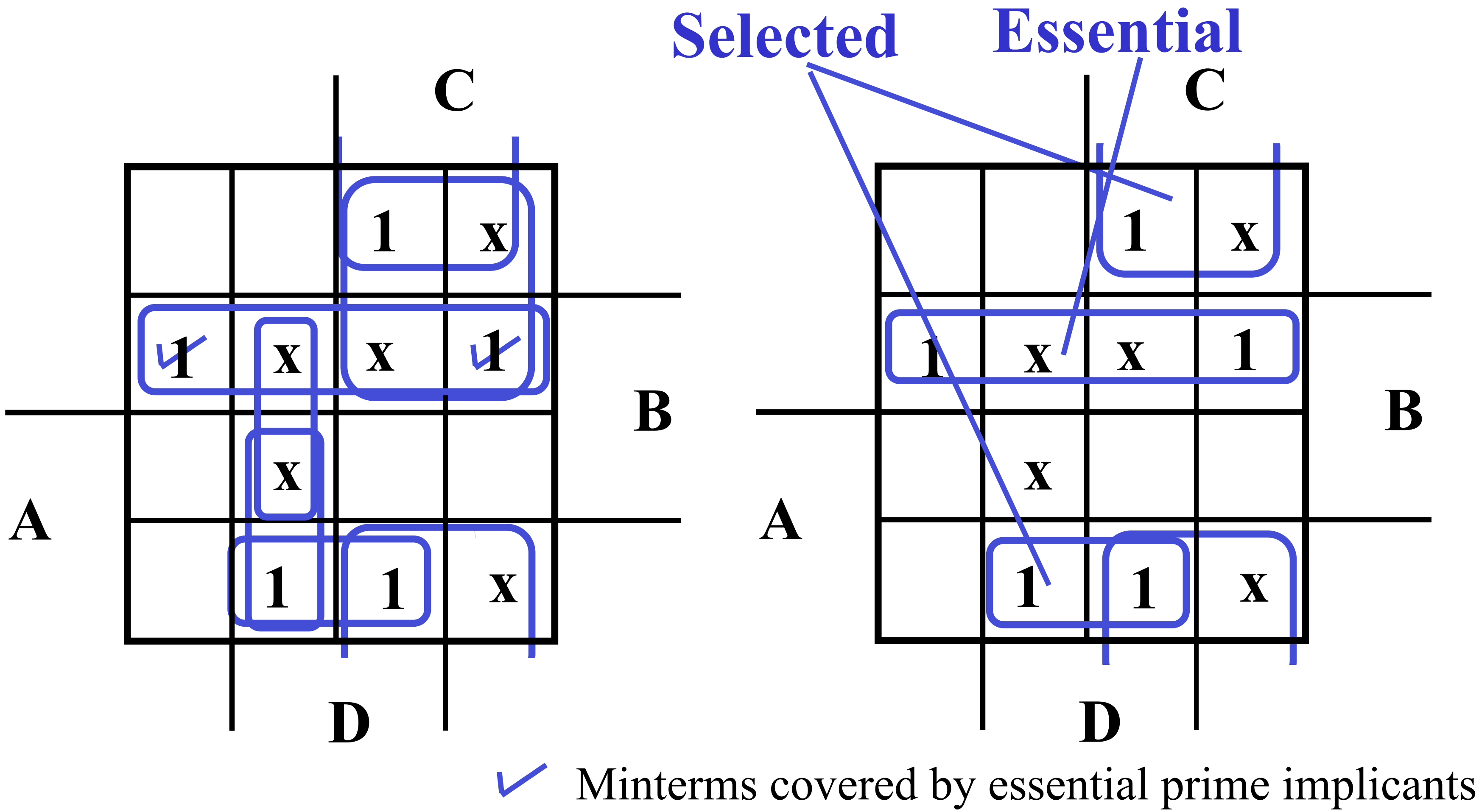
Selection Rule Example

- **Simplify F(A, B, C, D) given on the K-map.**



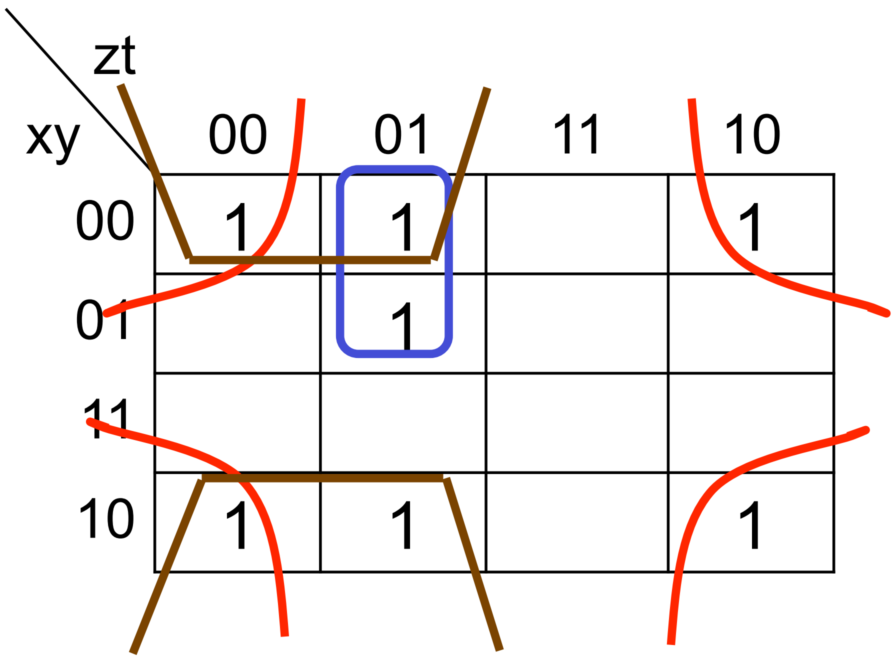
Selection Rule Example with Don't Cares

- Simplify F(A, B, C, D) given on the K-map.



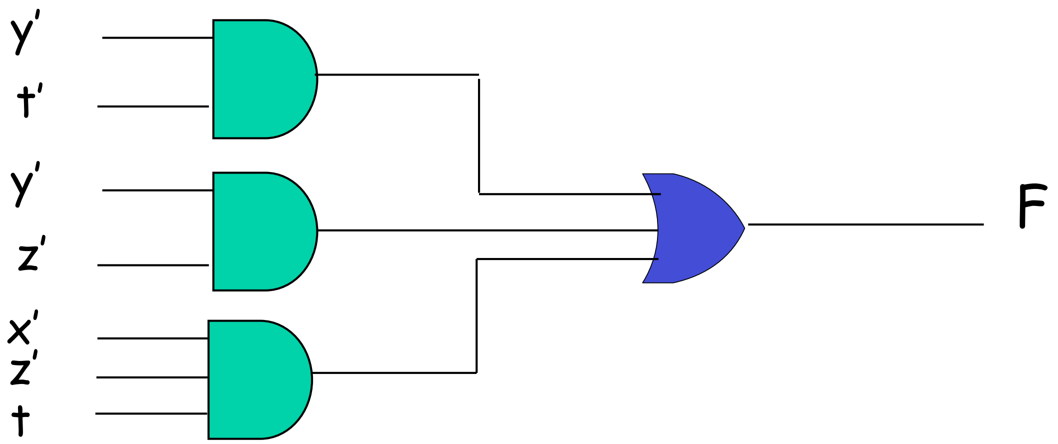
Example: Product of Sums

- $F(x, y, z, t) = \Sigma\,(0, 1, 2, 5, 8, 9, 10)$
  - Simplify this function in
    a. sum of products
    b. product of sums

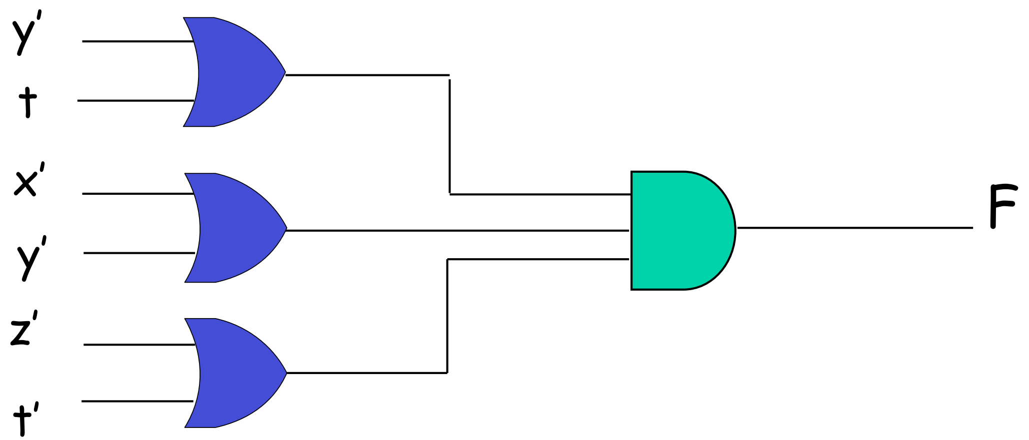


F(x, y, z, t) =

Example: Product of Sums



F(x,y,z,t) = y't' + y'z' + x'z't: sum of products implementation



F = (y' + t)(x' + y')(z' + t'): product of sums implementation