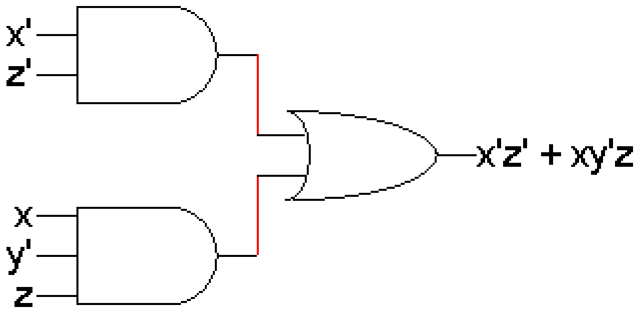
Karnaugh Maps

- Applications of Boolean logic to circuit design
  - The basic Boolean operations are AND, OR and NOT
  - These operations can be combined to form complex expressions, which can also be directly translated into a hardware circuit
  - Boolean algebra helps us simplify expressions and circuits
- Karnaugh Map: A graphical technique for simplifying an expression into a minimal sum of products (MSP) form:
  - There are a minimal number of product terms in the expression
  - Each term has a minimal number of literals
- Circuit-wise, this leads to a *minimal* two-level implementation



Review: Minterm

- A product term in which all the variables appear exactly once, either complemented or uncomplemented, is called a minterm
- A minterm represents exacly one combination of the binary variables in a truth table. It has the value of 1 for that combination and 0 for the others

X	Y	Z	Product Term	Symbol	m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7
0	0	0	$\overline{X}\,\overline{Y}\,\overline{Z}$	m_0	1	0	0	0	0	0	0	0
0	0	1	$\overline{X}\,\overline{Y}Z$	m_1	0	1	0	0	0	0	0	0
0	1	0	$\overline{X}Y\overline{Z}$	m_2	0	0	1	0	0	0	0	0
0	1	1	$\overline{X}YZ$	m_3	0	0	0	1	0	0	0	0
1	0	0	$X\overline{Y}\,\overline{Z}$	m_4	0	0	0	0	1	0	0	0
1	0	1	$X\overline{Y}Z$	m_5	0	0	0	0	0	1	0	0
1	1	0	$XY\overline{Z}$	m_6	0	0	0	0	0	0	1	0
1	1	1	XYZ	m_7	0	0	0	0	0	0	0	1

Table 2-6 Minterms for Three Variables

Review: Maxterm

- A sum term in which all the variables appear exactly once, either complemented or uncomplemented, is called a maxterm
- A maxterm represents exacly one combination of the binary variables in a truth table. It has the value of 0 for that combination and 1 for the others

X	Y	Z	Sum Term	Symbol	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7
0	0	0	$X+Y+Z$	M_0	0	1	1	1	1	1	1	1
0	0	1	$X+Y+\overline{Z}$	M_1	1	0	1	1	1	1	1	1
0	1	0	$X+\overline{Y}+Z$	M_2	1	1	0	1	1	1	1	1
0	1	1	$X+\overline{Y}+\overline{Z}$	M_3	1	1	1	0	1	1	1	1
1	0	0	$\overline{X}+Y+Z$	M_4	1	1	1	1	0	1	1	1
1	0	1	$\overline{X}+Y+\overline{Z}$	M_5	1	1	1	1	1	0	1	1
1	1	0	$\overline{X}+\overline{Y}+Z$	M_6	1	1	1	1	1	1	0	1
1	1	1	$\overline{X}+\overline{Y}+\overline{Z}$	M_7	1	1	1	1	1	1	1	0

Table 2-7 Maxterms for Three Variables

- A minterm and maxterm with the same subscript are the complements of each other, i.e., $M_j = m'_j$

Review: Sum of Minterms

- A Boolean function can be represented algebraically from a given truth table by forming the logical sum of all the minterms that produce a 1 in the function. This expression is called a sum of minterms

(a)	X	Y	Z	F	$\overline{F}$
	0	0	0	1	0
	0	0	1	0	1
	0	1	0	1	0
	0	1	1	0	1
	1	0	0	0	1
	1	0	1	1	0
	1	1	0	0	1
	1	1	1	1	0

$$F = X'Y'Z' + X'YZ' + XY'Z + XYZ$$

$$= m_0 + m_2 + m_5 + m_7$$

$$F(X,Y,Z) = \Sigma m(0,2,5,7)$$

Review: Product of Maxterms

- A Boolean function can be represented algebraically from a given truth table by forming the logical product of all the maxterms that produce a 0 in the function. This expression is called a product of maxterms

(a) X	Y	Z	F	$\overline{F}$
0	0	0	1	0
0	0	1	0	1
0	1	0	1	0
0	1	1	0	1
1	0	0	0	1
1	0	1	1	0
1	1	0	0	1
1	1	1	1	0

$F= (X+Y+Z')(X+Y'+Z')(X'+Y+Z)(X'+Y'+Z)$

$= M_1 . M_3 . M_4 . M_6$

$F(X,Y,Z)= \prod M(1,3,4,6)$

- To convert a Boolean function F from SoM to PoM:
  - Find F' in SoM form
  - Find F= (F')' in PoM form

Review: Important Properties of Minterms

- There are 2^n minterms for n Boolean variables. These minterms can be evaluated from the binary numbers from 0 to 2^n-1
- Any Boolean function can be expressed as a logical sum of minterms
- The complement of a function contains those minterms not included in the original function

$$F(X,Y,Z)= \Sigma m(0,2,5,7) \Rightarrow F'(X,Y,Z)= \Sigma m(1,3,4,6)$$

- A function that includes all the 2^n minterms is equal to logic 1

$$G(X,Y)= \Sigma m(0,1,2,3)= 1$$

Review: Sum-of-Products

- The sum-of-minterms form is a standard algebraic expression that is obtained from a truth table
- When we simplify a function in SoM form by reducing the number of product terms or by reducing the number of literals in the terms, the simplified expression is said to be in Sum-of-Products form
- Sum-of-Products expression can be implemented using a two-level circuit

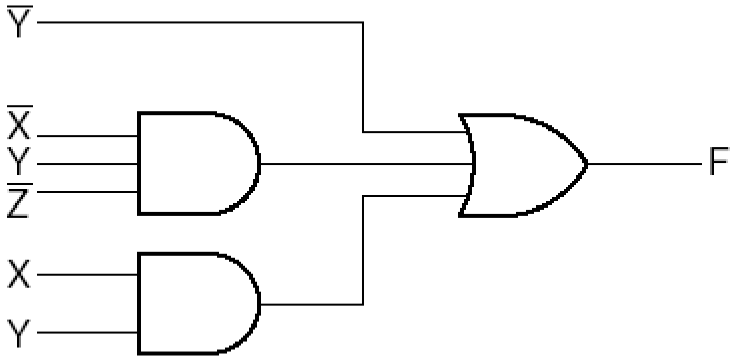


Fig. 2-5 Sum-of-Products Implementation

$$F = \Sigma m(0,1,2,3,4,5,7) \qquad \text{(SoM)}$$

$$= Y' + X'YZ' + XY \qquad \text{(SoP)}$$

Review: Product-of-Sums

- The product-of-maxterms form is a standard algebraic expression that is obtained from a truth table
- When we simplify a function in PoM form by reducing the number of sum terms or by reducing the number of literals in the terms, the simplified expression is said to be in Product-of-Sums form
- Product-of-Sums expression can be implemented using a two-level circuit

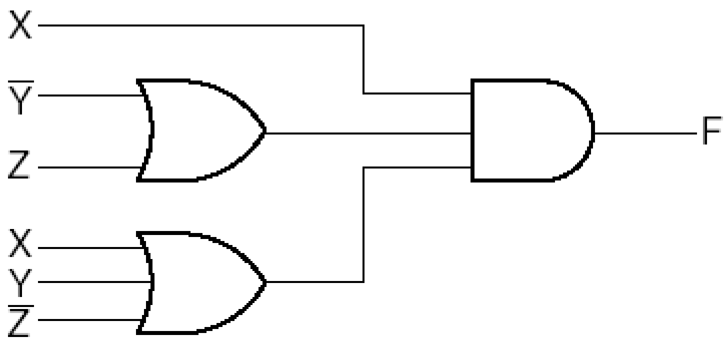


Fig. 2-7 Product-of-Sums Implementation

$F = \prod M(0,2,3,4,5,6)$ (PoM)

$= X(Y' + Z)(X + Y + Z')$ (PoS)

Re-arranging the Truth Table

- A two-variable function has four possible minterms. We can re-arrange these minterms into a Karnaugh map

x	y	minterm
0	0	$x'y'$
0	1	$x'y$
1	0	xy'
1	1	xy

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

- Now we can easily see which minterms contain common literals
  - Minterms on the left and right sides contain y' and y respectively
  - Minterms in the top and bottom rows contain x' and x respectively

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

	y'	y
x'	$x'y'$	$x'y$
x	xy'	xy

Karnaugh Map Simplifications

- Imagine a two-variable sum of minterms:

$$x'y' + x'y$$

- Both of these minterms appear in the top row of a Karnaugh map, which means that they both contain the literal x'

		y
	$x'y'$	$x'y$
x	xy'	xy

- What happens if you simplify this expression using Boolean algebra?

$$\begin{aligned} x'y' + x'y &= x'(y' + y) && [\text{ Distributive }] \\ &= x' \bullet 1 && [\ y + y' = 1\ ] \\ &= x' && [\ x \bullet 1 = x\ ] \end{aligned}$$

More Two-Variable Examples

- Another example expression is $x'y + xy$
  - Both minterms appear in the right side, where y is uncomplemented
  - Thus, we can reduce $x'y + xy$ to just y

		y
	$x'y'$	$x'y$
x	xy'	xy

- How about $x'y' + x'y + xy$?
  - We have $x'y' + x'y$ in the top row, corresponding to x'
  - There's also $x'y + xy$ in the right side, corresponding to y
  - This whole expression can be reduced to $x' + y$

		y
	$x'y'$	$x'y$
x	xy'	xy

A Three-Variable Karnaugh Map

- For a three-variable expression with inputs x, y, z, the arrangement of minterms is more tricky:

YZ

		00	01	11	10
X	0	x'y'z'	x'y'z	x'yz	x'yz'
	1	xy'z'	xy'z	xyz	xyz'

YZ

		00	01	11	10
X	0	m_0	m_1	m_3	m_2
	1	m_4	m_5	m_7	m_6

- Another way to label the K-map (use whichever you like):

y

	x'y'z'	x'y'z	x'yz	x'yz'
X	xy'z'	xy'z	xyz	xyz'

Z

y

	m_0	m_1	m_3	m_2
X	m_4	m_5	m_7	m_6

Z

Why the funny ordering?

- With this ordering, any group of 2, 4 or 8 adjacent squares on the map contains common literals that can be factored out

			y	
	$x'y'z'$	$x'y'z$	$x'yz$	$x'yz'$
X	$xy'z'$	$xy'z$	xyz	xyz'
		Z		

$$\begin{aligned} & x'y'z + x'yz \\ = \; & x'z(y' + y) \\ = \; & x'z \bullet 1 \\ = \; & x'z \end{aligned}$$

- "Adjacency" includes wrapping around the left and right sides:

			y	
	$x'y'z'$	$x'y'z$	$x'yz$	$x'yz'$
X	$xy'z'$	$xy'z$	xyz	xyz'
		Z		

$$\begin{aligned} & x'y'z' + xy'z' + x'yz' + xyz' \\ = \; & z'(x'y' + xy' + x'y + xy) \\ = \; & z'(y'(x' + x) + y(x' + x)) \\ = \; & z'(y'+y) \\ = \; & z' \end{aligned}$$

- We'll use this property of adjacent squares to do our simplifications.

Example K-map Simplification

- Let's consider simplifying $f(x,y,z) = xy + y'z + xz$
- First, you should convert the expression into a sum of minterms form, if it's not already
  - The easiest way to do this is to make a truth table for the function, and then read off the minterms
  - You can either write out the literals or use the minterm shorthand
- Here is the truth table and sum of minterms for our example:

x	y	z	f(x,y,z)
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	1
1	1	1	1

$$\begin{aligned} f(x,y,z) &= x'y'z + xy'z + xyz' + xyz \\ &= m_1 + m_5 + m_6 + m_7 \end{aligned}$$

Unsimplifying Expressions

- You can also convert the expression to a sum of minterms with Boolean algebra
  - Apply the distributive law in reverse to add in missing variables.
  - Very few people actually do this, but it's occasionally useful.

$$
\begin{aligned}
xy + y'z + xz &= (xy \bullet 1) + (y'z \bullet 1) + (xz \bullet 1) \\
&= (xy \bullet (z' + z)) + (y'z \bullet (x' + x)) + (xz \bullet (y' + y)) \\
&= (xyz' + xyz) + (x'y'z + xy'z) + (xy'z + xyz) \\
&= xyz' + xyz + x'y'z + xy'z
\end{aligned}
$$

- In both cases, we're actually "unsimplifying" our example expression
  - The resulting expression is larger than the original one!
  - But having all the individual minterms makes it easy to combine them together with the K-map

Making the Example K-map

- Next up is drawing and filling in the K-map
  - Put 1s in the map for each minterm, and 0s in the other squares
  - You can use either the minterm products or the shorthand to show you where the 1s and 0s belong
- In our example, we can write $f(x,y,z)$ in two equivalent ways

$$f(x,y,z) = x'y'z + xy'z + xyz' + xyz$$

			y	y
	$x'y'z'$	$x'y'z$	$x'yz$	$x'yz'$
X	$xy'z'$	$xy'z$	xyz	xyz'
		Z	Z	

$$f(x,y,z) = m_1 + m_5 + m_6 + m_7$$

			y	y
	m_0	m_1	m_3	m_2
X	m_4	m_5	m_7	m_6
		Z	Z	

- In either case, the resulting K-map is shown below

			y	y
	0	1	0	0
X	0	1	1	1
		Z	Z	

K-maps From Truth Tables

- You can also fill in the K-map directly from a truth table
  - The output in row *i* of the table goes into square m_i of the K-map
  - Remember that the rightmost columns of the K-map are "switched"

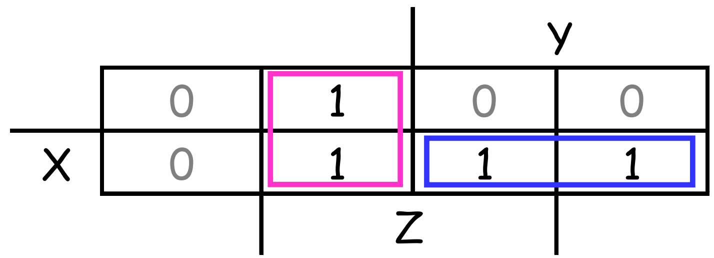
			y	y
	m_0	m_1	m_3	m_2
X	m_4	m_5	m_7	m_6
		Z	Z	

x	y	z	f(x,y,z)
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	1
1	1	1	1

			y	y
	0	1	0	0
X	0	1	1	1
		Z	Z	

Grouping the Minterms Together

- The most difficult step is grouping together all the 1s in the K-map
  - Make rectangles around groups of one, two, four or eight 1s
  - All of the 1s in the map should be included in at least one rectangle
  - Do *not* include any of the 0s



- Each group corresponds to one product term. For the simplest result:
  - Make as few rectangles as possible, to minimize the number of products in the final expression.
  - Make each rectangle as large as possible, to minimize the number of literals in each term.
  - It's all right for rectangles to overlap, if that makes them larger.

Reading the MSP from the K-map

- Finally, you can find the minimal SoP expression
  - Each rectangle corresponds to one product term
  - The product is determined by finding the common literals in that rectangle

			y	
	0	1	0	0
x	0	1	1	1
		z		

			y	
	$x'y'z'$	$x'y'z$	$x'yz$	$x'yz'$
x	$xy'z'$	$xy'z$	xyz	xyz'
		z		

- For our example, we find that $xy + y'z + xz = y'z + xy$. (This is one of the additional algebraic laws from last time.)

Practice K-map 1

- Simplify the sum of minterms $m_1 + m_3 + m_5 + m_6$

			y	
x				
		z		

			y	
	m_0	m_1	m_3	m_2
x	m_4	m_5	m_7	m_6
		z		

Solutions for Practice K-map 1

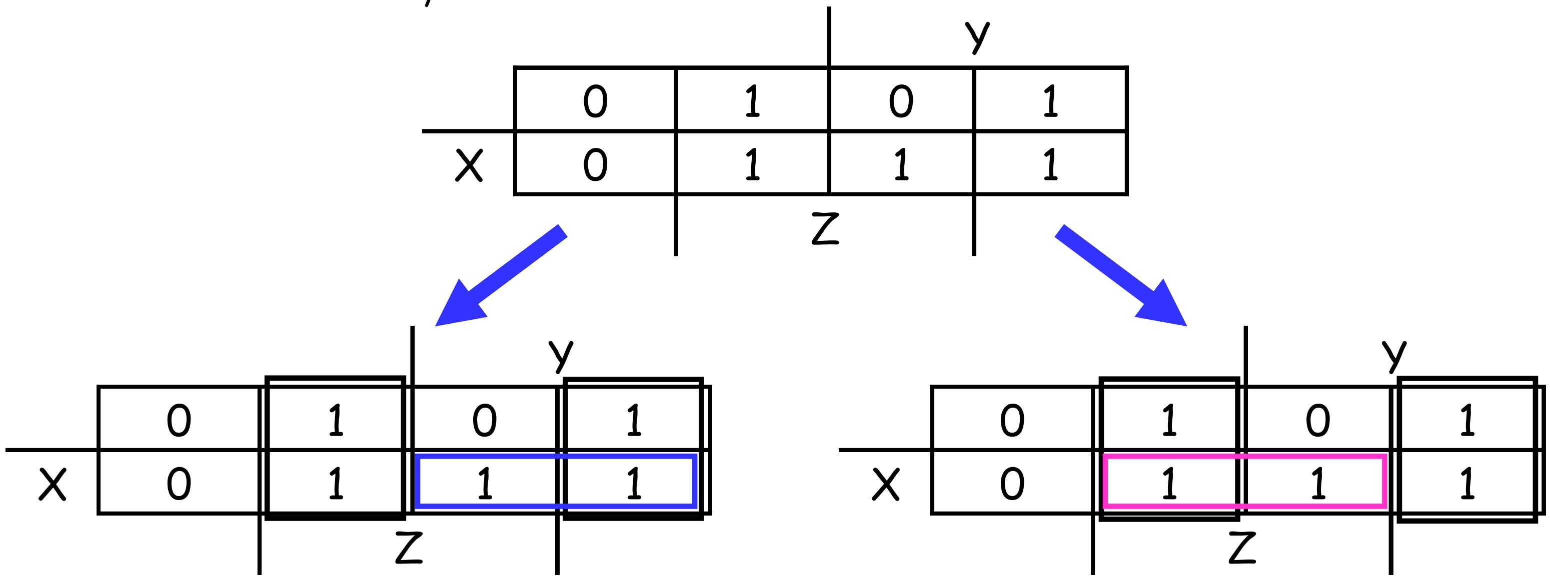
- Here is the filled in K-map, with all groups shown
  - The magenta and green groups overlap, which makes each of them as large as possible
  - Minterm m_6 is in a group all by its lonesome

				y
	0	1	1	0
x	0	1	0	1
		z		

- The final MSP here is $x'z + y'z + xyz'$

K-maps can be tricky!

- There may not necessarily be a *unique* MSP. The K-map below yields two valid and equivalent MSPs, because there are two possible ways to include minterm m_7

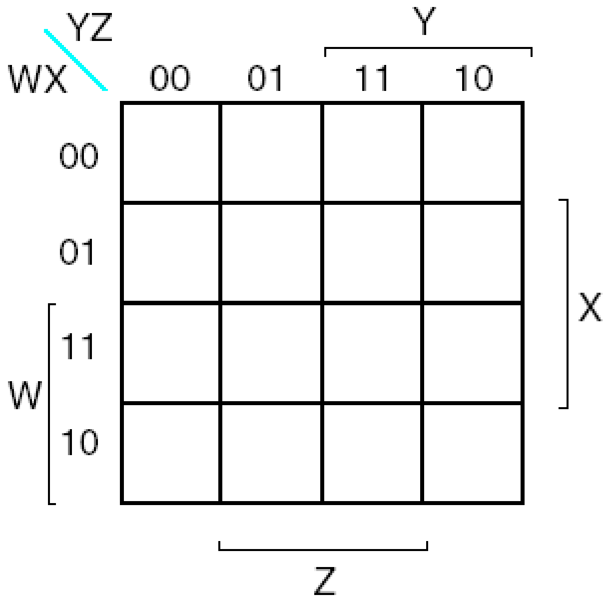


$y'z + yz' + xy$ $\qquad\qquad$ $y'z + yz' + xz$

- Remember that overlapping groups is possible, as shown above

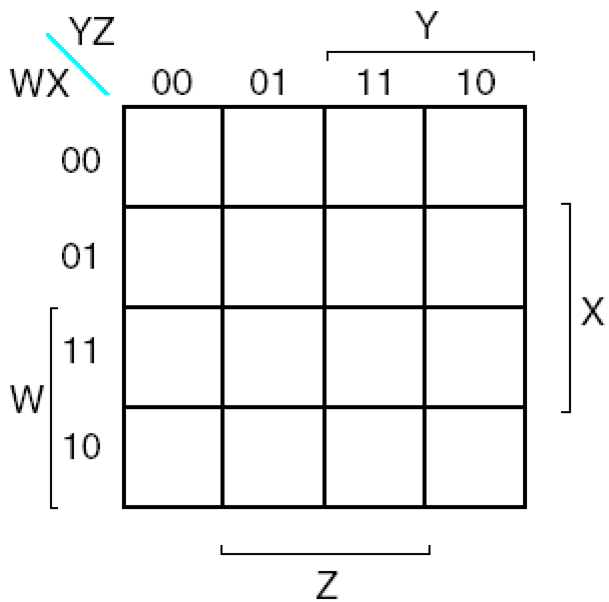
Four-variable K-maps

- We can do four-variable expressions too!
  - The minterms in the third and fourth columns, *and* in the third and fourth rows, are switched around.
  - Again, this ensures that adjacent squares have common literals



- Grouping minterms is similar to the three-variable case, but:
  - You can have rectangular groups of 1, 2, 4, 8 or 16 minterms
  - You can wrap around *all four* sides

Four-variable K-maps



			y	y	
	w'x'y'z'	w'x'y'z	w'x'yz	w'x'yz'	
	w'xy'z'	w'xy'z	w'xyz	w'xyz'	X
W	wxy'z'	wxy'z	wxyz	wxyz'	X
W	wx'y'z'	wx'y'z	wx'yz	wx'yz'	
		z	z		

			y	y	
	m_0	m_1	m_3	m_2	
	m_4	m_5	m_7	m_6	X
W	m_{12}	m_{13}	m_{15}	m_{14}	X
W	m_8	m_9	m_{11}	m_{10}	
		z	z		

Example: Simplify $m_0+m_2+m_5+m_8+m_{10}+m_{13}$

- The expression is already a sum of minterms, so here's the K-map:

1	0	0	1
0	1	0	0
0	1	0	0
1	0	0	1

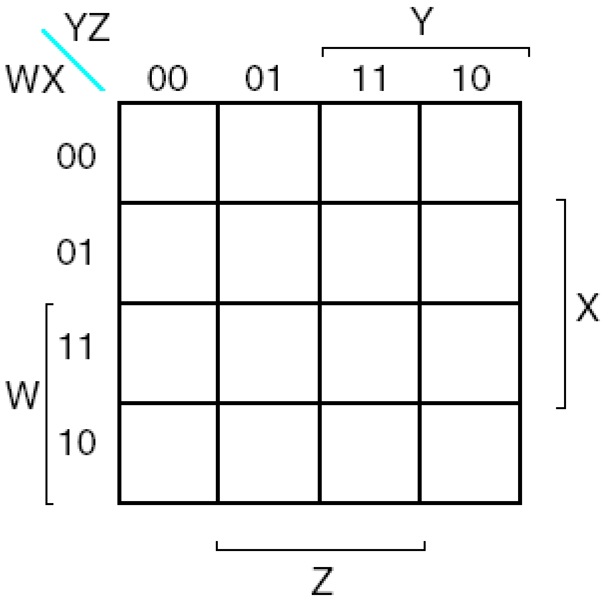
m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
m_{12}	m_{13}	m_{15}	m_{14}
m_8	m_9	m_{11}	m_{10}

- We can make the following groups, resulting in the MSP $x'z' + xy'z$

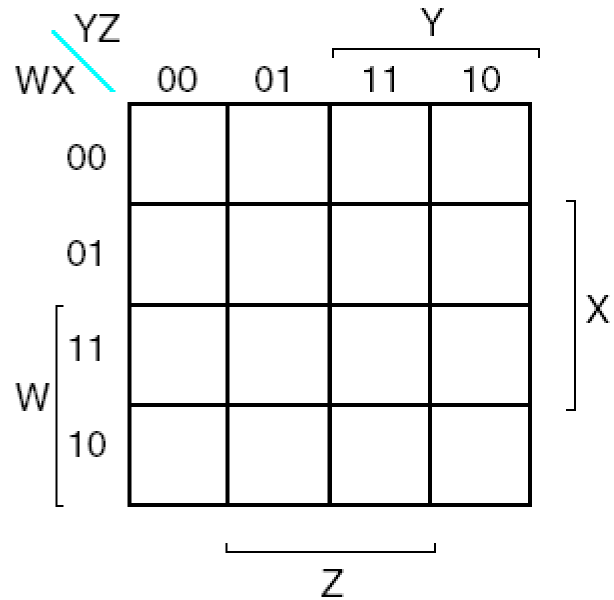
1	0	0	1
0	1	0	0
0	1	0	0
1	0	0	1

$w'x'y'z'$	$w'x'y'z$	$w'x'yz$	$w'x'yz'$
$w'xy'z'$	$w'xy'z$	$w'xyz$	$w'xyz'$
$wxy'z'$	$wxy'z$	$wxyz$	$wxyz'$
$wx'y'z'$	$wx'y'z$	$wx'yz$	$wx'yz'$

Five-variable K-maps



V= 0

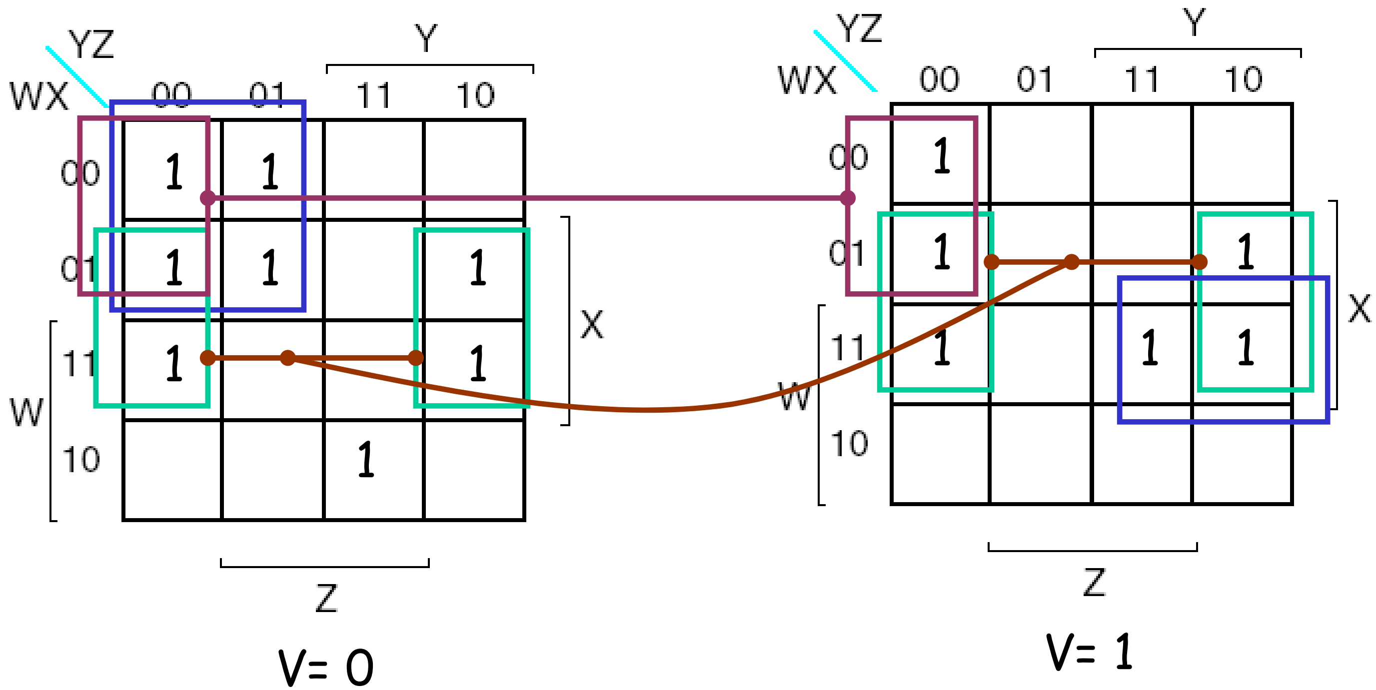


V= 1

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
m_{12}	m_{13}	m_{15}	m_{14}
m_8	m_9	m_{11}	m_{10}

m_{16}	m_{17}	m_{19}	m_8
m_{20}	m_{21}	m_{23}	m_{22}
m_{28}	m_{29}	m_{31}	m_{30}
m_{24}	m_{25}	m_{27}	m_{26}

Simplify f(V,W,X,Y,Z)=Σm(0,1,4,5,6,11,12,14,16,20,22,28,30,31)

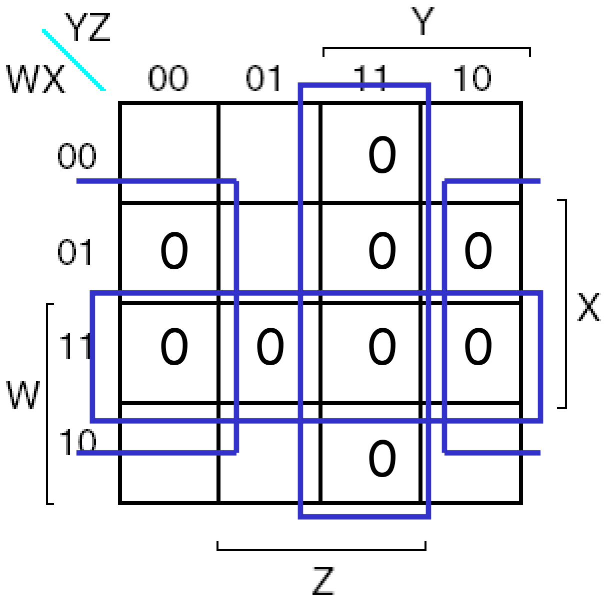


$f = XZ'$ $\Sigma m(4,6,12,14,20,22,28,30)$
$+ V'W'Y'$ $\Sigma m(0,1,4,5)$
$+ W'Y'Z'$ $\Sigma m(0,4,16,20)$
$+ VWXY$ $\Sigma m(30,31)$
$+ V'WX'YZ$ $m11$

PoS Optimization from SoP

$F(W,X,Y,Z) = \Sigma m(0,1,2,5,8,9,10)$

$= \prod M(3,4,6,7,11,12,13,14,15)$

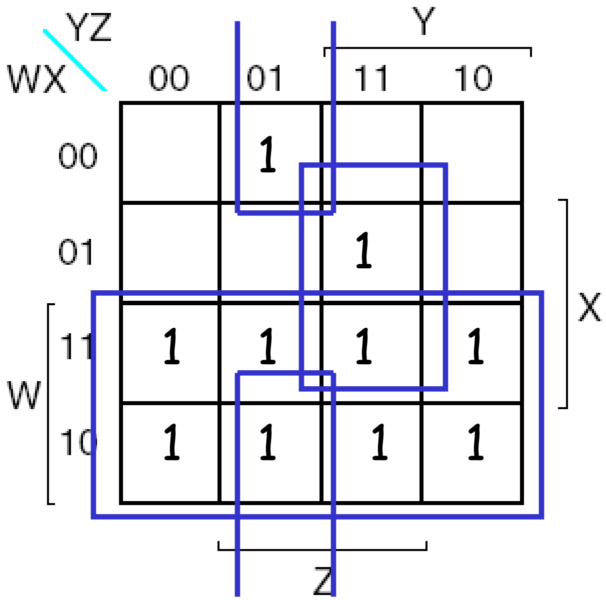


$F(W,X,Y,Z) = (W' + X')(Y' + Z')(X' + Z)$

SoP Optimization from PoS

$F(W,X,Y,Z) = \prod M(0,2,3,4,5,6)$

$= \sum m(1,7,8,9,10,11,12,13,14,15)$



$F(W,X,Y,Z) = W + XYZ + X'Y'Z$

I don't care!

- You don't always need all 2^n input combinations in an n-variable function
  - If you can guarantee that certain input combinations never occur
  - If some outputs aren't used in the rest of the circuit

- We mark don't-care outputs in truth tables and K-maps with Xs.

x	y	z	f(x,y,z)
0	0	0	0
0	0	1	1
0	1	0	X
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	X
1	1	1	1

- Within a K-map, each X can be considered as either 0 or 1. You should pick the interpretation that allows for the most simplification.

Practice K-map 3

- Find a MSP for

$$f(w,x,y,z) = \Sigma m(0,2,4,5,8,14,15),\ d(w,x,y,z) = \Sigma m(7,10,13)$$

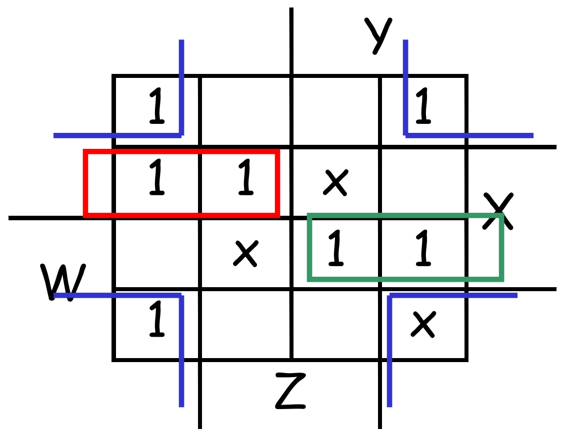
This notation means that input combinations wxyz = 0111, 1010 and 1101 (corresponding to minterms m_7, m_{10} and m_{13}) are unused.

			y	y	
	1	0	0	1	
	1	1	x	0	x
w	0	x	1	1	x
w	1	0	0	x	
		z	z		

Solutions for Practice K-map 3

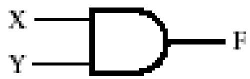
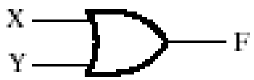
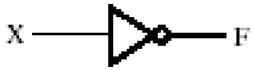
- Find a MSP for:

$$f(w,x,y,z) = \Sigma m(0,2,4,5,8,14,15),\ d(w,x,y,z) = \Sigma m(7,10,13)$$

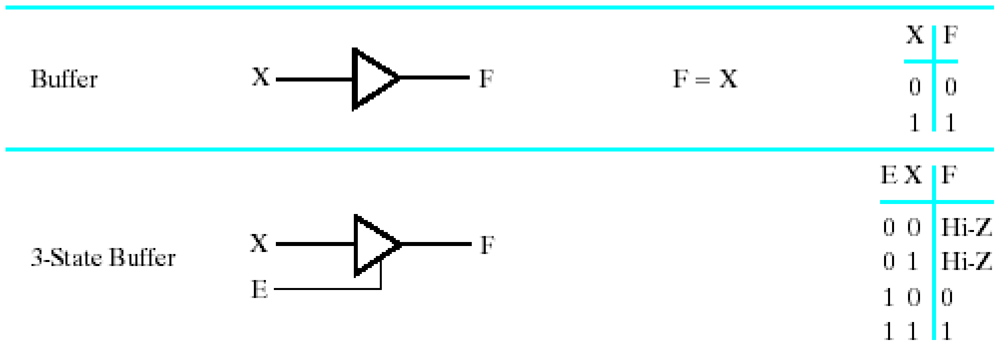


$$f(w,x,y,z) = x'z' + w'xy' + wxy$$

AND, OR, and NOT

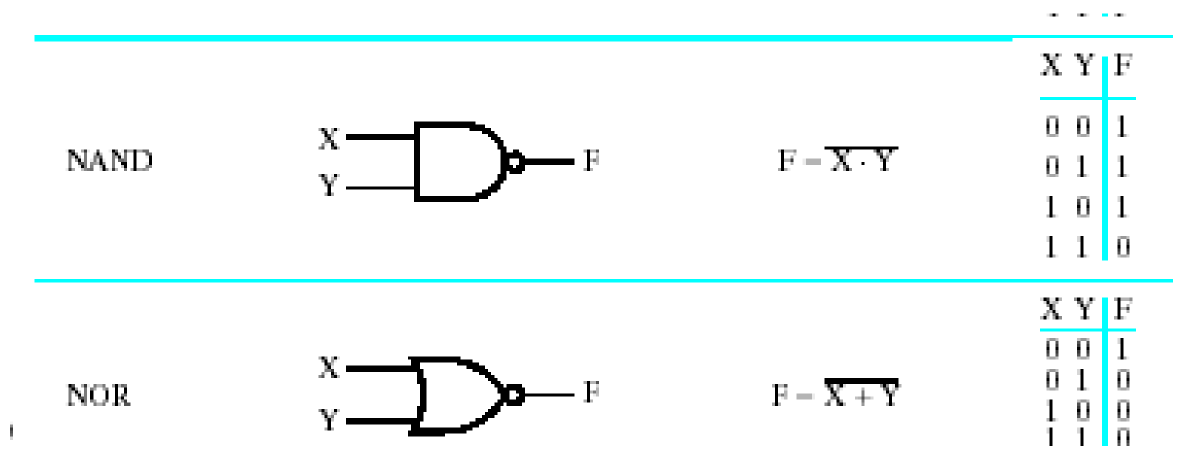
Name	Graphics Symbols Distinctive shape	Algebraic equation	Truth table
AND		$F = XY$	X Y | F 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR		$F = X + Y$	X Y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
NOT (inverter)		$F = \overline{X}$	X | F 0 | 1 1 | 0

Buffer and 3-State Buffer



- Buffer is used to amplify an electrical signal
  - Reconstructing the signal
  - More gates to be attached to the output
- Three state buffer
  - E (Enable): Controls the output
  - Hi-Z: High impedance

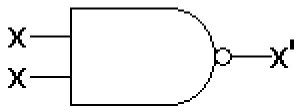
NAND and NOR



- NAND: Not AND, NOR: Not OR
- Both NAND and NOR are universal gates
- Universal gate: A gate that alone can be used to implement all Boolean functions
- It is sufficient to show that NAND (NOR) can be used to implement AND, OR, and NOT operations

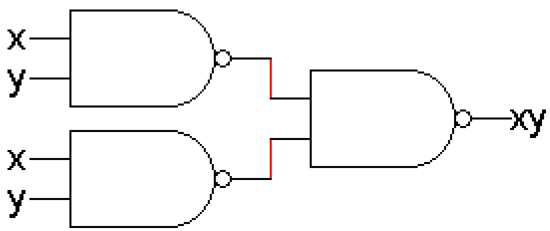
NANDs are special!

- The NAND gate is universal: it can replace all other gates!
  - NOT



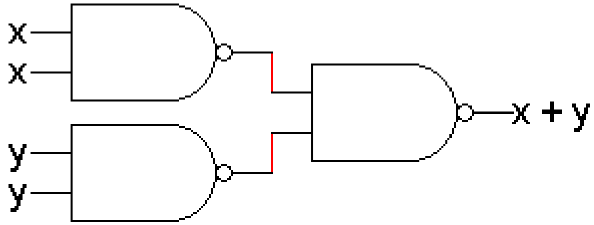
$(xx)' = x'$ [because $xx = x$]

  - AND



$((xy)'\,(xy)')' = xy$ [from NOT above]

  - OR

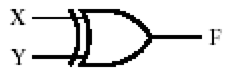
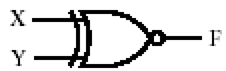


$((xx)'\,(yy)')' = (x'\,y')'$ [$xx = x$, and $yy = y$]

$= x + y$ [DeMorgan's law]

XOR and XNOR

Graphics Symbols

Name	Distinctive shape symbol	Algebraic equation	Truth table
Exclusive-OR (XOR)		$F = X\overline{Y} + \overline{X}Y$ $= X \oplus Y$	X Y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0
Exclusive-NOR (XNOR)		$F = XY + \overline{X}\,\overline{Y}$ $= \overline{X \oplus Y}$	X Y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1

- Exclusive-OR (XOR): $X \oplus Y = XY' + X'Y$
- Exclusive-NOR (XNOR): $(X \oplus Y)' = XY + X'Y'$

$X \oplus 0 = X$ $\qquad X \oplus 1 = X'$

$X \oplus X = 0$ $\qquad X \oplus X' = 1$

$X \oplus Y' = (X \oplus Y)'$ $\qquad X' \oplus Y = (X \oplus Y)'$

$X \oplus Y = Y \oplus X$

$(X \oplus Y) \oplus Z = X \oplus (Y \oplus Z) = X \oplus Y \oplus Z$

More on XOR

- The general XOR function is true when an odd number of its arguments are true

- For example, we can use Boolean algebra to simplify a three-input XOR to the following expression and truth table.

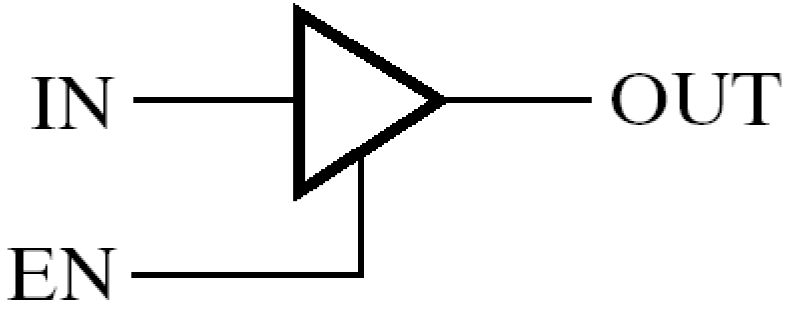
$x \oplus (y \oplus z)$	
$= x \oplus (y'z + yz')$	[Definition of XOR]
$= x'(y'z + yz') + x(y'z + yz')'$	[Definition of XOR]
$= x'y'z + x'yz' + x(y'z + yz')'$	[Distributive]
$= x'y'z + x'yz' + x((y'z)' (yz')')$	[DeMorgan's]
$= x'y'z + x'yz' + x((y + z')(y' + z))$	[DeMorgan's]
$= x'y'z + x'yz' + x(yz + y'z')$	[Distributive]
$= x'y'z + x'yz' + xyz + xy'z'$	[Distributive]

x	y	z	$x \oplus y \oplus z$
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

High-Impedance Outputs

- Gates with only output values logic 0 and logic 1
  - The output is connected to either Vcc or Gnd
- A third output value: High-Impedance (Hi-Z, Z, or z)
  - The output behaves as an open-circuit, i.e., it appears to be disconnected
- Gates with Hi-Z output values can have their outputs connected together if no two gates drive the line at the same time to opposite 0 and 1 values
- Gates with only logic 0 and logic 1 outputs cannot have their outputs connected together

Three-State Buffers

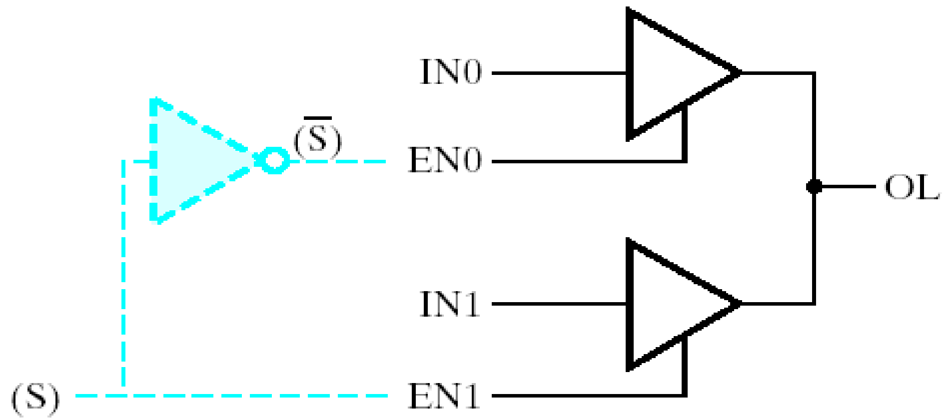


(a) Logic symbol

EN	IN	OUT
0	X	Hi-Z
1	0	0
1	1	1

(b) Truth table

Three-State Buffers

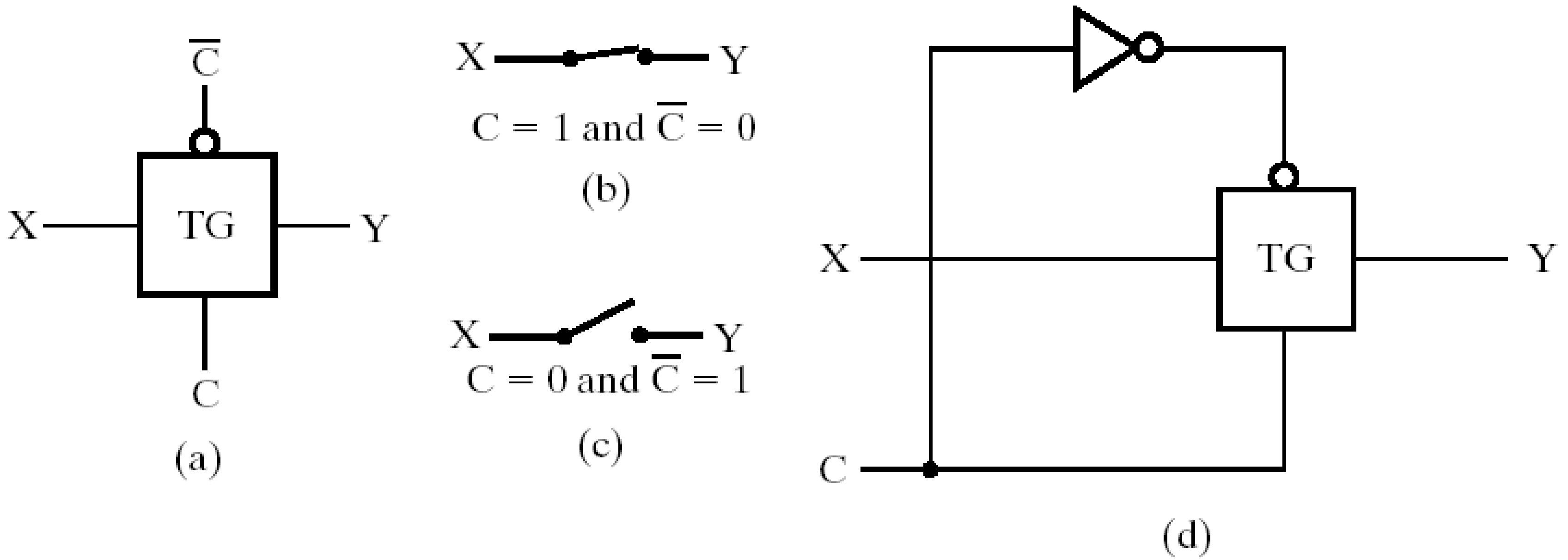


(a) Logic Diagram

EN1	EN0	IN1	IN0	OL
0	0	X	X	Hi-Z
(S) 0	$(\bar{S})$ 1	X	0	0
0	1	X	1	1
1	0	0	X	0
1	0	1	X	1
1	1	0	0	0
1	1	1	1	1
1	1	0	1	
1	1	1	0	

(b) Truth table

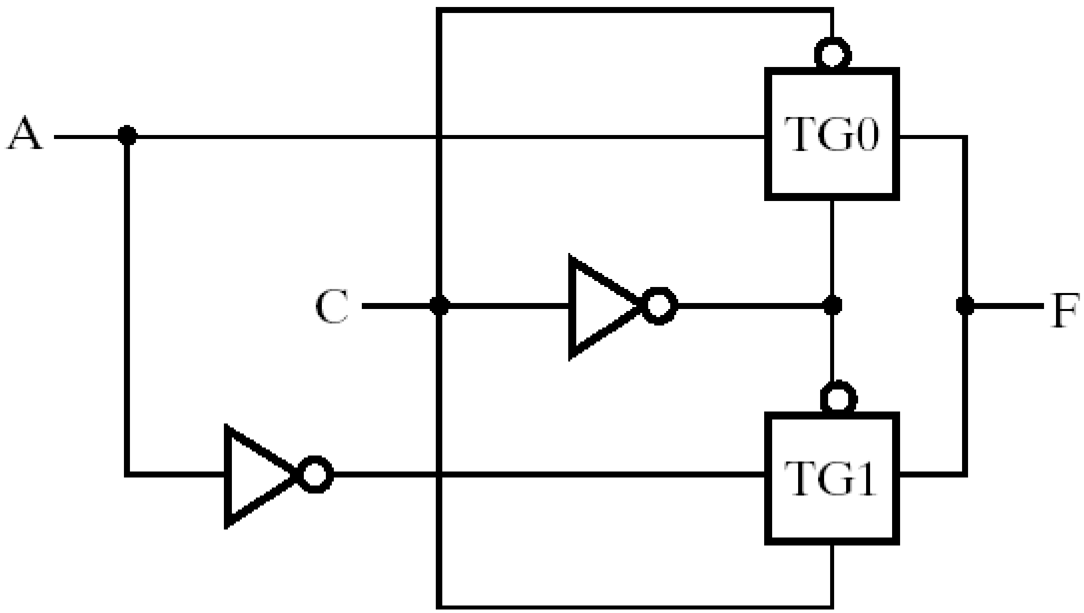
Transmission Gate



If C= 1 (C'= 0) => Y = X

If C= 0 (C'= 1) => Y = Hi-Z

Transmission Gate XOR



(a)

A	C	TG1	TG0	F
0	0	No path	Path	0
0	1	Path	No path	1
1	0	No path	Path	1
1	1	Path	No path	0

(b)

K-map Summary

- K-maps are an alternative to algebra for simplifying expressions
  - The result is a *minimal sum of products*, which leads to a minimal two-level circuit
  - It's easy to handle don't-care conditions
  - K-maps are really only good for manual simplification of small expressions...
- Things to keep in mind:
  - Remember the correct order of minterms on the K-map
  - When grouping, you can wrap around all sides of the K-map, and your groups can overlap
  - Make as few rectangles as possible, but make each of them as large as possible. This leads to fewer, but simpler, product terms
  - There may be more than one valid solution