Computer Architecture I

CEG2136

DGD #1

Logic Functions Minimization

TA: Maryam Hezaveh

Today we will talk about:

- Logic function representation using
  a) Truth tables
  b) Boolean expressions
  c) K-maps
  d) Logic diagrams
- Simplification of Boolean functions using
  1. BOOLEAN ALGEBRA identities
  2. K-maps
- Combinational Circuits

... and will solve questions

1. Questions 1.3, 1.7 and 1.13 from Mano's textbook
2. Use the K-maps to simplify the following function:

   F(a,b,c,d) = a'b'cd' + a'bd + abc' + ab'c'd' + ab'cd' + ab
3. Find the PI's (prime implicants) of the following function:

   F(a,b,c,d) = a' b'c d + a'bd + abc'+ ab'c'd'+ ab'cd' + abcd
4. Given the following Boolean function together with the "don't care" conditions d:

   $F(x, y, z, w) = \sum(2, 6, 7, 11, 15)$

   $d(x, y, z, w) = \sum(0, 8, 10, 12, 13)$

a) find all the prime implicants and the essential prime implicants
b) find the minimal sum-of-products (SoP) expression
c) implement your minimized function with NAND gates only and draw the logic diagram of your circuit

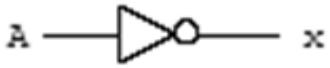
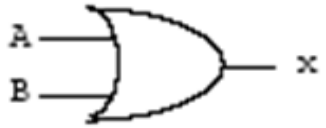
Boolean function and logic diagram

- **Boolean algebra**: Deals with logic variables and logic operations operating on those variables.

- **Logic diagram** (schematic): Composed of graphic symbols for logic gates. A simple circuit sketch that represents inputs and outputs of Boolean functions.

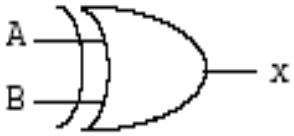
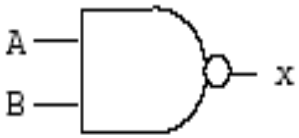
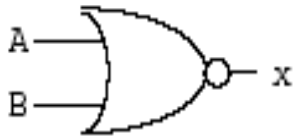
Gates

- Refer to the hardware to implement Boolean operators.
- The most basic gates are

Name	Graphic symbol	Algebraic function	Truth table	
Inverter		x = A'	A | x 0 | 1 1 | 0	
AND		x = AB	A B | x 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1	True if both are true.
OR		x = A + B	A B | x 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1	True if either one is true.

Boolean function and truth table

- Other common gates include:

Name	Graphic symbol	Algebraic function	Truth table	
Exclusive-OR (XOR)		$x = A \oplus B$ $= A'B + AB'$	A B | x 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0	Parity check: True if only one is true.
NAND		$x = (AB)'$	A B | x 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0	Inversion of AND.
NOR		$x = A + B$	A B | x 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0	Inversion of OR.

Basic Identities of Boolean Algebra

Existence of 1 and 0 element	***Existence of complement***	***Commutativity***	***Involution***
$x + 0 = x$	$x + x = x$	$x + y = y + x$	$(x')' = x$
$x \cdot 0 = 0$	$x \cdot x = x$	$xy = yx$	
$x + 1 = 1$	$x + x' = x$		
$x \cdot 1 = 1$	$x \cdot x' = 0$		
Distributivity	***DeMorgan's Theorem***	***Associativity***	
$x (y + z) = xy + xz$	$(x + y)' = x' y'$	$x + (y + z) = (x + y) + z$	
$x + yz = (x + y)(x + z)$	$(xy)' = x' + y'$	$x (yz) = (xy) z$	

Minimization of Boolean expressions

- *The minimization will result in reduction of*
  - *the number of gates (resulting from less number of terms) and*
  - *the number of inputs per gate (resulting from less number of variables per term)*
- *The minimization will reduce cost, efficiency and power consumption.*
- **y(x+x`) = y.1 = y**
- **y+xx`= y+0 = y**
- **(x`y+xy`) = x⊕y**
- **(x`y`+xy) = (x⊕y)`**

Q1-3

- Simplify the following expressions using Boolean algebra
  1. A + AB = A (1 + B) = A
  2. AB + AB' = A(B + B') = A
  3. A'BC + AC = (A'B + A) C = (A'+A)(B+A)C = (A+B)C
  4. A'B + ABC' +ABC = A'B + AB(C+C')=A'B + AB= (A'+A)B=B

Minimum some of products (MSOP)

- The *minimum sum of products (MSOP)* of a function, *f*, is a SOP representation of *f* that contains the fewest number of product terms and fewest number of literals of any SOP representation of *f*.
- The minimization can start from the minterm representation of the function (also called Canonical SoP or Canonical disjunctive form) or from any SoP expression.

Example

Truth table contains **logic** variables (a,b,c,d,f) and constants (0 for FALSE, and 1 for TRUE: they are not numbers!).
IF we map *d* to 2^0, *c* to 2^1, *b* to 2^2, *a* to 2^3, we can interprete *abcd* as binary numbers and then we can order all logic combinations in ascending order of these numbers.

- ***Use BOOLEAN ALGEBRA theorems and axioms to minimize the following logic function represented as a sum of products (SOP):***

f(a,b,c,d) = ∑m(3,7,11,12,13,14,15)

= a`b`cd + a`bcd + ab`cd + abc`d`+ abc`d+ abcd` + abcd
=cd(a`b` + a`b + ab`+ab) + ab(c`d` + c`d + cd` + cd)
=cd(a`[b` + b] + a[b`+b]) + ab(c`[d` + d] + c[d` + d])
=cd(a`[1] + a[1]) + ab(c`[1] + c[1]) = cd(a' + a) + ab (c'+c)
= cd (1) + ab (1) = cd + ab cost: 6 CU (gate in's)

ab \ cd	00	01	11	10
00	0	1	3 1	2
01	4	5	7 1	6
11	12 1	13 1	15 1	14 1
10	8	9	11 1	10

	abcd	
3	0011	a`b`cd
7	0111	a`bcd
11	1011	ab`cd
12	1100	abc`d`
13	1101	abc`d
14	1110	abcd`
15	1111	abcd

	a	b	c	d	f
(0)	0	0	0	0	0
(1)	0	0	0	1	0
(2)	0	0	1	0	0
(3)	0	0	1	1	1
(4)	0	1	0	0	0
(5)	0	1	0	1	0
(6)	0	1	1	0	0
(7)	0	1	1	1	1
(8)	1	0	0	0	0
(9)	1	0	0	1	0
(10)	1	0	1	0	0
(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

Q1-7

Given the Boolean function

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

a. List the truth table of the function
b. Draw the logic diagram using the original Boolean expression, i.e., disjunctive (SoP) canonical Boolean expression.
c. Simplify the canonical SoP algebraic expression using Boolean algebra
d. List the truth table of the function from the simplified expression and show that it is the same as the truth table in part (a)
e. Draw the logic diagram from the simplified expression and compare the total number of gates with the diagram of part (b)

Q1-7(a)

- List the truth table of the function

$$f = xy'z + x'y'z + xyz$$

x	y	z	F
0	0	0	**0**
0	0	1	**1**
0	1	0	**0**
0	1	1	**0**
1	0	0	**0**
1	0	1	**1**
1	1	0	**0**
1	1	1	**1**

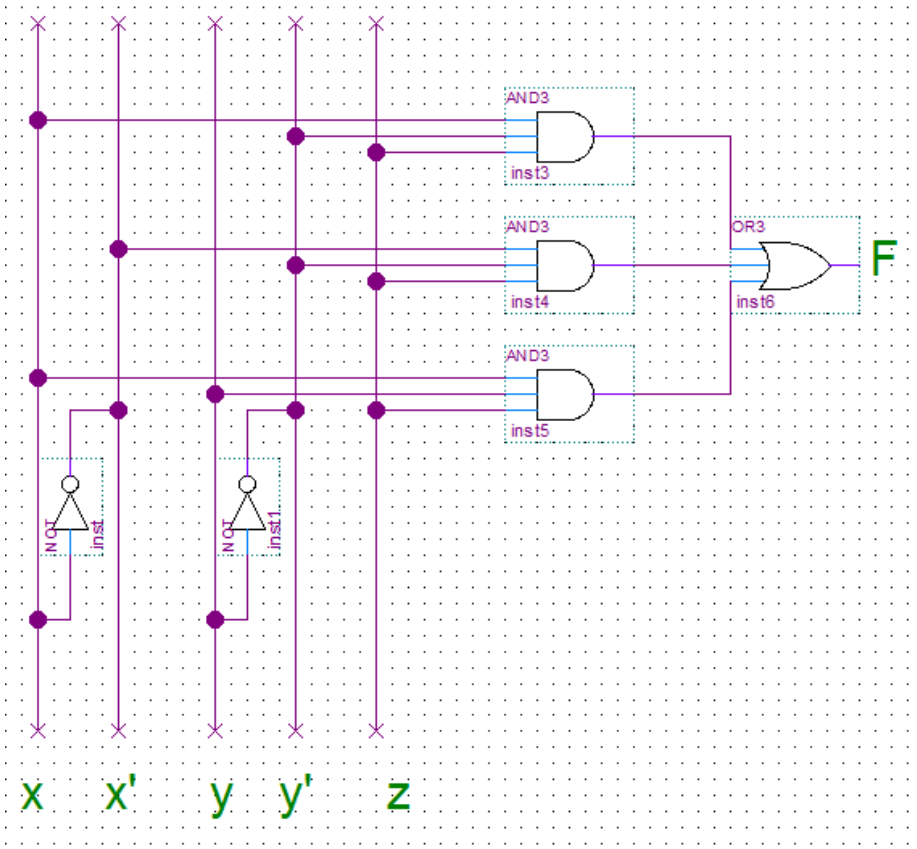
Q1-7(b)

- Draw the logic diagram using the original Boolean expression, i.e., disjunctive (SoP) canonical Boolean expression:

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

Cost: 12 in's

14 in's with inverters



Q1-7(c)

Simplify the canonical SoP algebraic expression using Boolean algebra

$$F = xy'z+x'y'z+xyz$$
$$\equiv y'z(x+x')+xz(y+y')$$
$$\equiv y'z+xz$$

$$F_{min} = y'z+xz$$

Q1-7(d)

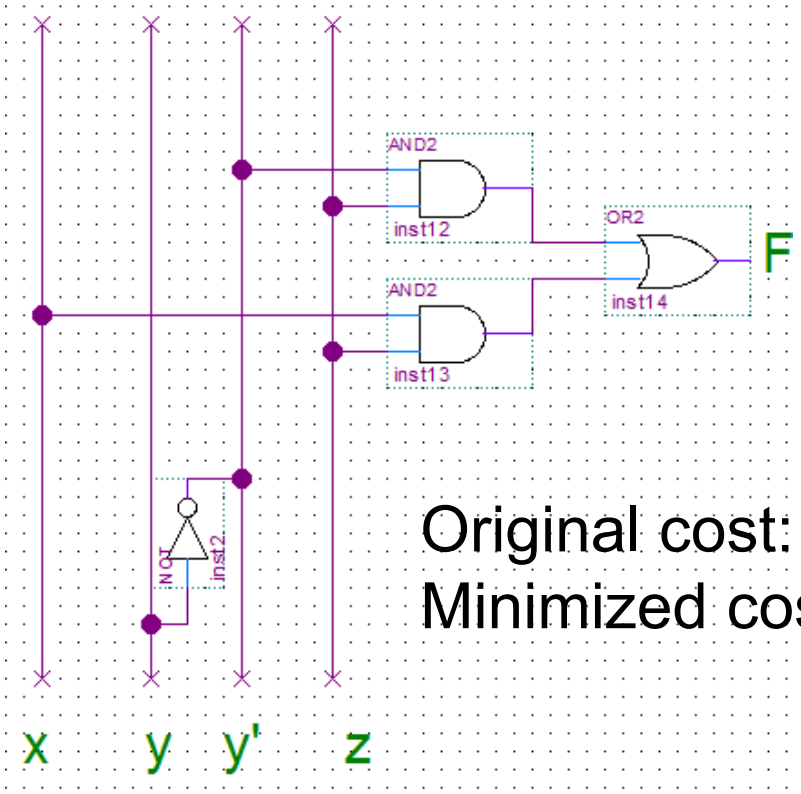
- List the truth table of the function from the simplified expression and show that is the same as the truth table in part (a)

$$F = y'z + xz$$

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

Q1-7(e)

- Draw the logic diagram from the simplified expression and compare the total number of gates with the diagram of part (b)



$F_{min} = y'z+xz$

Original cost: 12 in's (14 in's with inverters)
Minimized cost: 6 in's (7 in's with inverter)

Minimum product of sums (MPOS)

- Canonical POS (or conjunctive form) represents the 0's of a function and it is expressed as a product of MAXTERMS ($\prod M_k$).
- The *minimum product of sums (MPOS)* of a function, *f*, is a POS representation of *f* that contains the fewest number of sum terms and the fewest number of literals of any POS representation of *f.*
- The zeroes are considered exactly the same as ones in the case of sum of product (SOP)

ABC	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7
000	**0**	1	1	1	1	1	1	1
001	1	**0**	1	1	1	1	1	1
010	1	1	**0**	1	1	1	1	1
011	1	1	1	**0**	1	1	1	1
100	1	1	1	1	**0**	1	1	1
101	1	1	1	1	1	**0**	1	1
110	1	1	1	1	1	1	**0**	1
111	1	1	1	1	1	1	1	**0**

A	B	C	Maxterms
0	0	0	$A + B + \overline{C} = M_0$
0	0	1	$A + \overline{B} + C = M_1$
0	1	0	$A + \overline{B} + \overline{C} = M_2$
0	1	1	$\overline{A} + B + C = M_3$
1	0	0	$\overline{A} + B + \overline{C} = M_4$
1	0	1	$\overline{A} + \overline{B} + C = M_5$
1	1	0	$\overline{A} + \overline{B} + \overline{C} = M_6$
1	1	1	$\overline{A} + \overline{B} + \overline{C} = M_7$

Example - POS Representation

$f(a,b,c,d) = \prod M(0,1,2,4,5,6,8,9,10) = \sum m(3,7,11,12,13,14,15)$

Canonical POS = Canonical conjunctive form = :

1. Direct with maxterm POS: $f(a,b,c,d) = \prod M(0,1,2,4,5,6,8,9,10)$

M0, M1, M2, M4,

$f(a,b,c,d)$ = [(a+b+c+d)(a+b+c+d`) (a+b+c'+d) (a+b'+c+d)

M5, M6, M8, M9, M10

(a+b'+c+d')(a+b'+c'+d) (a'+b+c+d) (a'+b+c+d') (a'+b+c'+d)

2. Or negate f' (the complement of the function) SoP and then apply De Morgan:

$f'(a,b,c,d) = \sum m\ (0,1,2,4,5,6,8,9,10)$

$\overline{f'(a,b,c,d)} = \overline{(a'\bullet b'\bullet c'\bullet d')+(a'\bullet b'\bullet c'\bullet d)+(a'\bullet b`\bullet c\bullet d`)+(a`\bullet b\bullet c`\bullet d`)+}$

$+(a`\bullet b\bullet c'\bullet\ d)+(a`\bullet b\bullet c\bullet\ d`)+(a\bullet b`\bullet c`\bullet d')+(a\bullet b`\bullet c`\bullet d)+(a\bullet b'\bullet c\bullet d')]$

$f(a,b,c,d) = [\overline{(a'\bullet b'\bullet c'\bullet d')}\bullet\overline{(a'\bullet b'\bullet c'\bullet d)}\bullet\overline{(a'\bullet b`\bullet c\bullet d`)}\bullet\overline{(a`\bullet b\bullet c`\bullet d`)}\bullet$

$\bullet\overline{(a`\bullet b\bullet c'\bullet\ d)}\bullet\overline{(a`\bullet b\bullet c\bullet\ d`)}\bullet\overline{(a\bullet b`\bullet c`\bullet d')}\bullet\overline{(a\bullet b`\bullet c`\bullet d)}\bullet\overline{(a\bullet b'\bullet c\bullet d')}] =$

$f(a,b,c,d) = [(a+b+c+d)\bullet(a+b+c+d`)\bullet(a+b+c'+d)\bullet(a+b'+c+d)\bullet$

$\bullet(a+b'+c+d')\bullet(a+b'+c'+d)\bullet(a'+b+c+d)\bullet(a'+b+c+d')\bullet(a'+b+c'+d)$

	a	b	c	d	f	f'
(0)	0	0	0	0	0	1
(1)	0	0	0	1	0	1
(2)	0	0	1	0	0	1
(3)	0	0	1	1	1	0
(4)	0	1	0	0	0	1
(5)	0	1	0	1	0	1
(6)	0	1	1	0	0	1
(7)	0	1	1	1	1	0
(8)	1	0	0	0	0	1
(9)	1	0	0	1	0	1
(10)	1	0	1	0	0	1
(11)	1	0	1	1	1	0
(12)	1	1	0	0	1	0
(13)	1	1	0	1	1	0
(14)	1	1	1	0	1	0
(15)	1	1	1	1	1	0

A	B	C	Maxterms
0	0	0	$A+B+\overline{C}=M_0$
0	0	1	$A+B+\overline{C}=M_1$
0	1	0	$A+\overline{B}+C=M_2$
0	1	1	$A+\overline{B}+\overline{C}=M_3$
1	0	0	$\overline{A}+B+C=M_4$
1	0	1	$\overline{A}+B+\overline{C}=M_5$
1	1	0	$\overline{A}+\overline{B}+C=M_6$
1	1	1	$\overline{A}+\overline{B}+\overline{C}=M_7$

Karnaugh Maps (K-maps)

- Karnaugh maps -- A tool for representing Boolean functions of up to six variables.

- K-maps are tables of rows and columns with entries represent 1`s or 0`s of SOP and POS representations.

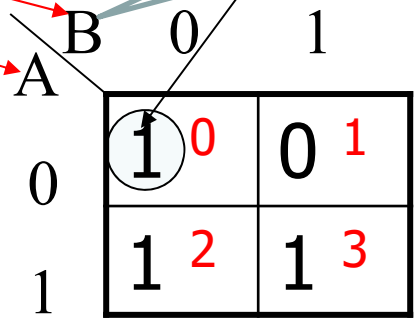
SOP Example

Two variable K-map: f(A,B)=∑m(0,1,3)=A`B`+A`B+AB

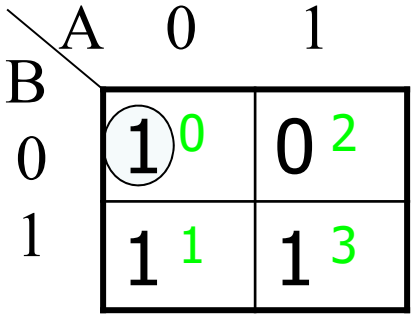
Truth table contains **logic** variables (A,B,f) and constants (0 for FALSE, and 1 for TRUE: they are not numbers!).
IF we map *B* to 2^0 and *A* to 2^1, we can interprete *AB* as binary numbers and then we can order all their logic combinations in ascending order of these numbers.
This is done just to make sure that we take all the possible combinations.

In this course we use this arrangement (a) rather than (b) !!!

	2^1(msb)	2^0(lsb)	
	A	***B***	***f***
0	0	0	1
1	0	1	1
2	1	0	0
3	1	1	1



(a)

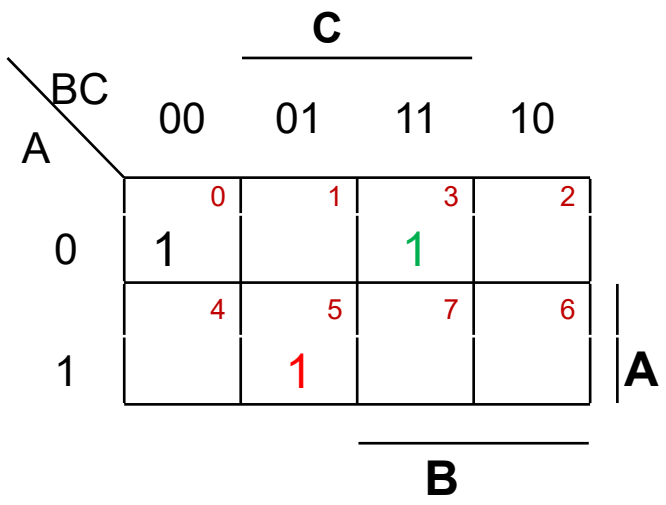


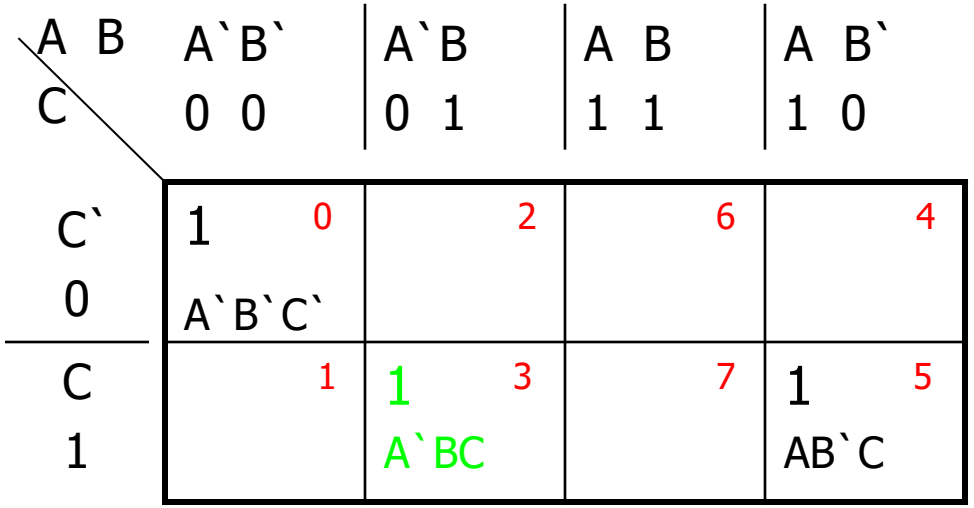
(b)

SOP Example

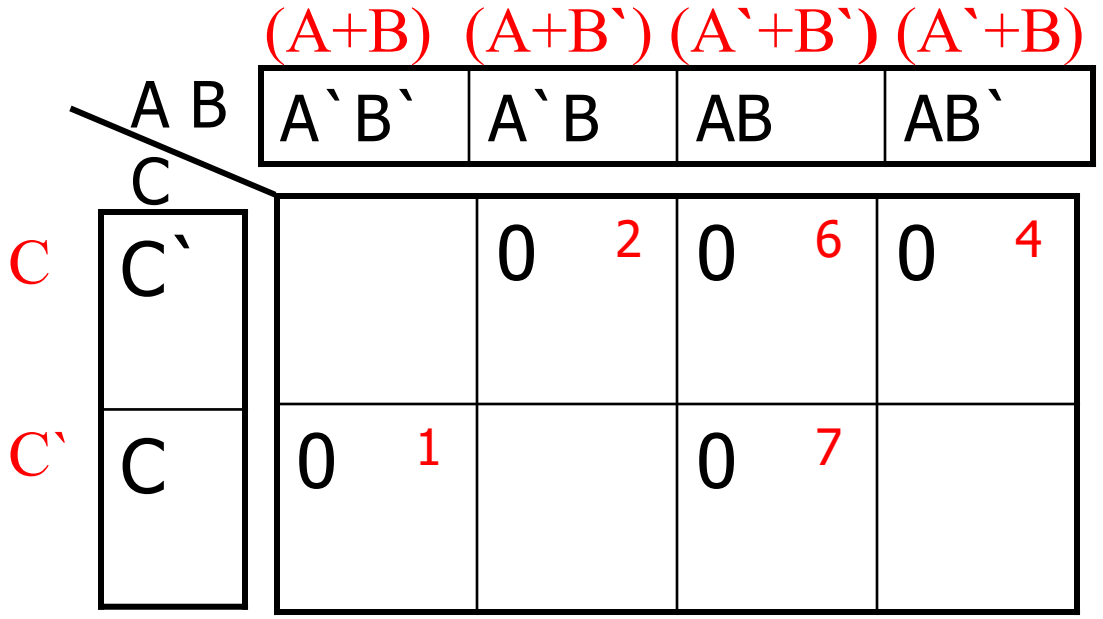
- *Three variable K-map*

$f(A,B,C) = \sum m(0,3,5) = A`B`C` + A`BC + AB`C$





POS example



$f(A,B,C) = \prod M(1,2,4,6,7)$

=(A+B+C`)(A+B`+C)(A`+B+C) )(A`+B`+C) (A`+B`+C`)

Note that the complements are (0,3,5) which are the minterms of the previous example

Q3 - Prime Implicants

Find the prime implicants (PI's) of the following function:

F(a,b,c,d) =

a' b'c d + a'bd + abc'+ ab'c'd'+ ab'cd' + abcd

- A **prime implicant (PI)** of a function is an implicant that cannot be covered by a more reduced (with fewer literals) implicant.
- A PI in Karnaugh map or K-map is
  - Any single "1" or group (a power of 2) of "1"-s that can be combined and can't be covered by a larger group in K-map.

Essential Prime Implicants

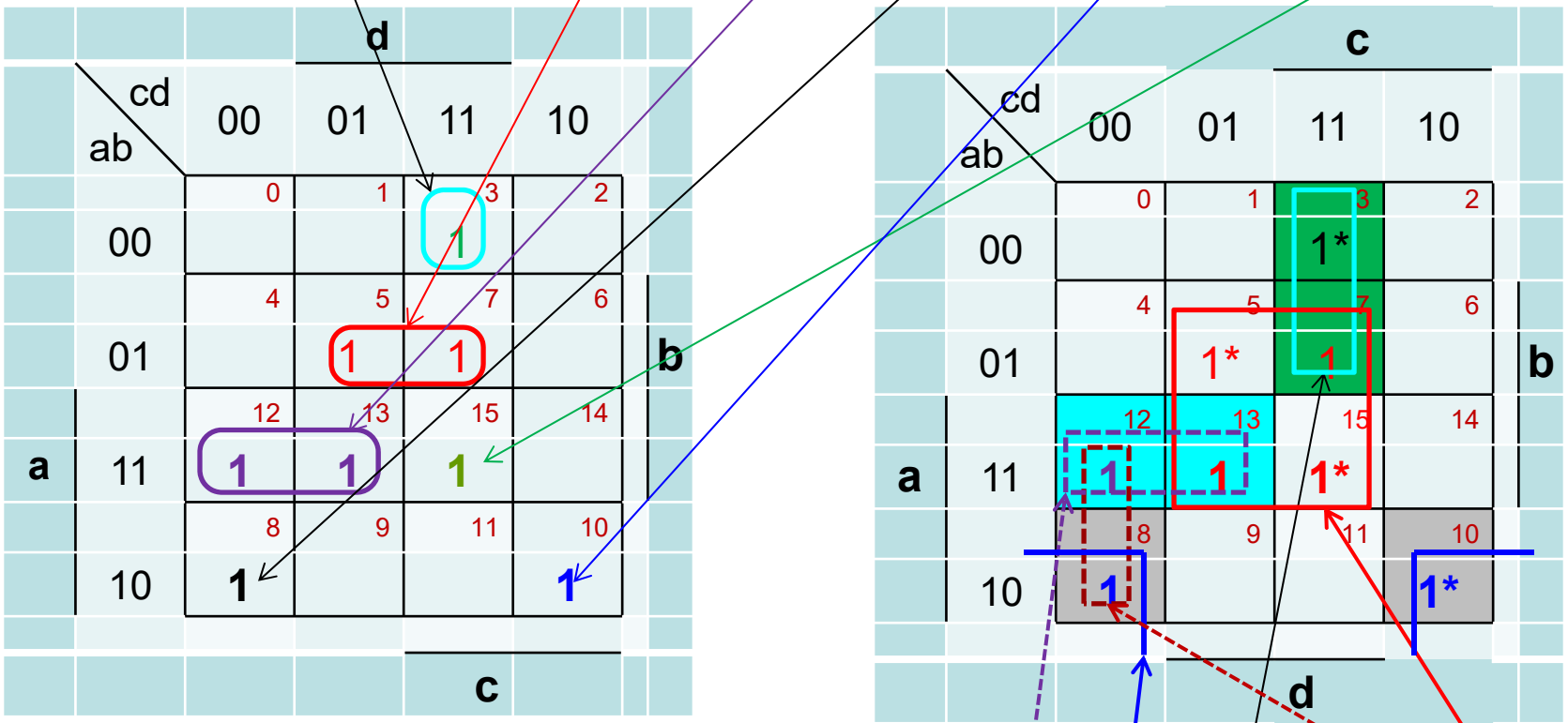
- **Essential prime implicants** (EPI) are prime implicants that cover an output of the function that no combination of other prime implicants is able to cover.

Logic Function minimization using K-maps

- To obtain the SoP of F_{min} (the minimized F), one has to find the minimum set of PI's which covers F as follows:
  1. Represent the given function F on a K-map
  2. Find the PI set of the function F
  3. Mark with * all K-map cells that contain a 1 which is covered by only a single PI; each PI that contains a * is an EPI.
  4. Include all EPI's in the minimal set which covers the function.
  5. Then find a minimum set from the rest of the non-EPI prime implicants that cover the remaining 1's on the map.

Find the PI's (prime implicants) of the following function:

$F(a,b,c,d) = a'b'c\,d + a'bd + abc' + ab'c'd' + ab'cd' + abcd$



Prime implicants PI : {***abc'***, ***ab'd'***, ***a'cd***, *ac'd'*, ***bd***}

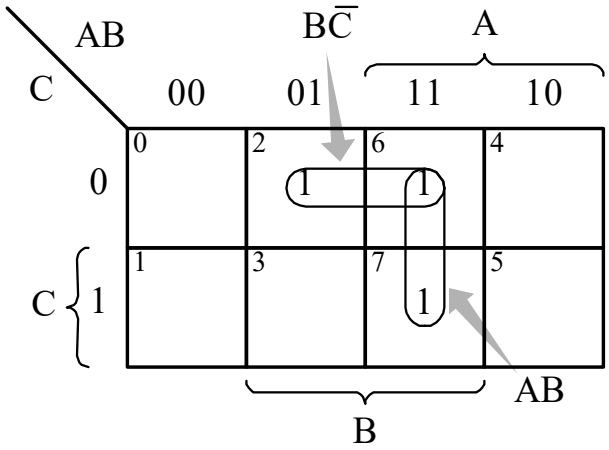
EPI (Essential Prime Implicants): {***ab'd'***, ***a'cd***, ***bd***} contain 1's marked with * and cover all 1's, except **abc'd'** which can be covered by *ac'd'* or ***abc'***.

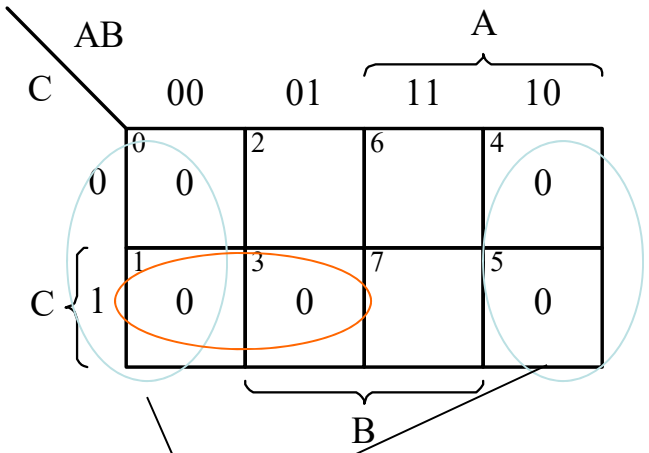
$F_{min1}(a,b,c,d) = ab'd' + a'cd + bd + ac'd'$ $F_{min2}(a,b,c,d) = ab'd' + a'cd + bd + abc'$

Notes on Simplification of Boolean Functions Using K-maps

- K-map cells that are physically adjacent are also logically adjacent (because this is how **we** draw the K-maps). Also, cells on an edge of a K-map are logically adjacent to cells on the opposite edge of the map.
- If two logically adjacent cells both contain logical 1s, the two cells can be combined to eliminate the variable that has value 1 in one cell's label and value 0 in the other.
- Always combine as many cells in a group as possible. This will result in the fewest number of literals in the term that represents the group.
- Make as few groupings as possible to cover all minterms. This will result in the fewest product terms.
- Always begin with the largest group, which means if you can find eight members group is better than two four groups and one four group is better than pair of two-group.

- Simplify f= A`BC`+ A B C`+ A B C using;
  (a) Sum of minterms. (b) Maxterms
- Each cell of an *n*-variable K-map has *n* logically adjacent cells.





F`= B` + A`C

F = B(A+C`)

Use method 2 for PoS: find SoP of f', then apply De Morgan

a- $f(A,B,C) = AB + BC'$ *b-* $f(A,B,C) = B(A + C')$

Q1-13

- Simplify the Boolean function *F* together with the don't care conditions *d* in

(1) sum-of-products form and

(2) product-of-sums form.

$F(w,x,y,z) = \sum(0,1,2,3,7,8,10) + dc(5,6,11,15)$

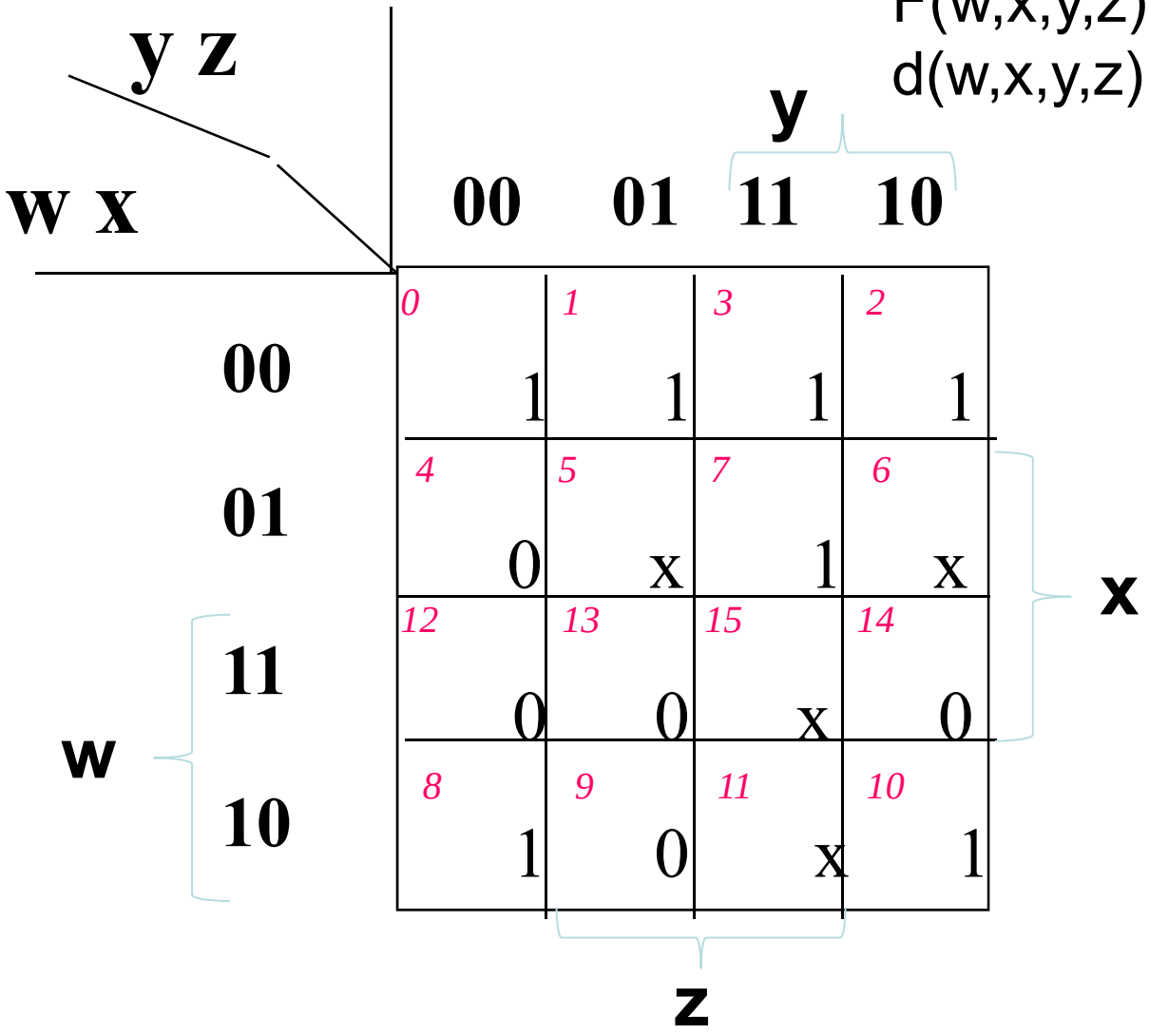
Or

$F(w,x,y,z) = \sum(0,1,2,3,7,8,10)$

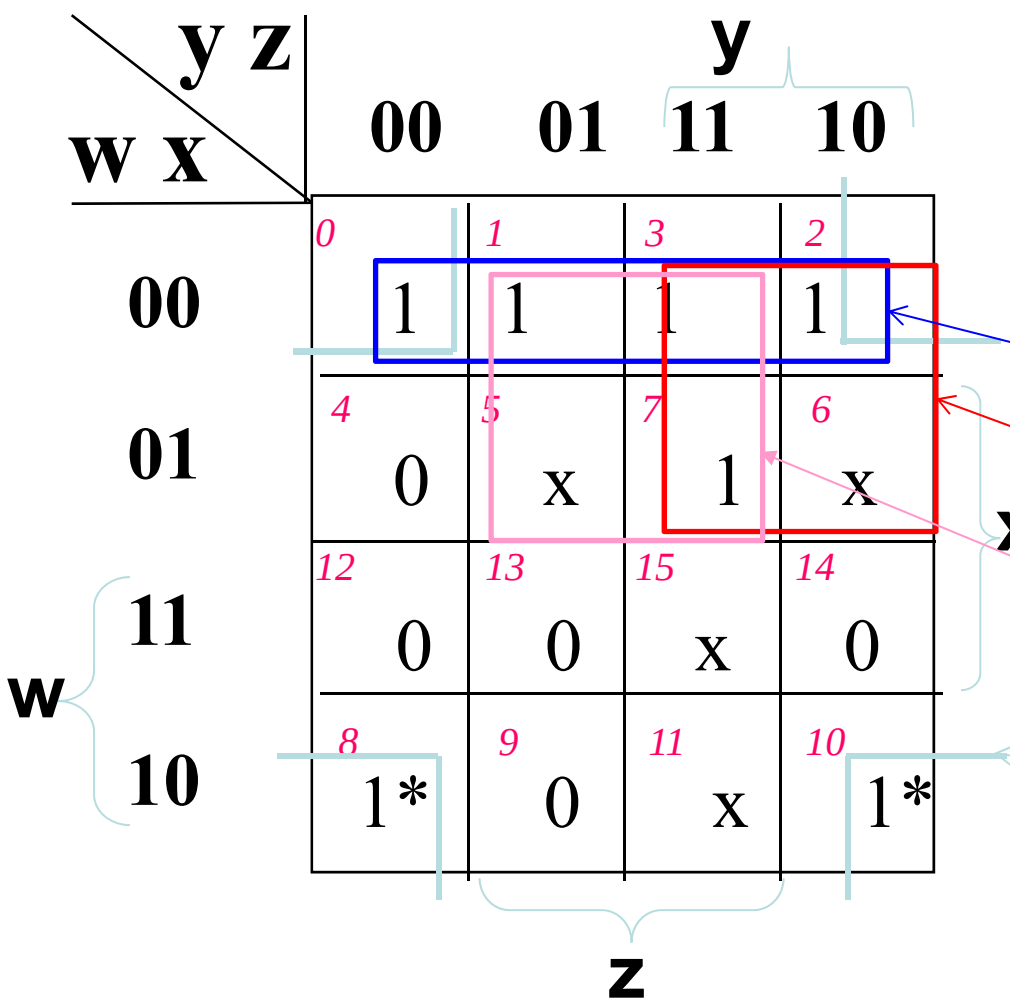
$d(w,x,y,z) = \sum(5,6,11,15)$

Q1-13

$F(w,x,y,z) = \sum(0,1,2,3,7,8,10)$
$d(w,x,y,z) = \sum(5,6,11,15)$



Q1-13 Sum-of-Products form



F(w,x,y,z) = ∑(0,1,2,3,7,8,10)
d(w,x,y,z) = ∑(5,6,11,15)
Minimize SoP by grouping 1's in the K-map, i.e., finding the PI's, EPI's (*)....

w'x'

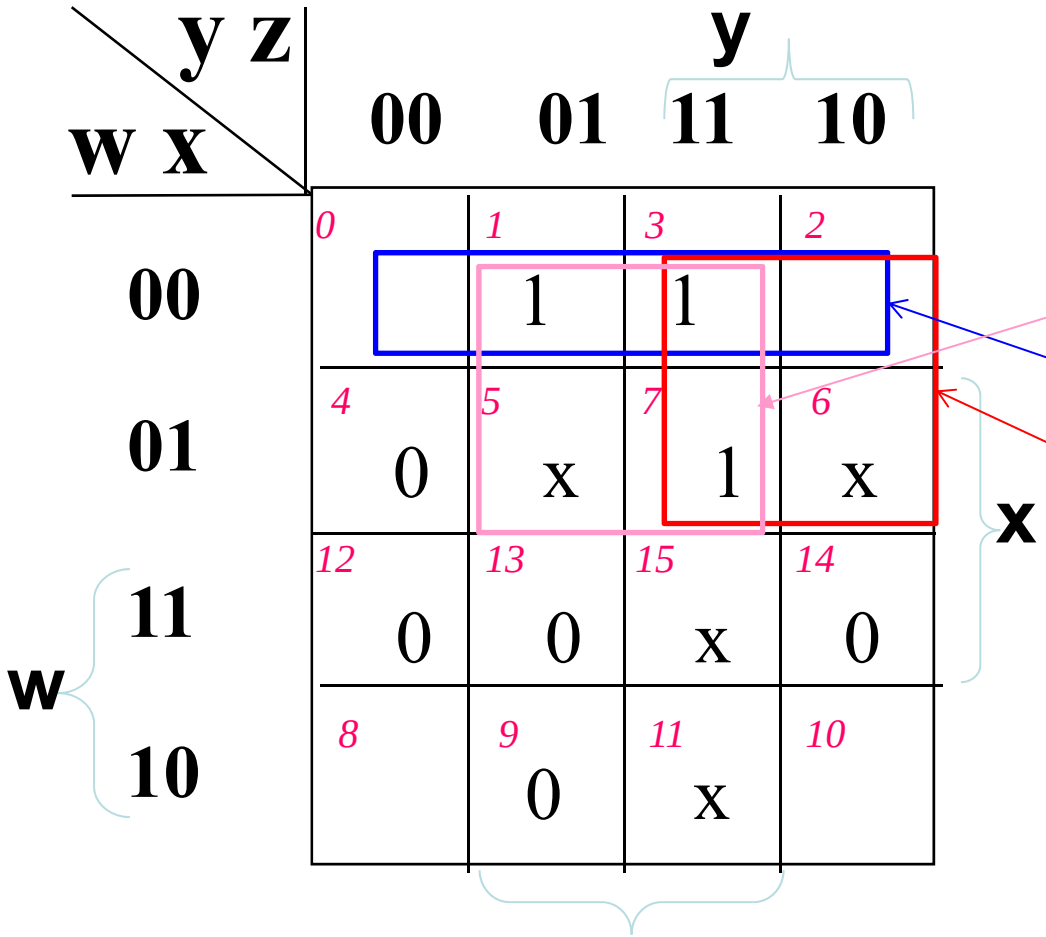
yw'

w'z

z'x' *

F = z'x' + …

Q1-13 Sum-of-Products form



No secondary EPI's (*)...
w'z is the cheapest cover of the rest of 1's

w'z

w'x'

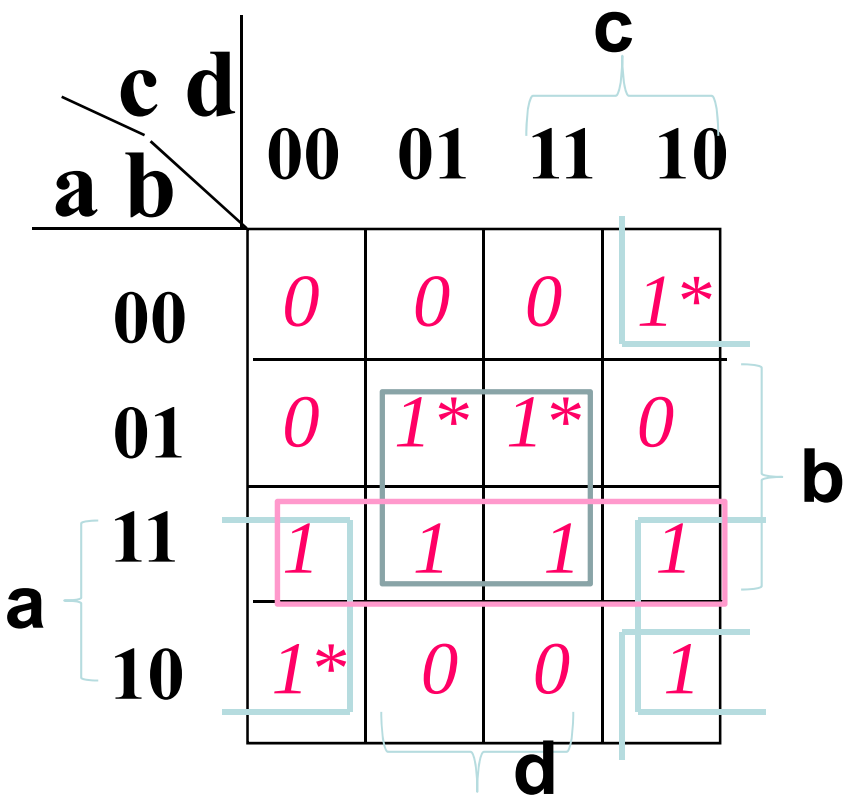
yw'

z'x' *

F = z'x' + w'z

Q2

- Use the K-maps to simplify the following function
  - F(a,b,c,d) = a'b'cd' + a'bd + abc' + ab'c'd' + ab'cd' + ab



PI's: bd, ad', b'cd', ab

EPI's: bd, ad', b'cd'
cover the whole function

F= bd + ad' + b'cd'

Q4

Given the following Boolean function together with the "don't care" conditions d:

$$F(x, y, z, w) = \sum(2, 6, 7, 11, 15)$$
$$d(x, y, z, w) = \sum(0, 8, 10, 12, 13)$$

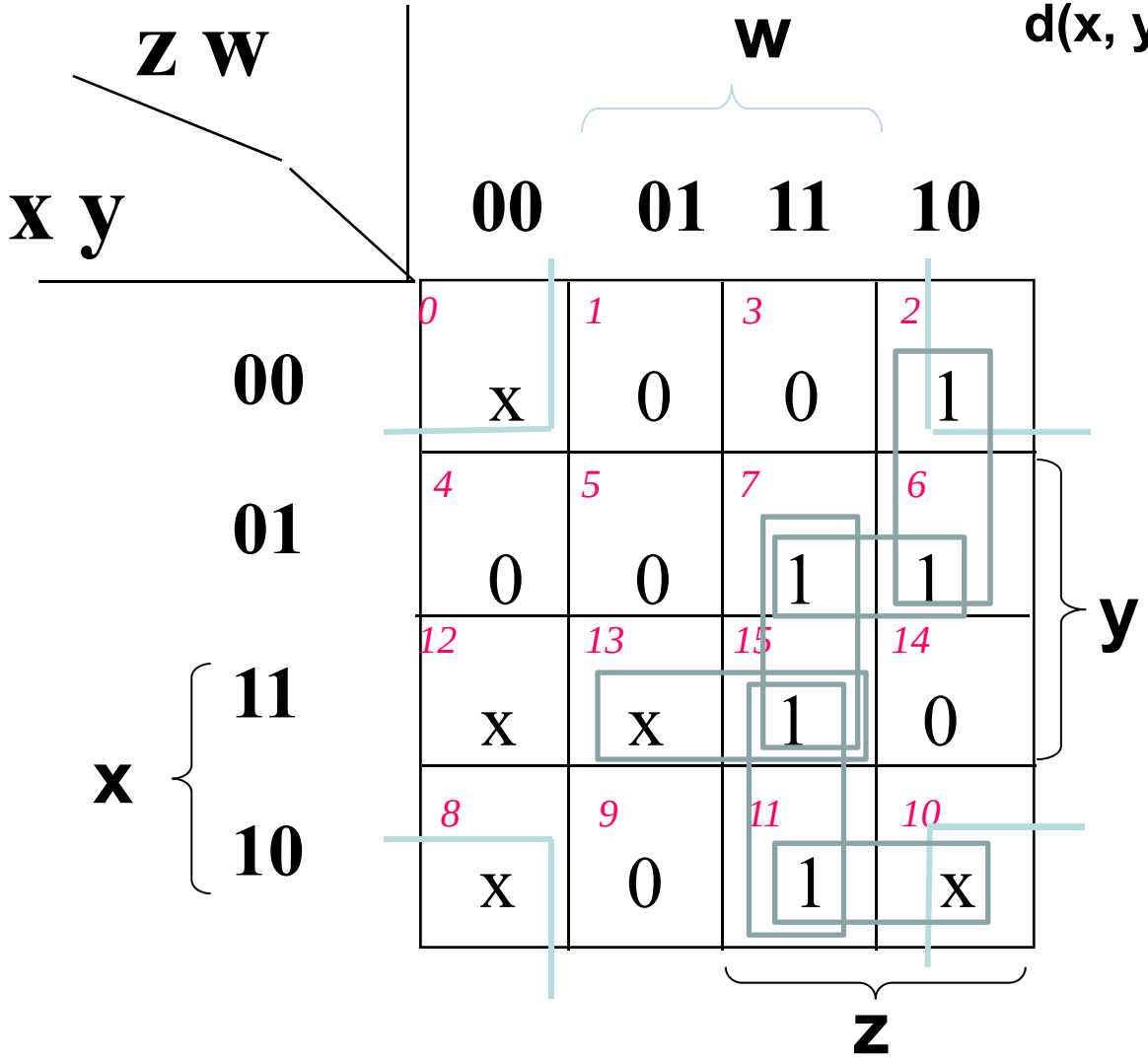
a) find all the prime implicants and the essential prime implicants

b) find the minimal sum-of-products (SoP) expression

c) implement your minimized function with NAND gates only and draw the logic diagram of your circuit

Q4(a)

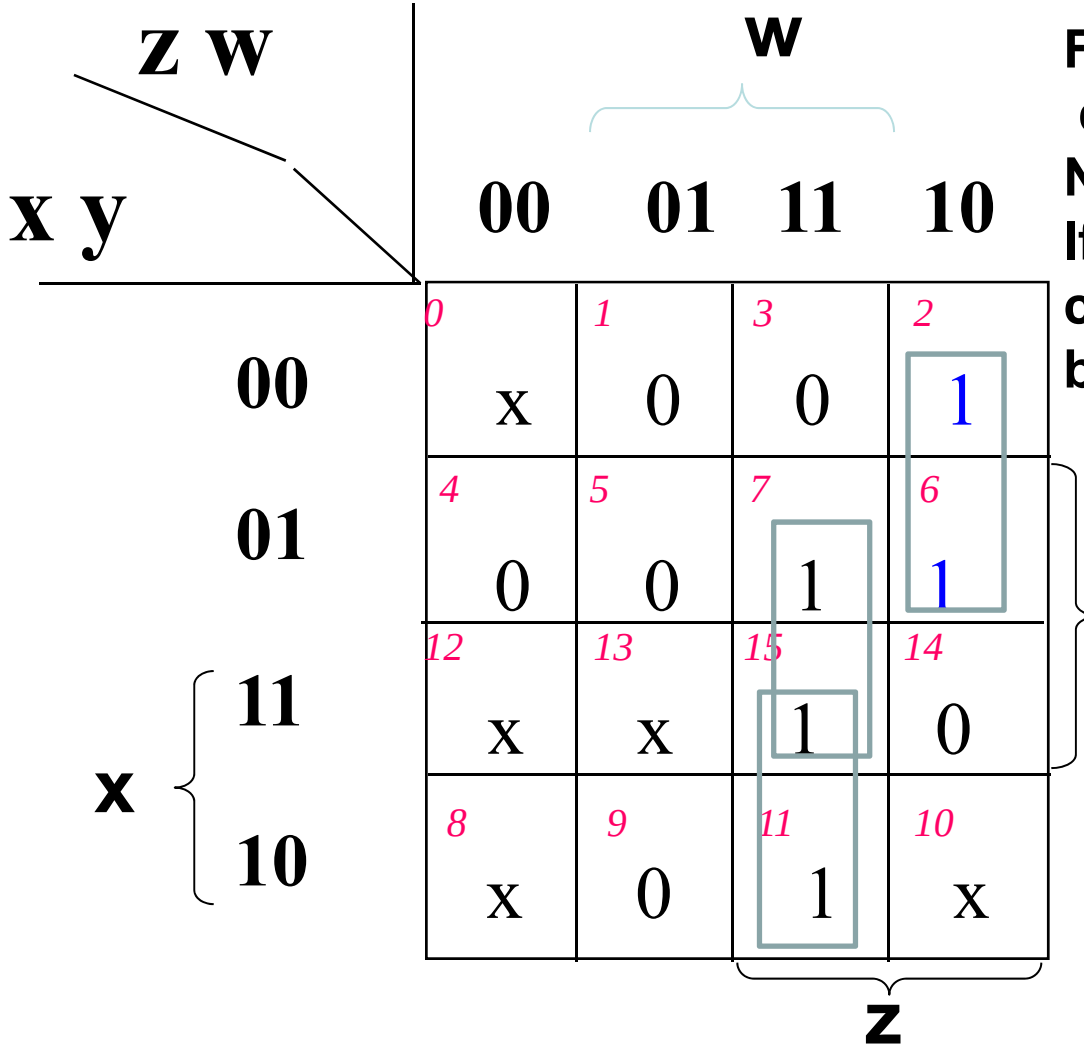
$F(x, y, z, w) = \sum(2, 6, 7, 11, 15)$
$d(x, y, z, w) = \sum(0, 8, 10, 12, 13)$



PIs:

1. y'w'
2. x'zw'
3. x'yz
4. yzw
5. xyw
6. xzw
7. xy'z

Q4(b) Minimal SoP

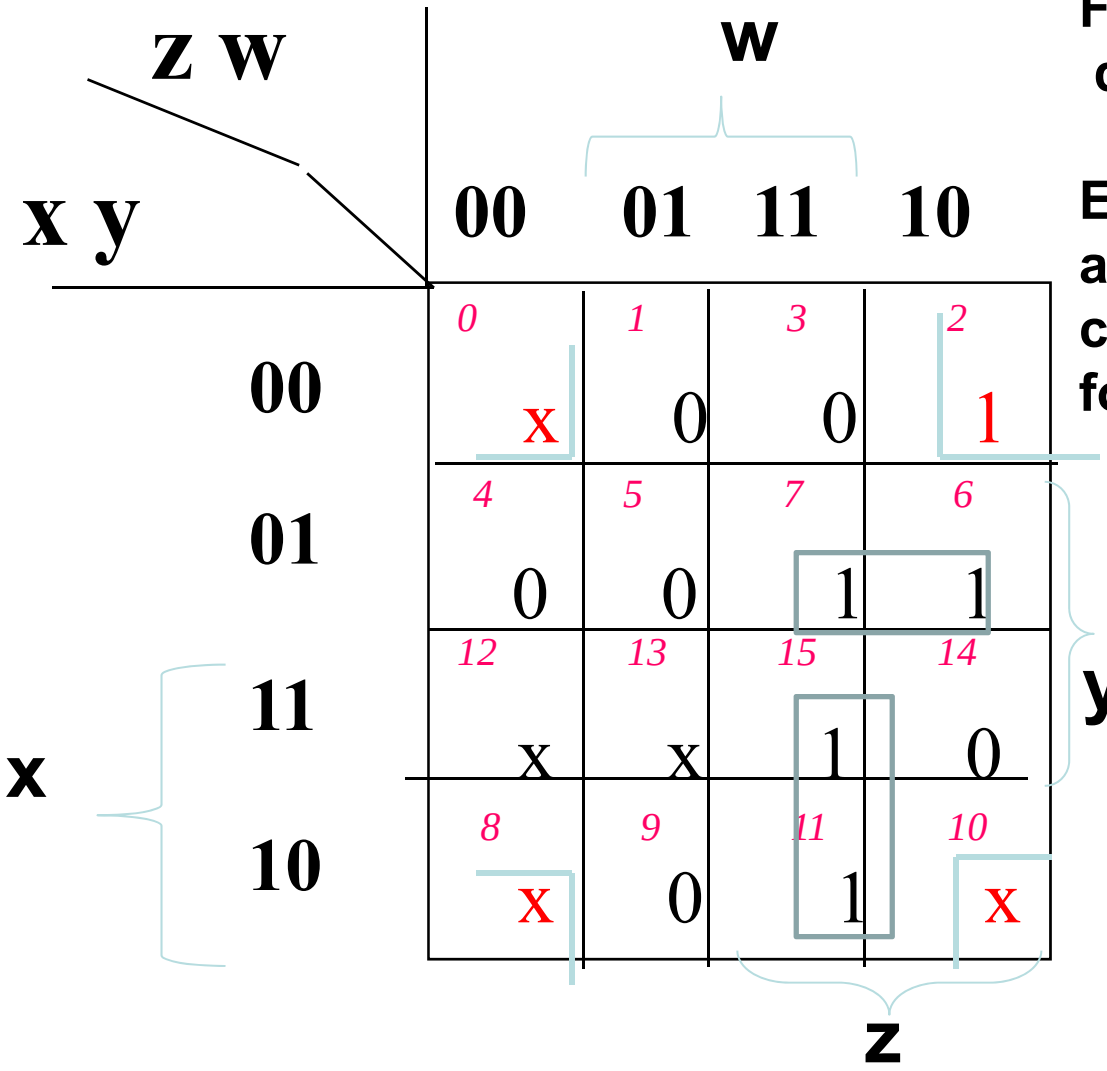


F(x, y, z, w) = ∑(2, 6, 7, 11, 15)
d(x, y, z, w) = ∑(0, 8, 10, 12, 13)
NO EPI's!
If start by selecting x'zw', a full cover of the function is given by:

2. x'zw'
4. yzw
7. xzw

F_{min1}=x'zw'+yzw+xzw

Q4(b) Minimal SoP



F(x, y, z, w) = ∑(2, 6, 7, 11, 15)
d(x, y, z, w) = ∑(0, 8, 10, 12, 13)

Else, if selecting first y'w', another set of PI's that fully covers the function can be found:

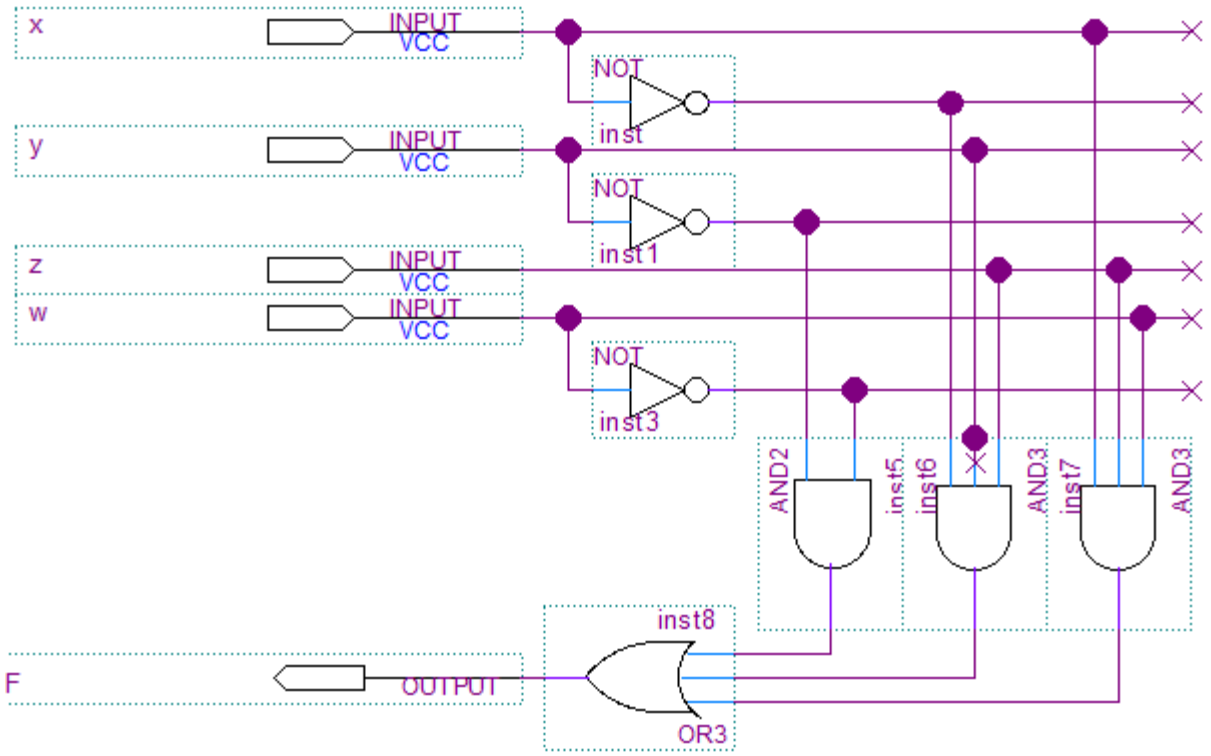
1. y'w' 3. x'yz 6. xzw

F_{min2} = y'w'+x'yz+xzw

Other minimized functions can be generated from the same set of PI's

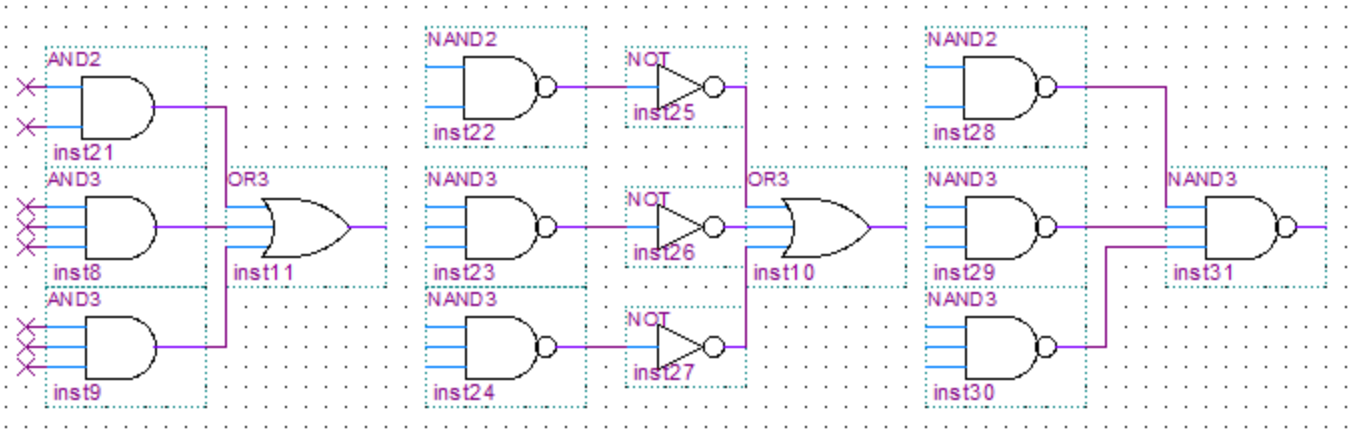
Q4(c)

- implement your minimized function with NAND gates only and draw the logic diagram of your circuit
- $F_{min2} = y'w'+x'yz+xzw$

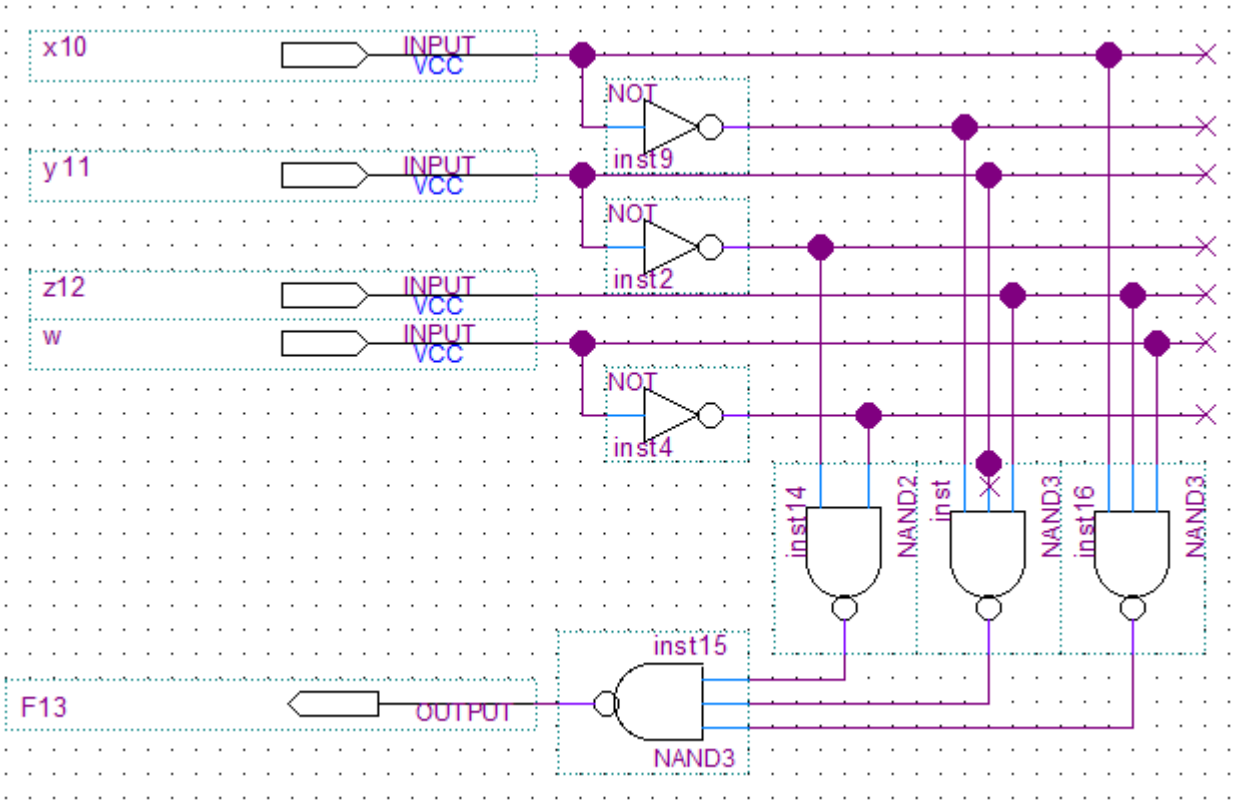


Q4(c)

- Circuit Transformation



Q4


x10
INPUT
VCC
y11
INPUT
VCC
z12
INPUT
VCC
w
INPUT
VCC
NOT
inst9
NOT
inst2
NOT
inst4
inst14
NAND2
inst
NAND3
inst16
NAND3
inst15
NAND3
F13
OUTPUT

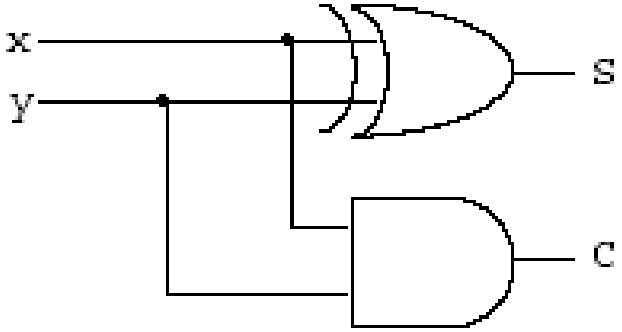
Combinational Circuits

From the truth table write the Boolean function outputs for the sum S and the carry out C:

S=x`y +xy` (Exclusive OR)

C=xy (AND)

(Logic diagram)



(Truth table for half-adder)			
x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Boolean Theorems to Simplify Expressions

- Theorems of Boolean Algebra can simplify Boolean expressions
  - e.g., full adder's carry-out function (same rules apply to any function)

A	B	Cin	S	Cout
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

Cout = A' B Cin + A B' Cin + A B Cin' + A B Cin
= A' B Cin + A B' Cin + A B Cin' + A B Cin + A B Cin
= A' B Cin + A B Cin + A B' Cin + A B Cin' + A B Cin
= (A' + A) B Cin + A B' Cin + A B Cin' + A B Cin
= (1) B Cin + A B' Cin + A B Cin' + A B Cin
= B Cin + A B' Cin + A B Cin' + A B Cin + A B Cin
= B Cin + A B' Cin + A B Cin + A B Cin' + A B Cin
= B Cin + A (B' + B) Cin + A B Cin' + A B Cin
= B Cin + A (1) Cin + A B Cin' + A B Cin
= B Cin + A Cin + A B (Cin' + Cin)
= B Cin + A Cin + A B (1)
= B Cin + A Cin + A B