Digital Logic Circuit (SE273 – Fall 2020)

Lecture 3: Boolean Algebra

Jaesok Yu, Ph.D. (jaesok.yu@dgist.ac.kr)

Assistant Professor

Department of Robotics Engineering, DGIST

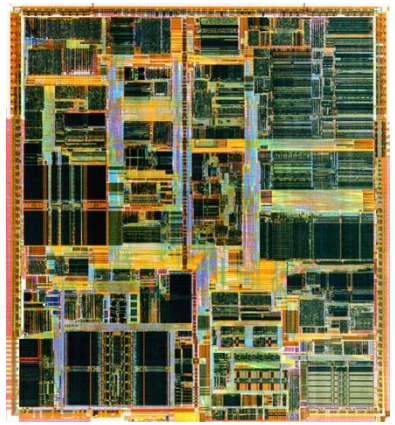
Goal

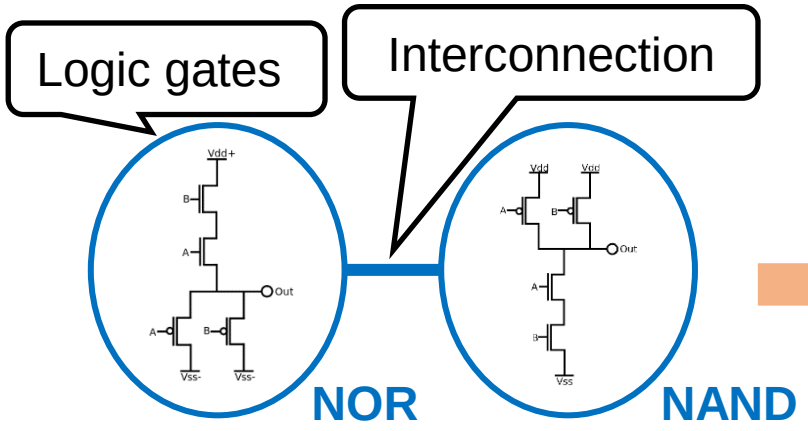
- Learn Boolean algebra and logic gates
  - Basic theory and properties of Boolean algebra
  - Digital logic gates
- Apply simplification of Boolean functions
  - Karnaugh map (K-map)
  - Sum-of-products/product-of-sums simplification

Binary Logic

- Digital circuits manipulates binary information
  - Composed of transistors and interconnections
- Basic circuit is referred to as "**logic gate**"
  - Each gate performs a specific logical operation

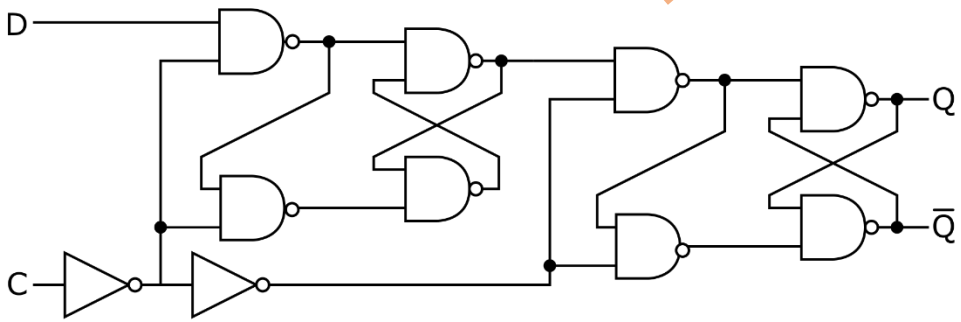
Digital circuits





Part of digital circuit

D Flip-flop circuit

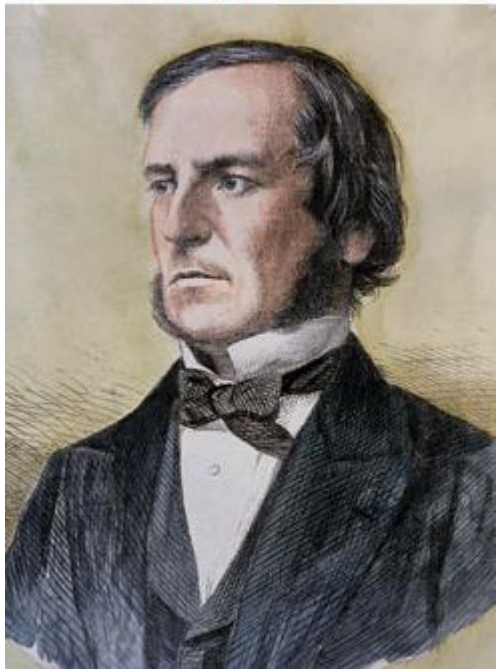


Logic Gates and Boolean Algebra

- Why do we need an abstraction of a logic gate?
  - A designer need not be concerned with the internal electronics
  - Only their external logic properties are important

- To describe operational properties of digital circuits,
  - We introduce a mathematical notation to analyze/design circuits
  - The binary logic system is called **Boolean algebras**

George Boole

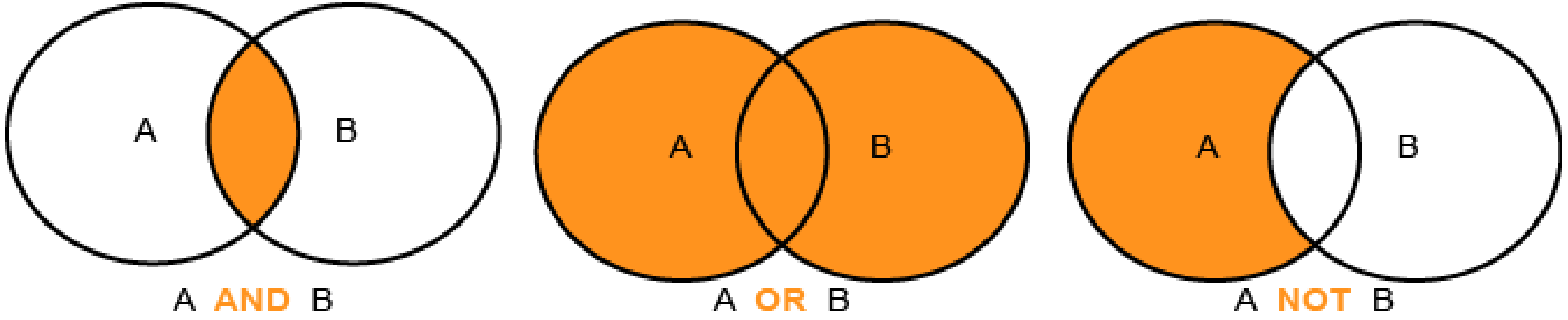


Published a book in 1854
on mathematical theory of logics

Binary Logic

- It deals with binary variables (A,…,X,Y,Z) that take on two discrete values (0 or 1)
- Three basic logical operations

Boolean AND, OR, and NOT



Truth Table

- The definition of the logic operation may be listed in compact form (truth table)
  - A table of combinations of the binary variables showing the relation btw the values that the variables take on and its result

Truth Tables for the Three Basic Logical Operations

AND		
X	Y	$Z = X \cdot Y$
0	0	0
0	1	0
1	0	0
1	1	1

OR		
X	Y	$Z = X + Y$
0	0	0
0	1	1
1	0	1
1	1	1

NOT	
X	$Z = \overline{X}$
0	1
1	0

Boolean Algebra - Binary Logic

- Three basic logical operations

Operation:	AND (product) of two inputs	OR (sum) of two inputs	NOT (complement) on one input
Expression:	$X \cdot Y$, $X \& Y$	$X + Y$, $X \mid Y$	X', $\bar{X}$, $\sim X$

Truth table:

AND

X	Y	$Z = X \cdot Y$
0	0	0
0	1	0
1	0	0
1	1	1

OR

X	Y	$Z = X + Y$
0	0	0
0	1	1
1	0	1
1	1	1

NOT

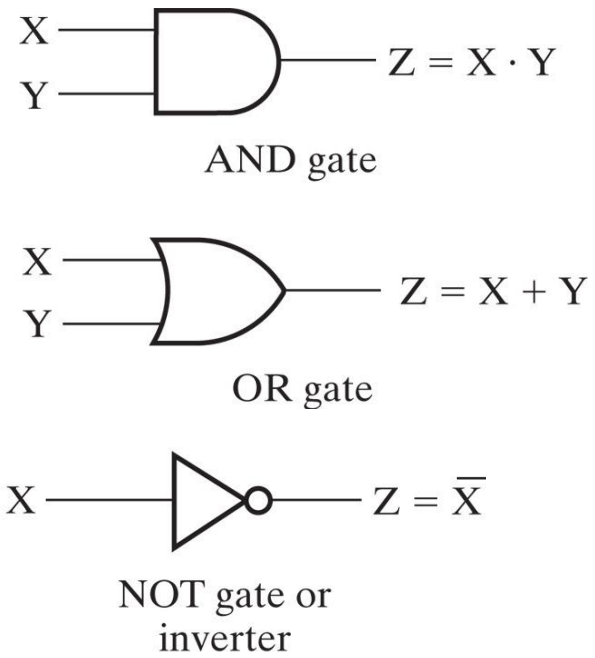
X	$Z = \bar{X}$
0	1
1	0

Do not confuse with binary arithmetic: ($1_2 + 1_2 = 10_2$)

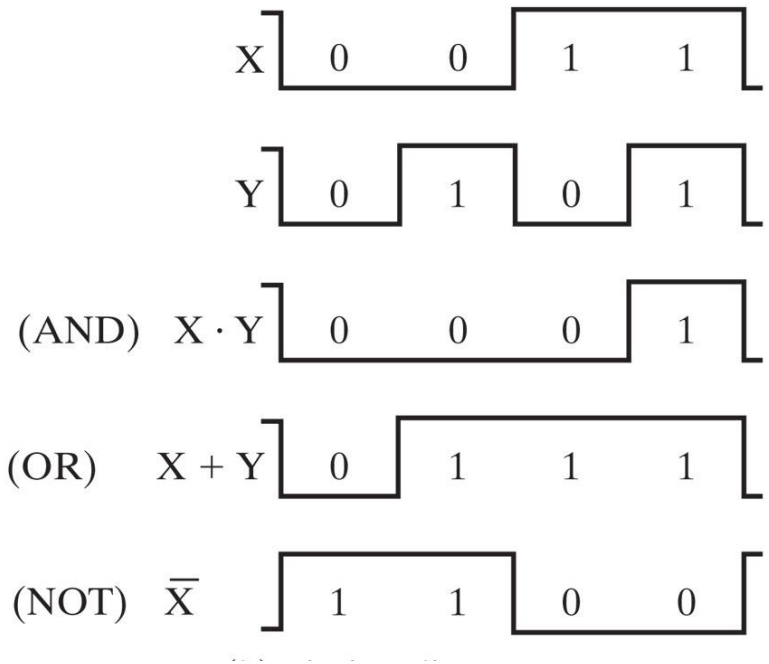
Logic Gates

- They are electronic circuits that operate on one or more input signals to produce an output signal

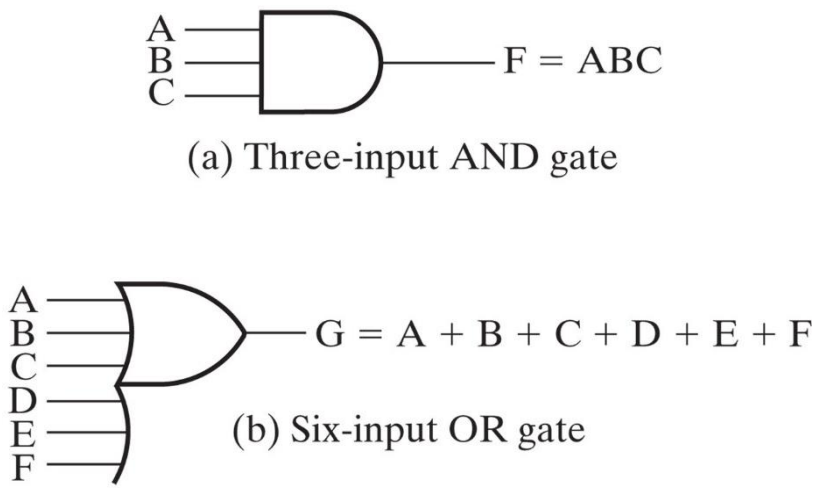
Graphic symbols



Timing diagram

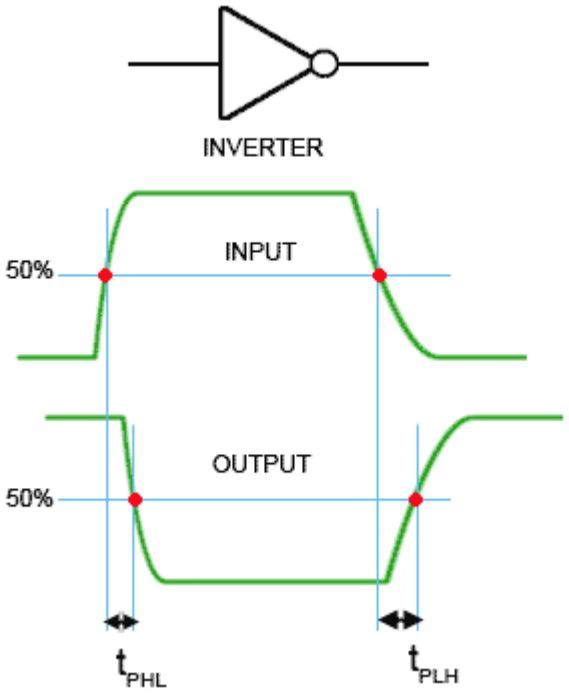


Multi-input gate



Gate Delay

- Each gate has a very important property called gate delay
  - The length of time it takes for an input change to result in the corresponding output change
  - It depends on the technology node (ex: 7nm vs. 65nm), # of inputs, or a gate type



t_{PHL} = Propagation delay high-low
t_{PLH} = Propagation delay low-high

Example – Reading the datasheet

- SN74LVC1G04 – Single Inverter Gate

TEXAS INSTRUMENTS

SN74LVC1G04
SCES214AD–APRIL1999–REVISED OCTOBER 2014

SN74LVC1G04 Single Inverter Gate

1 Features

- Available in the Ultra-Small 0.64-mm^2 Package (DPW) with 0.5-mm Pitch
- Supports 5-V V_{CC} Operation
- Inputs Accept Voltages up to 5.5 V Allowing Down Translation to V_{CC}
- Max t_{pd} of 3.3 ns at 3.3-V
- Low Power Consumption, 10-μA Max I_{CC}
- ±24-mA Output Drive at 3.3-V
- I_{off} Supports Live-Insertion, Partial-Power-Down Mode, and Back-Drive Protection
- Latch-Up Performance Exceeds 100 mA Per JESD 78, Class II
- ESD Protection Exceeds JESD 22
  - 2000-V Human-Body Model (A114-A)
  - 200-V Machine Model (A115-A)
  - 1000-V Charged-Device Model (C101)

2 Applications

- AV Receiver
- Audio Dock: Portable
- Blu-ray Player and Home Theater
- Embedded PC
- MP3 Player/Recorder (Portable Audio)
- Personal Digital Assistant (PDA)
- Power: Telecom/Server AC/DC Supply: Single Controller: Analog and Digital
- Solid State Drive (SSD): Client and Enterprise
- TV: LCD/Digital and High-Definition (HDTV)
- Tablet: Enterprise
- Video Analytics: Server
- Wireless Headset, Keyboard, and Mouse

4 Simplified Schematic

3 Description

This single inverter gate is designed for 1.65-V to 5.5-V V_{CC} operation.

The SN74LVC1G04 device performs the Boolean function $Y = \overline{A}$.

The CMOS device has high output drive while maintaining low static power dissipation over a broad V_{CC} operating range.

The SN74LVC1G04 device is available in a variety of packages, including the ultra-small DPW package with a body size of 0.8 mm × 0.8 mm.

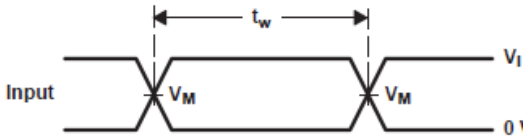
Device Information(1)

DEVICE NAME	PACKAGE	BODY SIZE
SN74LVC1G04	SOT-23 (5)	2.9mm × 1.6mm
	SC70 (5)	2.0mm × 1.25mm
	SON (6)	1.45mm × 1.0mm
	SON (6)	1.0mm × 1.0mm
	X2SON (4)	0.8mm × 0.8mm

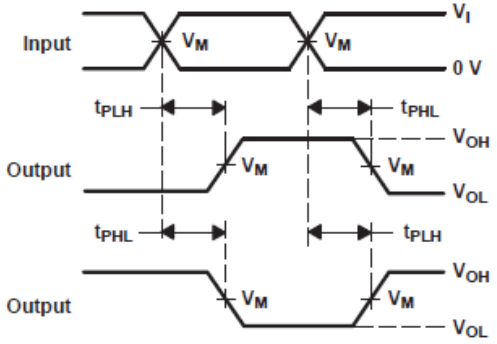
(1) For all available packages, see the orderable addendum at the end of the datasheet.

Function Table

INPUT A	OUTPUT Y
H	L
L	H



VOLTAGE WAVEFORMS
PULSE DURATION



VOLTAGE WAVEFORMS
PROPAGATION DELAY TIMES
INVERTING AND NONINVERTING OUTPUTS

7.6 Switching Characteristics, C_L = 15 pF

over recommended operating free-air temperature range, C_L = 15 pF (unless otherwise noted) (see Figure 3)

PARAMETER	FROM (INPUT)	TO (OUTPUT)	–40°C to 85°C								UNIT
			V_{CC} = 1.8 V ± 0.15 V		V_{CC} = 2.5 V ± 0.2 V		V_{CC} = 3.3 V ± 0.3 V		V_{CC} = 5 V ± 0.5 V		
			MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	
t_{pd}	A	Y	2	6.4	1	4.2	0.7	3.3	0.7	3.1	ns

7.7 Switching Characteristics, C_L = 30 pF or 50 pF, –40°C to 85°C

over recommended operating free-air temperature range, C_L = 30 pF or 50 pF (unless otherwise noted) (see Figure 4)

PARAMETER	FROM (INPUT)	TO (OUTPUT)	–40°C to 85°C								UNIT
			V_{CC} = 1.8 V ± 0.15 V		V_{CC} = 2.5 V ± 0.2 V		V_{CC} = 3.3 V ± 0.3 V		V_{CC} = 5 V ± 0.5 V		
			MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	
t_{pd}	A	Y	3	7.5	1.4	5.2	1	4.2	1	3.7	ns

7.8 Switching Characteristics, C_L = 15 pF, –40°C to 125°C

over recommended operating free-air temperature range, C_L = 15 pF (unless otherwise noted) (see Figure 3)

PARAMETER	FROM (INPUT)	TO (OUTPUT)	–40°C to 125°C								UNIT
			V_{CC} = 1.8 V ± 0.15 V		V_{CC} = 2.5 V ± 0.2 V		V_{CC} = 3.3 V ± 0.3 V		V_{CC} = 5 V ± 0.5 V		
			MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	
t_{pd}	A	Y	2	6.4	1	4.2	0.7	3.3	0.7	3.1	ns

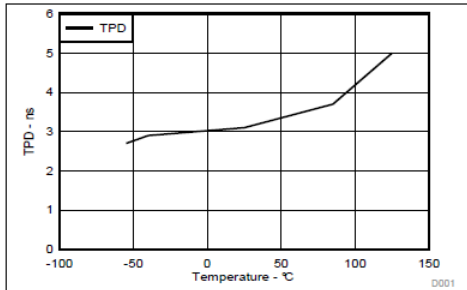


Figure 1. TPD Across Temperature at 3.3-V V_{CC}

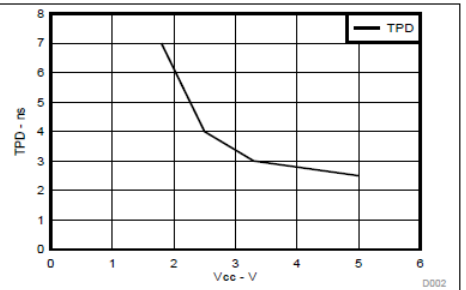


Figure 2. TPD Across V_{CC} at 25°C

G. t_{PLH} and t_{PHL} are the same as t_{pd}.

Boolean Function

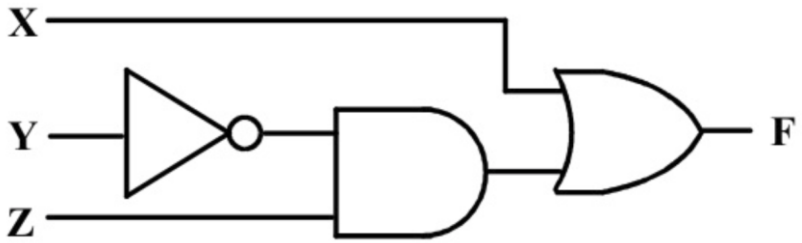
- Boolean algebra: an algebra dealing with binary variables
- Boolean function *F* (or Boolean expression)
  - *X* and $\bar{Y}Z$ are called '*terms*'

$$F = X + \bar{Y}Z$$

Denotes the Boolean function

Algebraic expression formed by binary variables

Logic circuit diagram



NOT UNIQUE!!
Can be simplified in some cases
(The end justifies the means)

Truth table

X Y Z	$F = X + \bar{Y} \cdot Z$
0 0 0	0
0 0 1	1
0 1 0	0
0 1 1	0
1 0 0	1
1 0 1	1
1 1 0	1
1 1 1	1

UNIQUE!!

Boolean Function Example

- We can design a logic for lowering the driver's power window

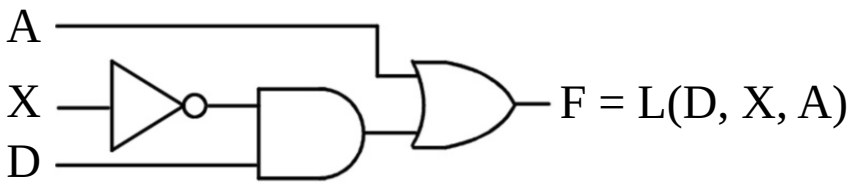
$$L(D,X,A) = D\overline{X} + A$$

- *L*: "lower the window" command
- *D*: output produced by pushing the ↓ button
- *X*: output of a mechanical limit
- *A*: onset of automated lowering operation (ex: when *D*=1 for more than 0.5s)

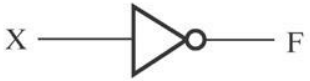
Truth Table??

A X D	$F = A + \overline{X} \cdot D$
0 0 0	0
0 0 1	1
0 1 0	0
0 1 1	0
1 0 0	1
1 0 1	1
1 1 0	1
1 1 1	1

Circuit Diagram



Logic Gates & Boolean Function

Name	Distinctive-Shape Graphics Symbol	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

Name	Distinctive-Shape Graphics Symbol	Algebraic Equation	Truth Table
NAND		$F = \overline{X \cdot Y}$	X Y | F 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR		$F = \overline{X + Y}$	X Y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0

Universal Gates

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

Boolean Algebra

- Basic identities of Boolean algebra

1.	$X + 0 = X$		2. $X \cdot 1 = X$	Identity element
3.	$X + 1 = 1$	Duality	4. $X \cdot 0 = 0$	
5.	$X + X = X$		6. $X \cdot X = X$	Idempotence
7.	$X + \overline{X} = 1$		8. $X \cdot \overline{X} = 0$	Complement
9.	$\overline{\overline{X}} = X$			Involution
10.	$X + Y = Y + X$		11. $XY = YX$	Commutative
12.	$(X + Y) + Z = X + (Y + Z)$		13. $(XY)Z = X(YZ)$	Associative
14.	$X(Y + Z) = XY + XZ$		15. $X + YZ = (X + Y)(X + Z)$	Distributive
16.	$\overline{X + Y} = \overline{X} \cdot \overline{Y}$		17. $\overline{X \cdot Y} = \overline{X} + \overline{Y}$	DeMorgan's

Extension of DeMorgan's Theorem

- Very important in Boolean algebra
  - Used to obtain the complement of an expression
  - Manipulate it to reduce # of terms/literals in a function
- Can extend to multiple variables

$$\overline{X_1 + X_2 + \cdots + X_n} = \overline{X_1} \cdot \overline{X_2} \cdot \ldots \cdot \overline{X_n}$$

$$\overline{X_1 X_2 \ldots X_n} = \overline{X_1} + \overline{X_2} + \ldots + \overline{X_n}$$

Algebraic Manipulation

- Can **simplify digital circuits** at early design stage
- Let's consider

$$F = \bar{X}YZ + \bar{X}Y\bar{Z} + XZ$$

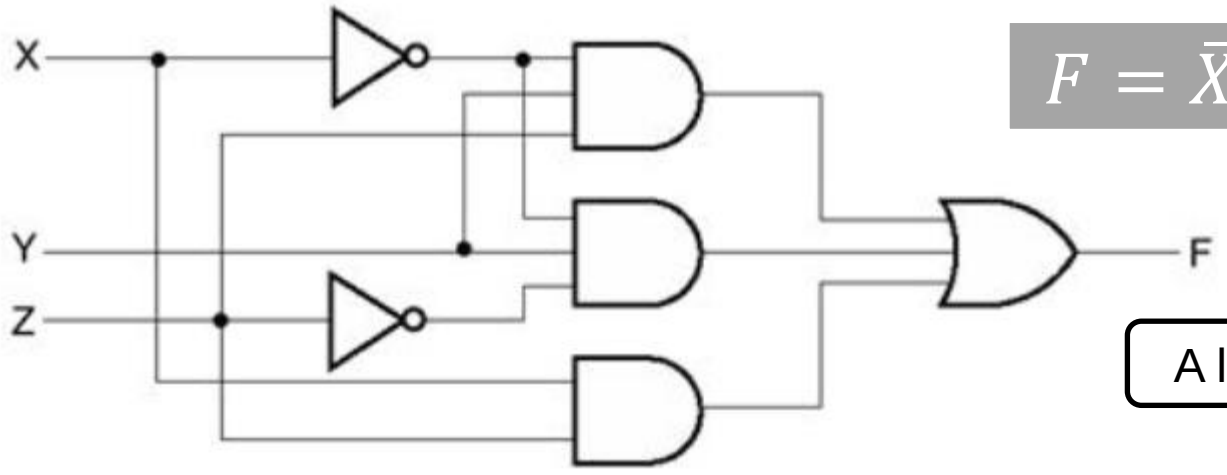
$$= \bar{X}Y(Z + \bar{Z}) + XZ$$ Identity 14: distributive

$$= \bar{X}Y \cdot 1 + XZ$$ Identity 7: Complement

$$= \bar{X}Y + XZ$$ Identity 2: Identity Element

Reduced to two terms from three

What is the Benefit of Algebraic Manipulation?



$$F = \bar{X}YZ + \bar{X}Y\bar{Z} + XZ$$

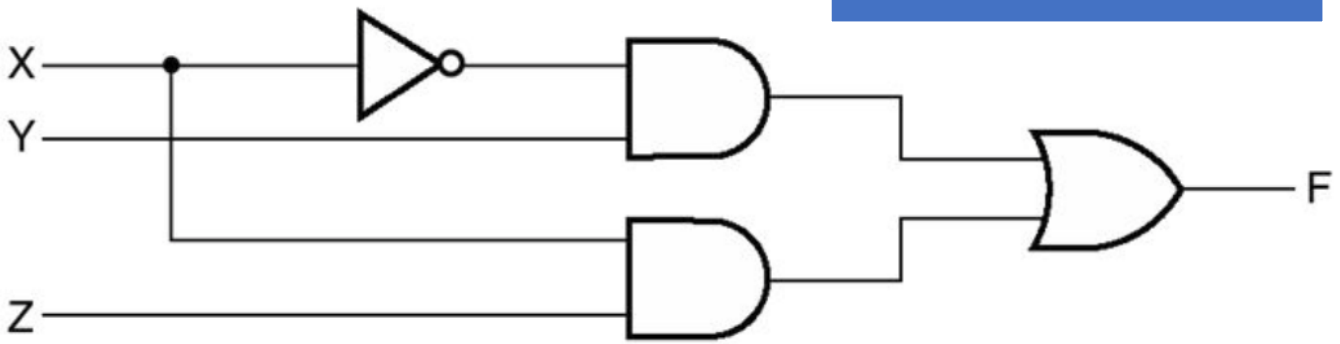
Three terms, eight literals

A logic gate

An input to a gate

Two terms, four literals

$$F = \bar{X}Y + XZ$$



Reduce an expression!!

1. **Fewer** gates
2. **Fewer** inputs per gate

But, Same Truth Table!

$$F = \bar{X}YZ + \bar{X}Y\bar{Z} + XZ$$

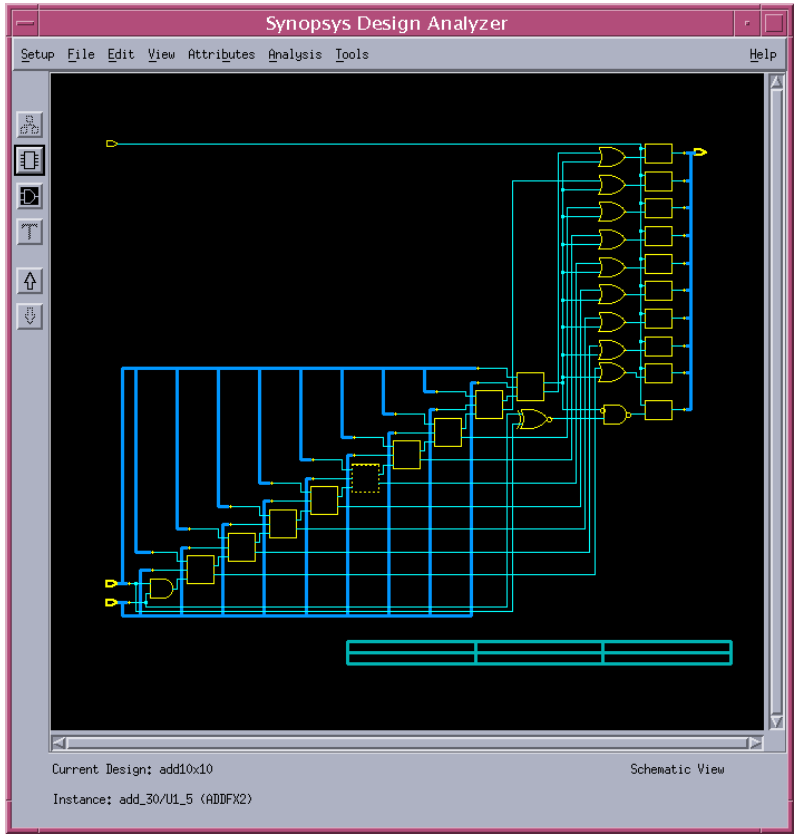
$$F = \bar{X}Y + XZ$$

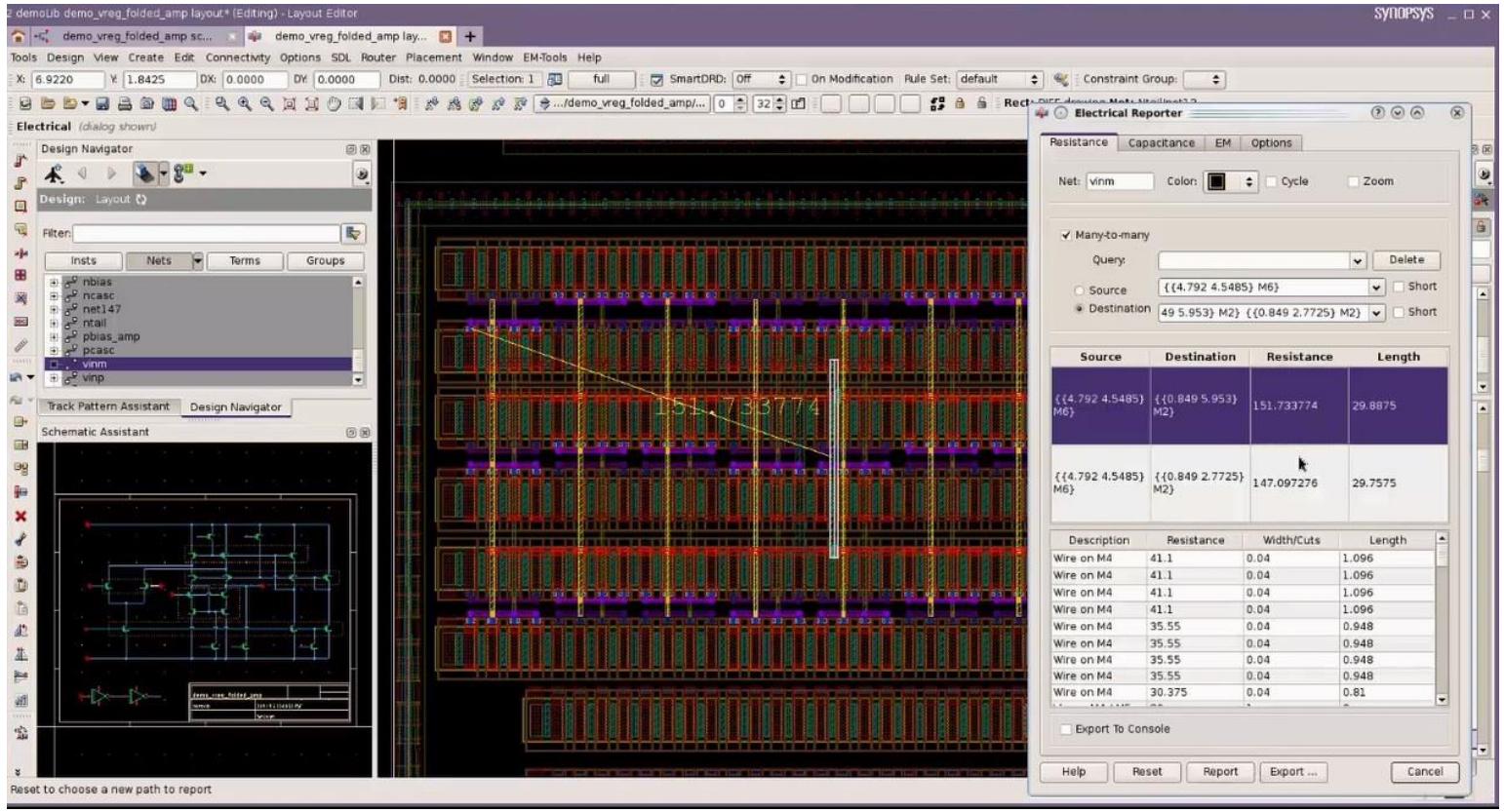
Truth Table for Boolean Function

X	Y	Z	(a) F	(b) F
0	0	0	0	0
0	0	1	0	0
0	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	0	0
1	1	1	1	1

Logic Synthesis: Synopsys Design Compiler

- Logic synthesis tools help designers reduce complex expressions





Logic Gate Reduction

- Again, use Boolean algebra to reduce complexity of digital circuits

1.	$X + 0 = X$	2.	$X \cdot 1 = X$	Identity element
3.	$X + 1 = 1$	4.	$X \cdot 0 = 0$	
5.	$X + X = X$	6.	$X \cdot X = X$	Idempotence
7.	$X + \overline{X} = 1$	8.	$X \cdot \overline{X} = 0$	Complement
9.	$\overline{\overline{X}} = X$			Involution
10.	$X + Y = Y + X$	11.	$XY = YX$	Commutative
12.	$(X + Y) + Z = X + (Y + Z)$	13.	$(XY)Z = X(YZ)$	Associative
14.	$X(Y + Z) = XY + XZ$	15.	$X + YZ = (X + Y)(X + Z)$	Distributive
16.	$\overline{X + Y} = \overline{X} \cdot \overline{Y}$	17.	$\overline{X \cdot Y} = \overline{X} + \overline{Y}$	DeMorgan's

$$X + XY = X(\mathbf{1} + \boldsymbol{Y}) = X$$

$$XY + X\overline{Y} = X(\boldsymbol{Y} + \overline{\boldsymbol{Y}}) = X$$

$$X + \overline{X}Y = (\boldsymbol{X} + \overline{\boldsymbol{X}})(X + Y) = X + Y$$

Logic Gate Reduction

1.	$X+0=X$	2.	$X\cdot 1=X$	Identity element
3.	$X+1=1$	4.	$X\cdot 0=0$	
5.	$X+X=X$	6.	$X\cdot X=X$	Idempotence
7.	$X+\overline{X}=1$	8.	$X\cdot\overline{X}=0$	Complement
9.	$\overline{\overline{X}}=X$			Involution
10.	$X+Y=Y+X$	11.	$XY=YX$	Commutative
12.	$(X+Y)+Z=X+(Y+Z)$	13.	$(XY)Z=X(YZ)$	Associative
14.	$X(Y+Z)=XY+XZ$	15.	$X+YZ=(X+Y)(X+Z)$	Distributive
16.	$\overline{X+Y}=\overline{X}\cdot\overline{Y}$	17.	$\overline{X\cdot Y}=\overline{X}+\overline{Y}$	DeMorgan's

$$X(X+Y)=X+XY=X$$

$$(X+Y)(X+\bar{Y})=X+Y\bar{Y}=X$$

$$X(\bar{X}+Y)=X\bar{X}+XY=XY$$

$$X+XY=X(\mathbf{1}+\boldsymbol{Y})=X$$

$$XY+X\bar{Y}=X(\boldsymbol{Y}+\bar{\boldsymbol{Y}})=X$$

$$X+\bar{X}Y=(\boldsymbol{X}+\bar{\boldsymbol{X}})(X+Y)=X+Y$$

Duals of previous examples

Consensus Theorem

- Allows us to remove a redundant term

$$XY + \bar{X}Z + YZ = XY + \bar{X}Z$$

Proof:

$$\begin{aligned} XY + \bar{X}Z + YZ &= XY + \bar{X}Z + YZ(X + \bar{X}) \\ &= XY + \bar{X}Z + XYZ + \bar{X}YZ \\ &= XY(1 + Z) + \bar{X}(Z + YZ) \\ &= XY + \bar{X}Z \end{aligned}$$

Sum of Product form

Dual of consensus theorem

$$(X + Y)(\bar{X} + Z)(Y + Z) = (X + Y)(\bar{X} + Z)$$

Product of Sum form

Consensus Theorem: Example

- Example of minimizing Boolean expression with consensus theorem

$$
\begin{aligned}
(A+B)(\bar{A}+C) &= A\bar{A} + AC + \bar{A}B + BC \\
&= AC + \bar{A}B + BC \\
&= AC + \bar{A}B
\end{aligned}
$$

Consensus theorem applied

Complement of a Function

- Obtain by interchanging 1's to 0's and vice versa in the truth table
- Also, it can be derived by using DeMorgan's theorem

$$\overline{F_1} = \overline{\bar{X}Y\bar{Z} + \bar{X}\bar{Y}Z} = \overline{(\bar{X}Y\bar{Z})} \cdot \overline{(\bar{X}\bar{Y}Z)}$$
$$= (X + \bar{Y} + Z) \cdot (\mathrm{X} + \mathrm{Y} + \bar{Z})$$

$$\overline{F_2} = \overline{X(\bar{Y}\bar{Z} + YZ)} = \bar{X} + \overline{(\bar{Y}\bar{Z} + YZ)}$$
$$= \bar{X} + (\overline{\bar{Y}\bar{Z}} \cdot \overline{YZ}) = \bar{X} + (Y + Z)(\bar{Y} + \bar{Z})$$

Standard Forms - Minterms

- **Minterm**: a product term in which all the variables appear only once
  - It represents exactly one combination of the binary variables in a truth table
  - For ‘n’ variables, there are ‘2^n’ distinct minterms

Minterms for Three Variables

1 for a specific binary combination

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

Two discrete signals

Standard Forms - Maxterms

- **Maxterm**: a sum term that contains all the variables
  - Each maxterm is a logical sum with each variable
  - Complemented if it is 1 and uncomplemented if it is 0

0 for a specific binary combination

Maxterms for Three Variables

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

$$M_j = \overline{m_j}$$

Representing a Boolean Function w/ Minterms

- A Boolean function can be expressed by forming logical sum of all the minterms that produce a 1 in the truth table

Boolean function F

X	Y	Z	F	$\bar{F}$
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

Sum of minterms

$$F = (\bar{X}\bar{Y}\bar{Z} + \bar{X}Y\bar{Z} + X\bar{Y}Z + XYZ)$$
$$= m_0 + m_2 + m_5 + m_7$$

Can be abbreviated by listing only decimal subscripts of minterms

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

Logical sum (Boolean OR)

Representing a Boolean Function w/ Maxterms

- Consider the complement of a Boolean function F

X	Y	Z	F	$\bar{F}$
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

$$\bar{F} = (\bar{X}\bar{Y}Z + \bar{X}YZ + X\bar{Y}\bar{Z} + XY\bar{Z})$$
$$= m_1 + m_3 + m_4 + m_6 = \sum m(1,3,4,6)$$

Take complement again

$$F = \overline{m_1 + m_3 + m_4 + m_6}$$
$$= \overline{m_1} \cdot \overline{m_3} \cdot \overline{m_4} \cdot \overline{m_6} = M_1 \cdot M_3 \cdot M_4 \cdot M_6$$
$$= (X + Y + \bar{Z})(X + \bar{Y} + \bar{Z})(\bar{X} + Y + Z)(\bar{X} + \bar{Y} + Z)$$

Logical product (Boolean AND)

Product of maxterms

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

Summary on Minterms

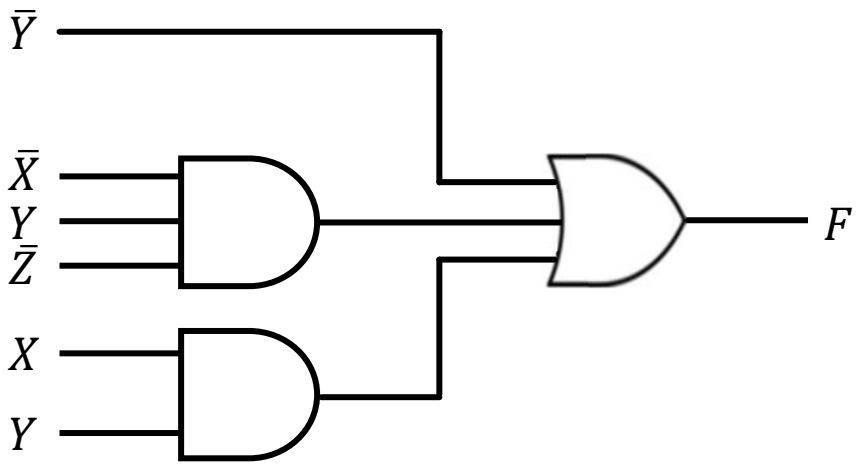
- Important properties of minterms
  1. There are 2^n minterms for n Boolean variables
  2. Any Boolean function can be expressed as a logical sum of minterms
  3. The complement of a function contains those minterms not included in the original function
  4. A function that includes all the 2^n minterms is equal to logic 1

Sum of Products (SoP)

- Sum of minterms: directly obtained from the truth table
  - Minterm contains all binary variables by definition
  - Each term has more literals than necessary (chance of simplification)
  - Reduce # of terms or # of literals in the terms

$$F(X,Y,Z) = \bar{Y} + \bar{X}Y\bar{Z} + XY$$

Not all variables (X,Y,Z)
are associated with each term
- Each term has up to 'n' literals



Logic diagram of SoP: a group of AND gates followed by a single OR gate

Two-level implementation

Conversion to Sum-of-Products Form

- Consider a Boolean function not in SoP form

$$F = AB + C(D + E)$$

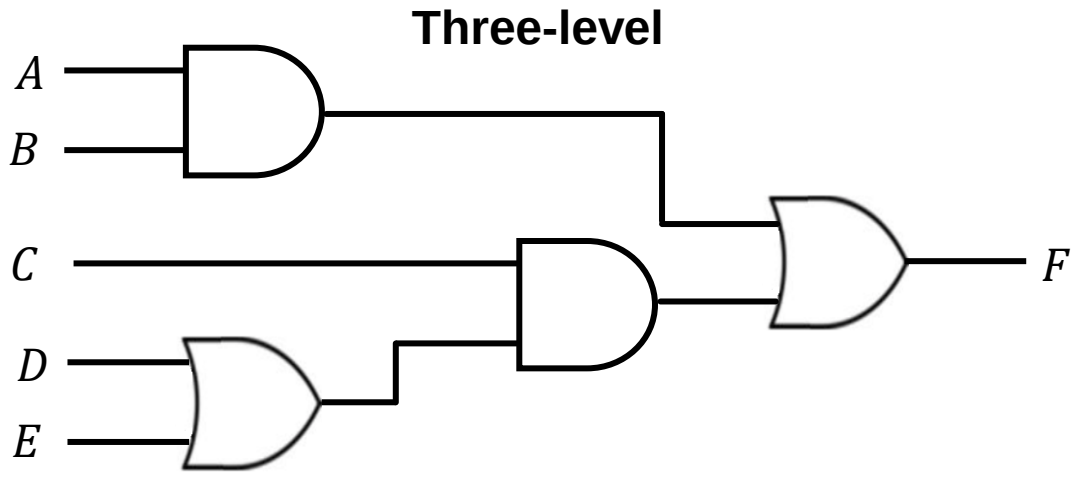
- $(D + E)$ is part of a product, but not a single literal
- The expression can be converted to SoP form by using distributive law as follows:

$$F = AB + C(D + E) = AB + CD + CE$$

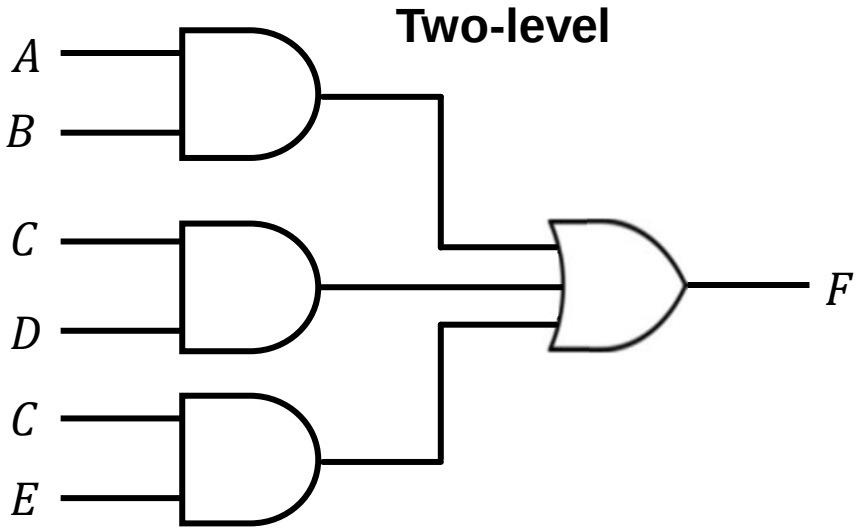
Two-level vs. Multi-level Implementation

- In the previous example, let's compare actual logic implementations

$$F = AB + C(D + E)$$



$$F = AB + CD + CE$$

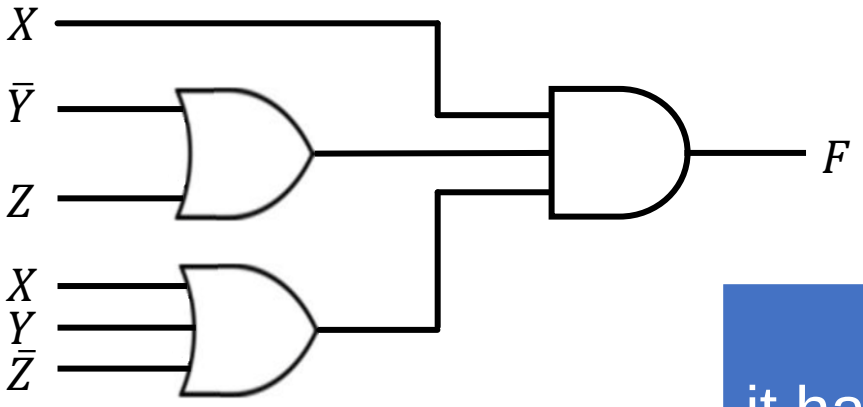


QUESTION: which one do you think is better??

Product of Sums (PoS)

- Another standard form of expressing Boolean functions
- Each logical sum term may have any number of distinct literals

$$F = X(\bar{Y} + Z)(X + Y + \bar{Z})$$



Similar to SoP form,
it has two-level implementation

Converting SoP to PoS Form

- **Step 1**: Evaluate each product term in the SoP expression.

  Determine the binary numbers that represent the product terms.

- **Step 2**: Determine all of the binary numbers not included in the evaluation in Step 1.

- **Step 3**: Write in equivalent sum term for each binary number Step 2 and expression in PoS form.

Example of Converting Between Forms

- Note that there are eight possible minterms/maxterms

$$SoP = \sum m(0,2,3,5,7)$$
$$= \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}C + ABC$$

$$PoS = \prod M(1,4,6)$$
$$= (\bar{A} + \bar{B} + C)(A + \bar{B} + \bar{C})(A + B + \bar{C})$$

Boolean Expression and Truth Table

- Convert SoP form to truth table

$$SoP = \bar{A}\bar{B}C + A\bar{B}\bar{C} + ABC = m(1,4,7)$$

Inputs A	B	C	Output X	Product Term
0	0	0	0	
0	0	1	1	A'B'C
0	1	0	0	
0	1	1	0	
1	0	0	1	AB'C'
1	0	1	0	
1	1	0	0	
1	1	1	1	ABC

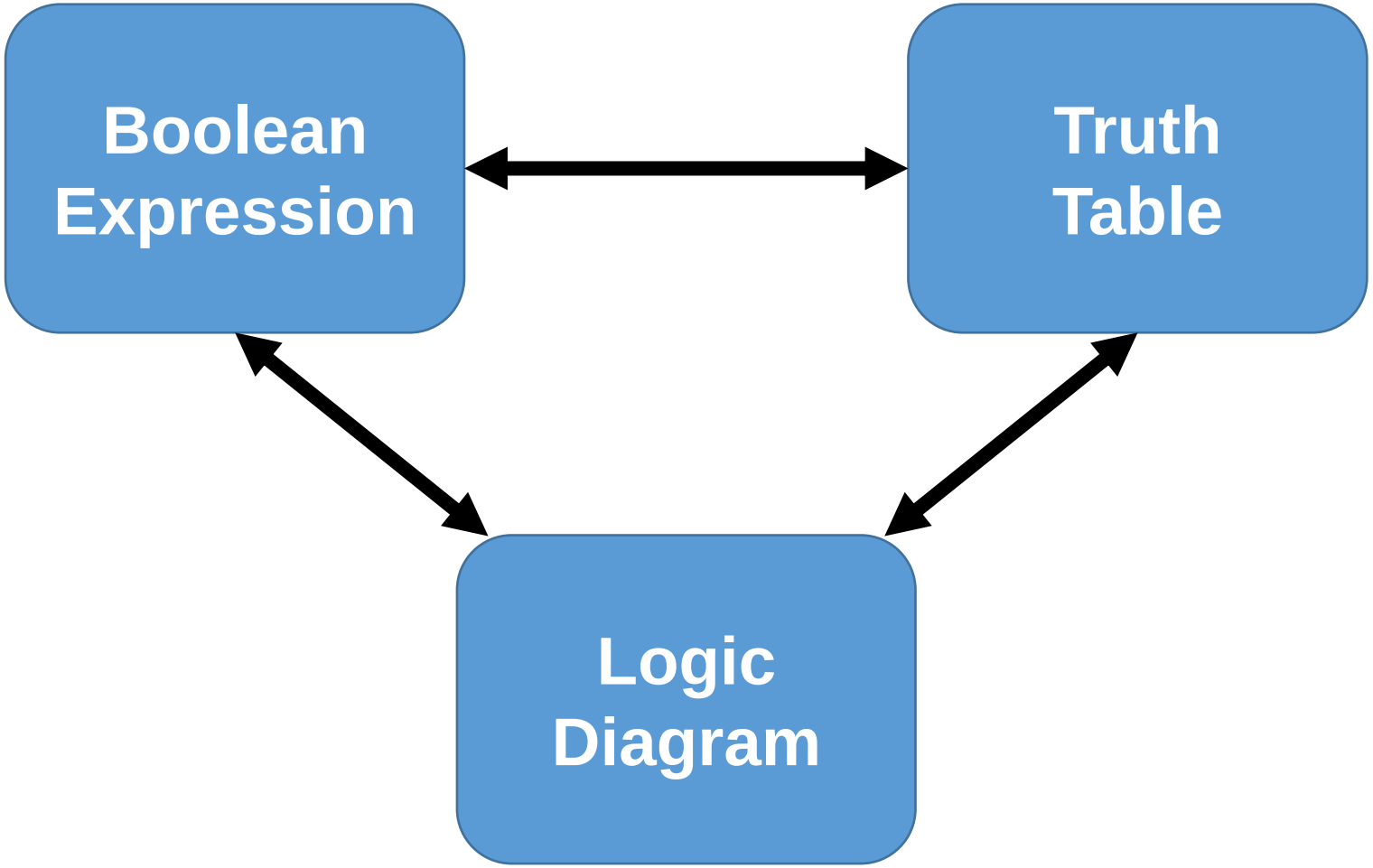
Boolean Expression and Truth Table

- Convert PoS form to truth table

$$PoS = (A + B + C)(A + \bar{B} + C)(A + \bar{B} + \bar{C})(\bar{A} + B + \bar{C})(\bar{A} + \bar{B} + C)$$
$$= \prod M(0,2,3,5,6)$$

Inputs A	B	C	Output X	Sum Term
0	0	0	0	A+B+C
0	0	1	1	
0	1	0	0	A+B'+C
0	1	1	0	A+B'+C'
1	0	0	1	
1	0	1	0	A'+B+C'
1	1	0	0	A'+B'+C
1	1	1	1	

How to Express Boolean Function?

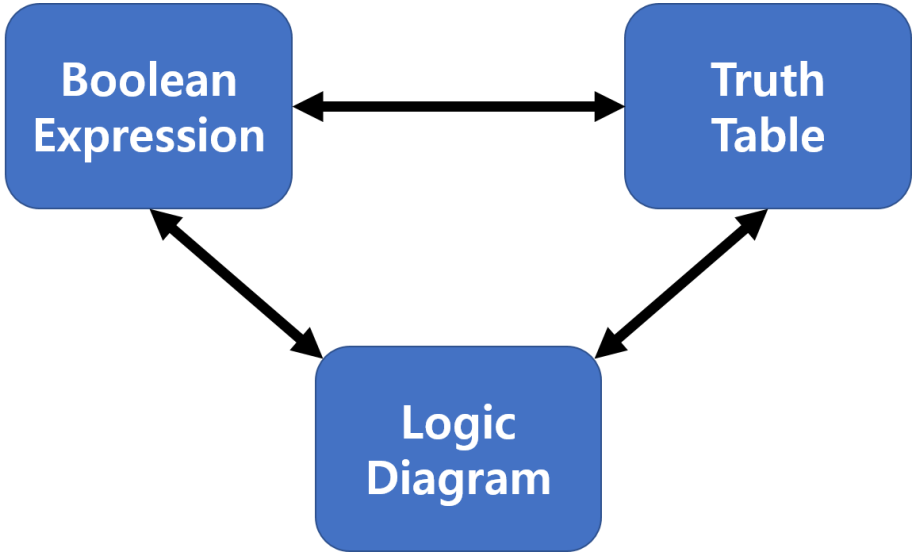


Recap on Previous Lecture

- We have looked at…
  - What binary system is
  - The basics of **Boolean algebra**
  - How to manipulate Boolean algebra in logic minimization
  - Standard forms: **sum-of-products**, product-of-sums
  - **Truth table** of a Boolean function

Boolean algebra

		2.	$X \cdot 1 = X$	Identity element
3.	$X + 1 = 1$	4.	$X \cdot 0 = 0$	
5.	$X + X = X$	6.	$X \cdot X = X$	Idempotence
7.	$X + \overline{X} = 1$	8.	$X \cdot \overline{X} = 0$	Complement
9.	$\overline{\overline{X}} = X$			Involution
10.	$X + Y = Y + X$	11.	$XY = YX$	Commutative
12.	$(X + Y) + Z = X + (Y + Z)$	13.	$(XY)Z = X(YZ)$	Associative
14.	$X(Y + Z) = XY + XZ$	15.	$X + YZ = (X + Y)(X + Z)$	Distributive
16.	$\overline{X + Y} = \overline{X} \cdot \overline{Y}$	17.	$\overline{X \cdot Y} = \overline{X} + \overline{Y}$	DeMorgan's

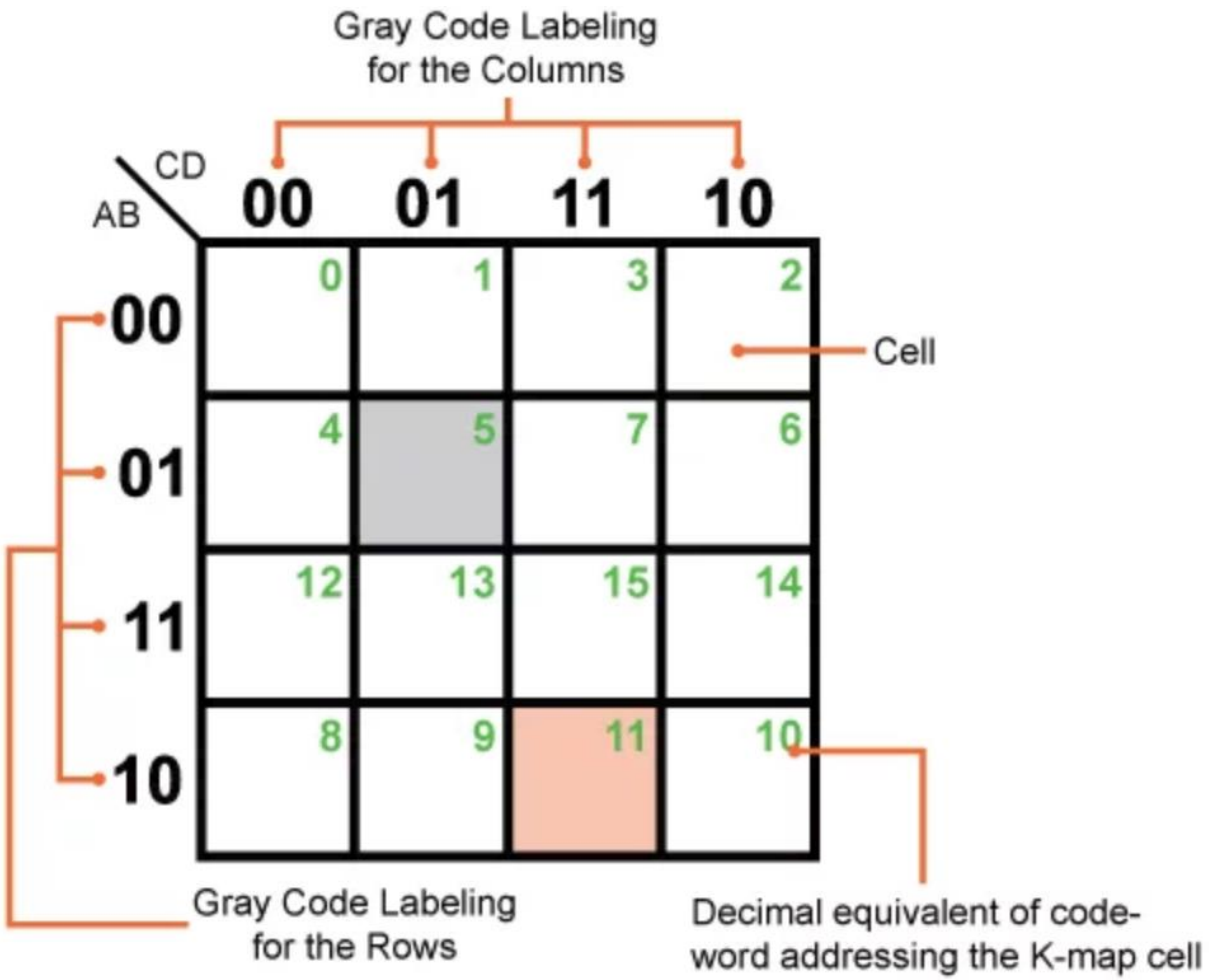


Map Simplification

- A Boolean function can be expressed in different algebraic forms
- How to obtain a form that simplifies the digital implementation?
  - It impacts power, area, performance…
- Algebraic manipulation allowed us to reduce logic gates
  - How do you guarantee a reduced form has the minimum # of gates?
- Map method provides nicer way to minimize logic gates!!
  - Up to four variables
  - This method is called **Karnaugh map, or K-map**

Karnaugh Map

- A map is a diagram made up of squares
  - Each square represents one minterm of the function
  - A visual diagram of all possible ways a function may be expressed in a standard form

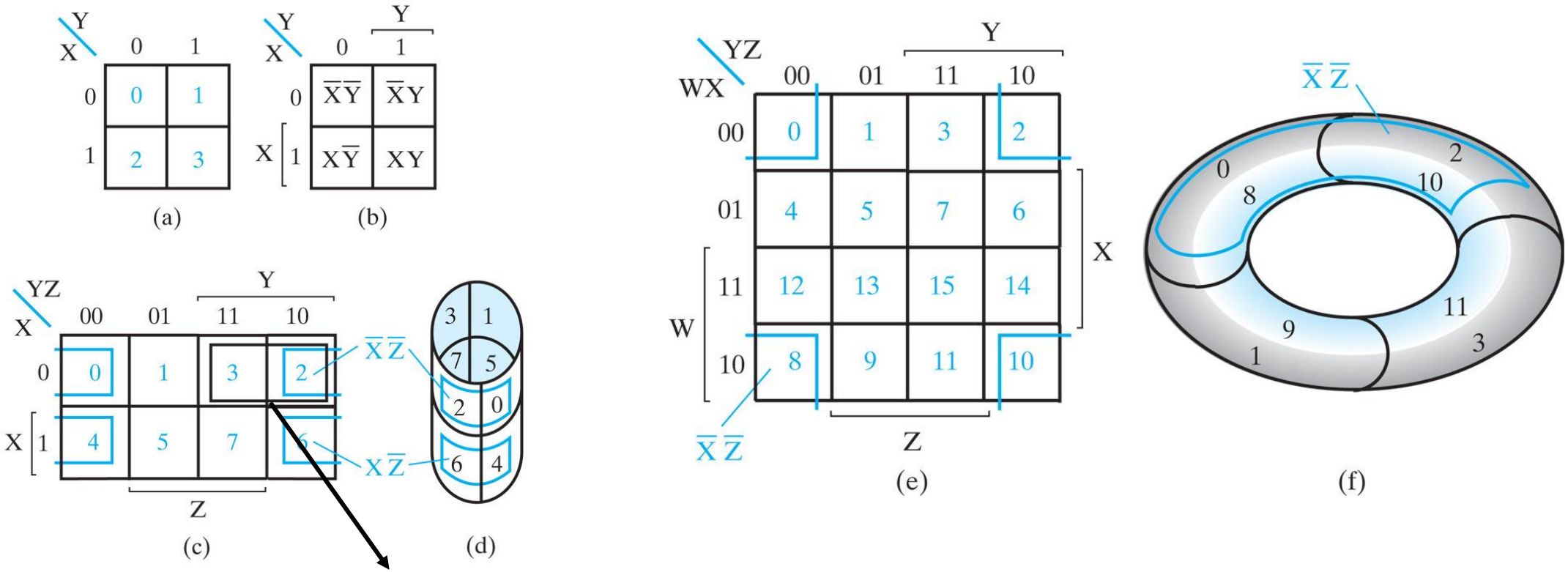


Outcome of K-map Simplification

- Always in sum-of-products or product-of-sums form
- Maps handle simplification of two-level implementations
  - Not directly to multi-level implementations
- What is the simplest algebraic expression?
  - One with a minimum # of terms (**# of gates**)
  - One with the fewest possible # of literals in each term (**# of inputs**)
  - Not necessarily unique!

Map Structures

- The structure depends on # of variables involved in expressing a Boolean function



'***Adjacent***' minterms

Two-Variable Map

- There exist four minterms for a Boolean function with two variables
  - Hence, four squares in the map
  - $\bar{X}$ = row 0; X = row 1
  - $\bar{Y}$ = column 0; Y = column 1

Example of two-variable map

m_0	m_1
m_2	m_3

$X \backslash Y$	0	1
0	m_0 $\bar{X}\bar{Y}$	m_1 $x'y$
1	m_2 xy'	m_3 xy

$X \backslash Y$	0	1
0		
1		1

$F = XY$

$X \backslash Y$	0	1
0		1
1	1	1

$F = X + Y$

Distributive law applied!!

$$F = \bar{X}Y + X\bar{Y} + XY = \bar{X}Y + X(\bar{Y} + Y) = X + Y$$

Three-Variable Map

- There exist eight minterms for three variables
  - Grey coding is used for placing minterms for multiple variables
  - Why Grey coding (allow one bit change)?
- Consider this example..
  - $F = m_5 + m_7 = X\bar{Y}Z + XYZ = XZ(\bar{Y} + Y) = XZ$
  - Can reduce a term by combining multiple squares

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

	YZ	00	01	11	10
	0	$\bar{X}\bar{Y}\bar{Z}$	$\bar{X}\bar{Y}Z$	$\bar{X}YZ$	$\bar{X}Y\bar{Z}$
X	1	$X\bar{Y}\bar{Z}$	$X\bar{Y}Z$	XYZ	$XY\bar{Z}$

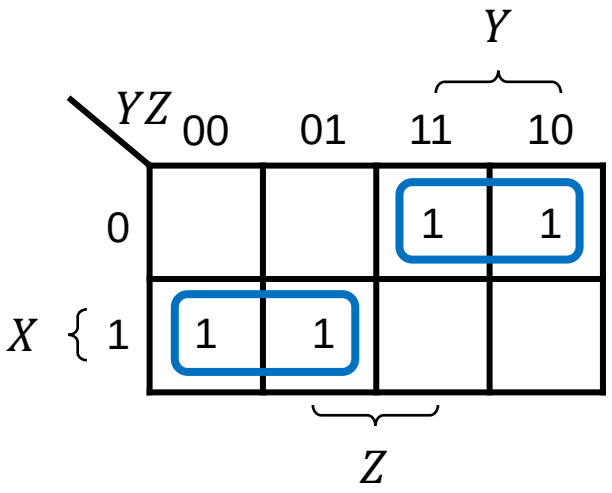
Y

Z

Example of Three-Variable Map Simplification

- Simplify the Boolean function

$$F(X,Y,Z) = \sum m(2,3,4,5)$$



- Then, we search for the collection of squares, called *rectangles*
  - Rectangle: # of squares is constrained to power of 2
- Objective is to find the fewest product terms (rectangles)

$$F = \bar{X}Y + X\bar{Y}$$

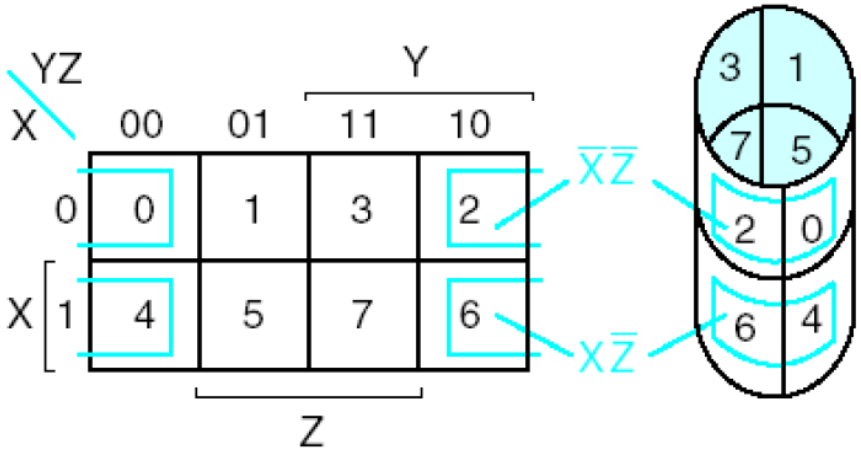
Two terms instead of four

More Examples

- What about this example?

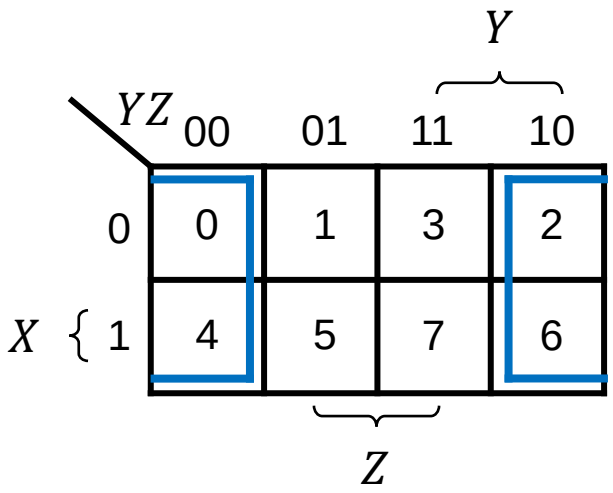
$$m_0 + m_2 = \bar{X}\bar{Y}\bar{Z} + \bar{X}Y\bar{Z} = \bar{X}\bar{Z}(\bar{Y} + Y) = \bar{X}\bar{Z}$$

$$m_4 + m_6 = X\bar{Y}\bar{Z} + XY\bar{Z} = X\bar{Z}(\bar{Y} + Y) = X\bar{Z}$$



- Can we reduce even further?

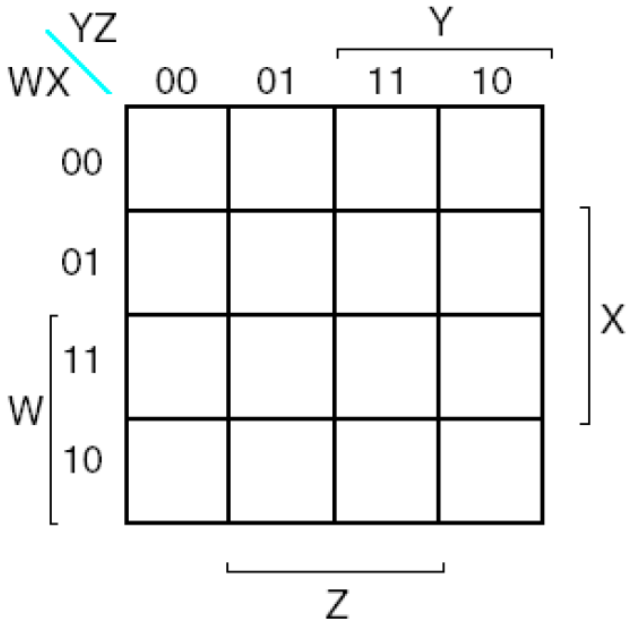
$$\begin{aligned} F &= m_0 + m_2 + m_4 + m_6 \\ &= X\bar{Y}\bar{Z} + XY\bar{Z} + \bar{X}\bar{Y}\bar{Z} + \bar{X}Y\bar{Z} \\ &= X\bar{Z}(\bar{Y} + Y) + \bar{X}\bar{Z}(\bar{Y} + Y) \\ &= X\bar{Z} + \bar{X}\bar{Z} = \bar{Z}(X + \bar{X}) = \bar{Z} \end{aligned}$$



Four Variable Map

- 4 binary variables correspond to 16 (2^4) minterms
- Possible rectangle groups
  - 2-squares: a product term w/ 3 literals
  - 4-squares: a product term w/ 2 literals
  - 8-squares: a product term w/ 1 literal
  - 16-squares: logic 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}



Simplifying a 4-Variable Function with Map

- Map for $F(W,X,Y,Z) = \sum m(0,1,2,4,5,6,8,9,12,13,14)$
- It is okay to use the same square more than once

WX \ YZ	00	01	11	10
00	1	1		1
01	1	1		1
11	1	1		1
10	1	1		

(2 AND + 1 OR) gates can implement *F*

$$F = \bar{Y} + \bar{W}\bar{Z} + X\bar{Z}$$

More Example

- Simplify the Boolean function

$$F(W,X,Y,Z) = \bar{W}\bar{X}\bar{Y} + \bar{X}Y\bar{Z} + W\bar{X}\bar{Y} + \bar{W}XY\bar{Z}$$

WX \ YZ	00	01	11	10
00	1	1		1
01				1
11				
10	1	1		1

Y, X, W, Z

Reduce to

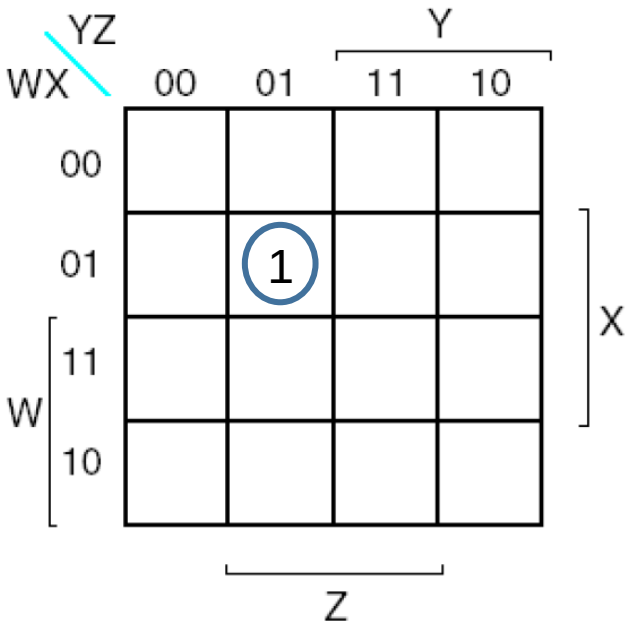
$$F = \bar{X}\bar{Z} + \bar{X}\bar{Y} + \bar{W}Y\bar{Z}$$

Essential Prime Implicants

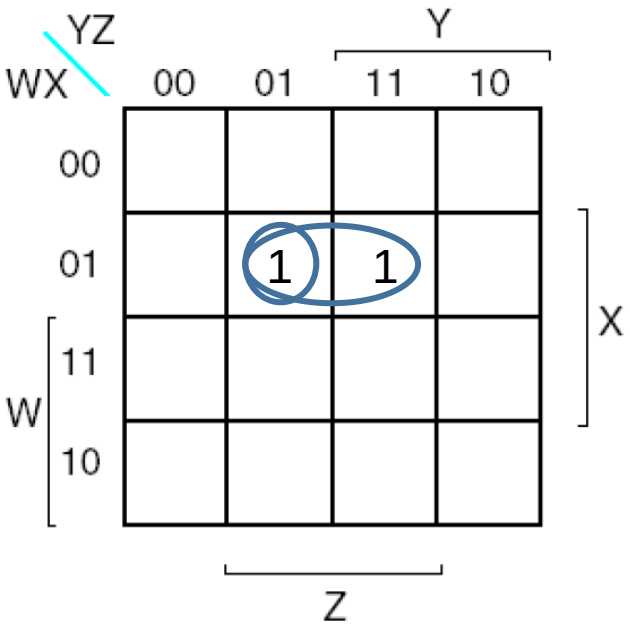
- **Implicant** of a function
  - A product term which has the value 1 for all minterms of the product term
  - **All rectangles on a map made up of squares containing 1's**
- **Prime implicant (PI)** of a function
  - If the removal of any literal from an implicant P results in a product term that is not an implicant of a function, then P is a prime implicant
  - **All rectangles made up of 2^m squares containing 1's**
- **Essential prime implicant (EPI)**
  - **If a minterm of a function is included in only one prime implicant**, that prime implicant is said to be essential

Example of Prime Implicants

- Is this a prime implicant?



- What about this?



If the removal of any literal from an implicant P
results in a product term that is not an implicant of a function,
then P is a prime implicant

Example of an Essential Prime Implicant

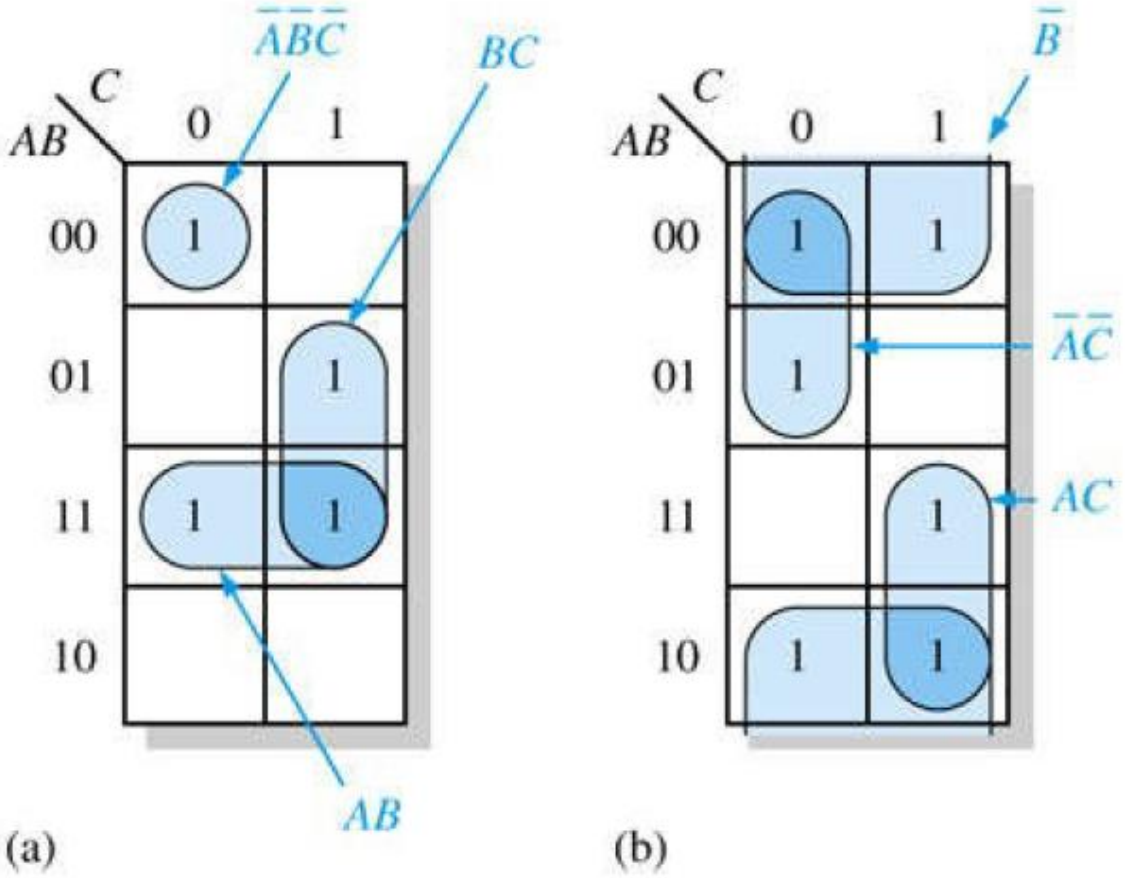
- Is this an essential prime implicant?

WX \ YZ	00	01	11	10
00				
01		1	1	
11			1	
10			1	

- What about this?

WX \ YZ	00	01	11	10
00				
01		1	1	
11			1	
10			1	

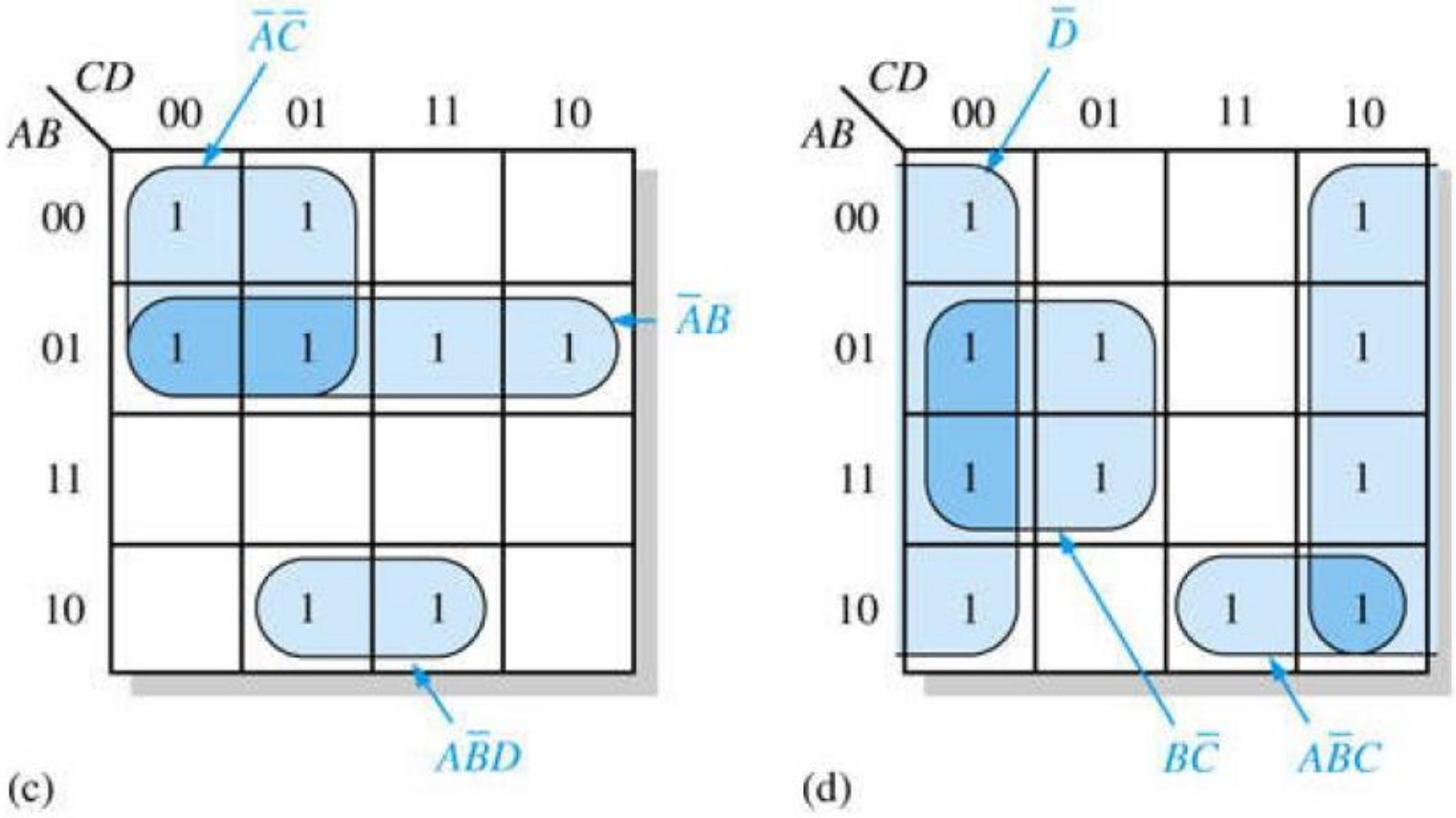
Example of an Essential Prime Implicant



$$AB + BC + \bar{A}\bar{B}\bar{C}$$

$$\bar{B} + AC + \bar{A}\bar{C}$$

Example of an Essential Prime Implicant



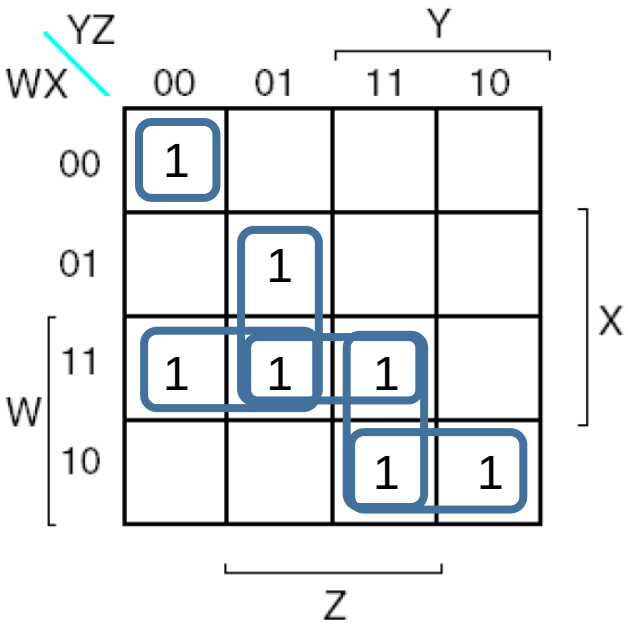
$$\bar{A}\bar{C} + \bar{A}B + A\bar{B}D$$

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

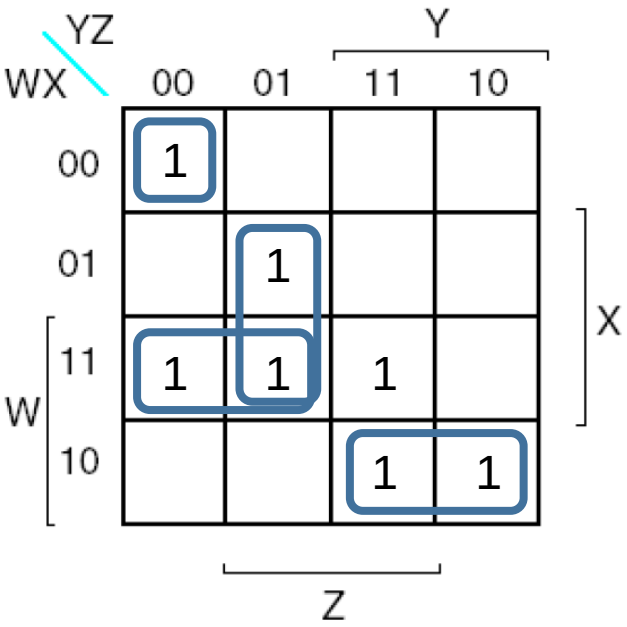
Procedure for Simplifying Expressions

- **Step 1**: Find all prime implicants
- **Step 2**: Obtain the logical sum of all the EPIs
- **Step 3**: Cover the remaining minterms not included in the EPIs

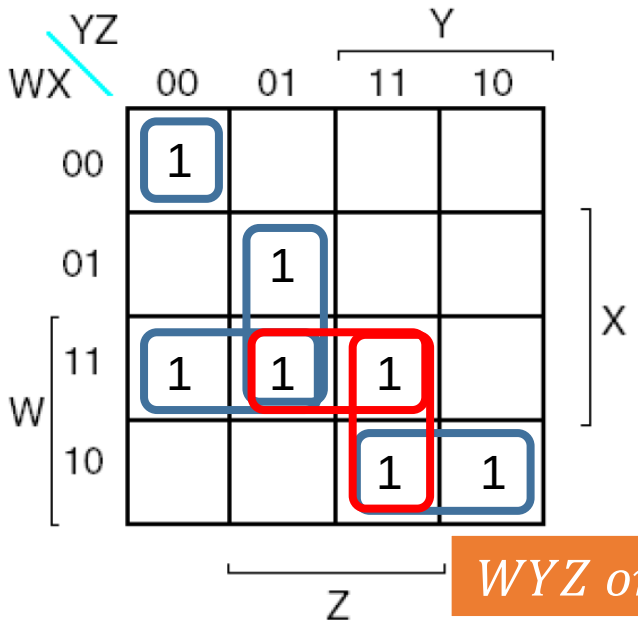
6 prime implicants



Cover EPIs



Cover the remaining minterms



WYZ or WXZ

Any Rule to Cover Nonessential Prime Implicants?

- **Selection rule**: minimize the overlap among prime implicants as much as possible

Only essential prime implicant

WX \ YZ	00	01	11	10
00	1	1		1
01	1	1		
11		1	1	
10			1	1

Minimize the overlap as much as possible

WX \ YZ	00	01	11	10
00	1	1		1
01	1	1		
11		1	1	
10			1	1

Product-of-Sums Simplification

- How can we obtain the simplified product-of-sums from the map?
  - 1's in the map represent the minterms
  - Let's simplify terms denoted by 0's → the complement of a function
  - Then, we can take complement on $\bar{F}$ to obtain F as a product of sums

Example

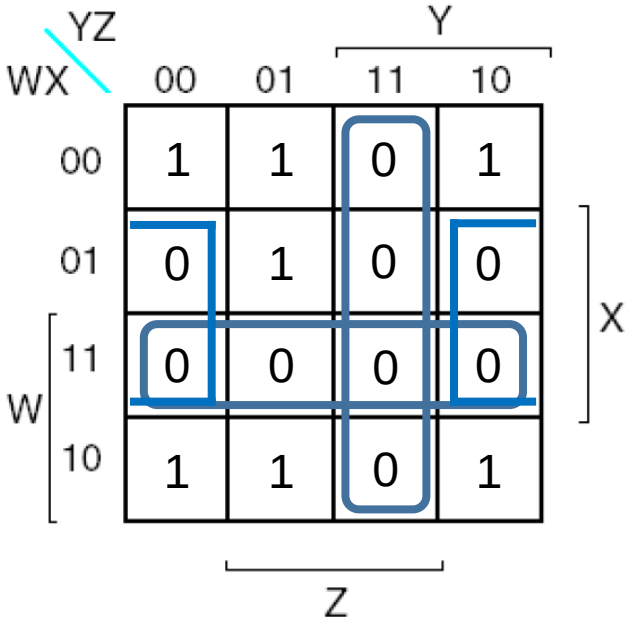
$$F(W,X,Y,Z) = \sum m(0,1,2,5,8,9,10)$$

- Combine the squares marked with 0's

$$\bar{F} = WX + YZ + X\bar{Z}$$

- Take the dual and complement each literal

$$F = (\bar{W} + \bar{X})(\bar{Y} + \bar{Z})(\bar{X} + Z)$$



Don’t Care Conditions

Incompletely specified function

- In some cases, there are applications where the function is not specified for certain variable value combinations
  - Certain input combinations may never happen (using 4-bit BCD code for decimal digits, Use only 0000 ~ 1001)

Binary-Coded Decimal (BCD)

Decimal Symbol	BCD Digit
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Decimal	Binay (BCD) 8 4 2 1
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=A	Don't Care
11=B	
12=C	
13=D	
14=E	
15=F	

Trivial!

Don’t Care Conditions

Incompletely specified function

- In some cases, there are applications where the function is not specified for certain variable value combinations
  - Certain input combinations may never happen (using 4-bit BCD code for decimal digits, Use only 0000 ~ 1001)
  - Sometimes we do not care about output for certain input combinations

Don’t care conditions

WX \ YZ	00	01	11	10
00	X	1	1	X
01	0	X	1	0
11	0	0	1	0
10	0	0	1	0

Y

X

W

Z

- We can’t put 0 or 1 in don’t care minterms
- To distinguish, we use ‘X’ mark
- It can be utilized to further simplify the function
- ‘X’ mark can be considered as either 0 or 1

Simplification Using Don't Care Conditions

- 'X' can be considered as either 0 or 1 to simplify the function in a map

$$F(W,X,Y,Z) = \sum m(1,3,7,11,15)$$

$$d(W,X,Y,Z) = \sum m(0,2,5)$$

Algebraically unequal,
but both solutions are acceptable

WX \ YZ	00	01	11	10
00	X	1	1	X
01	0	X	1	0
11	0	0	1	0
10	0	0	1	0

$$F = YZ + \bar{W}\bar{X}$$

WX \ YZ	00	01	11	10
00	X	1	1	X
01	0	X	1	0
11	0	0	1	0
10	0	0	1	0

$$F = YZ + \bar{W}Z$$

Simplification of PoS Using Don't Care Conditions

- Let's use the same example shown in the previous slide
  - Step 1: combine 0's using don't care conditions
  - Step 2: take the complement of $\bar{F}$, then we get F

Step 1

WX \ YZ	00	01	11	10
00	X	1	1	X
01	0	X	1	0
11	0	0	1	0
10	0	0	1	0

Y

X

W

Z

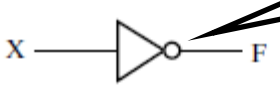
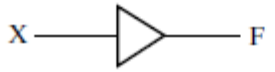
$$\bar{F} = \bar{Z} + W\bar{Y}$$

Step 2

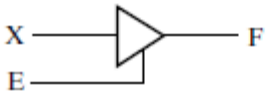
$$F = \overline{\bar{Z} + W\bar{Y}}$$
$$= Z(\bar{W} + Y)$$

Primitive Logic Gates

- So far, the Boolean functions are expressed using only AND, OR, NOT operations/gates (SoP or PoS)
- There exist other logic gates!

Name	Distinctive-Shape Graphics Symbol	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		$F = X$	X | F 0 | 0 1 | 1

Bubble: negation indicator

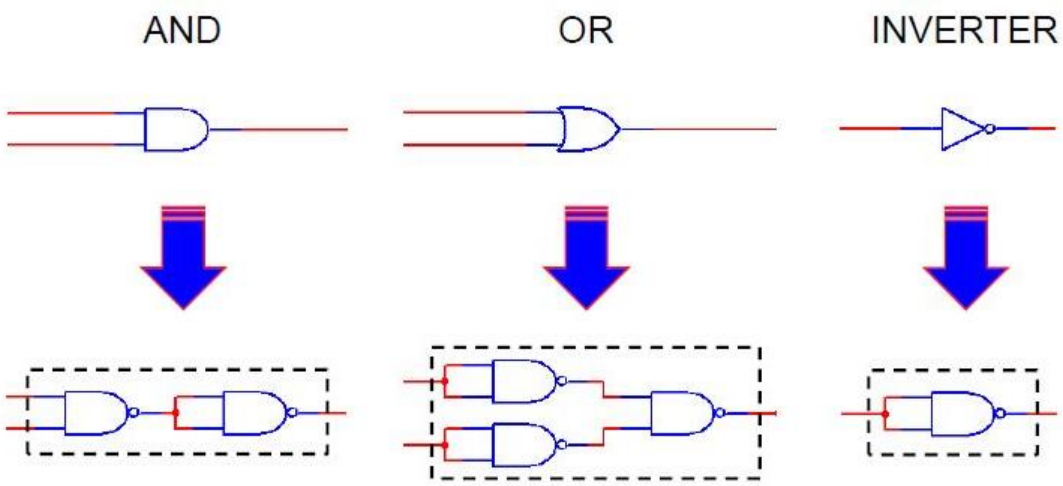
Name	Distinctive-Shape Graphics Symbol	Algebraic Equation	Truth Table
3-State Buffer			E X | F 0 0 | Hi-Z 0 1 | Hi-Z 1 0 | 0 1 1 | 1
NAND		$F = \overline{X \cdot Y}$	X Y | F 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR		$F = \overline{X + Y}$	X Y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0

IEEE Standard Graphic Symbols for Logic Functions

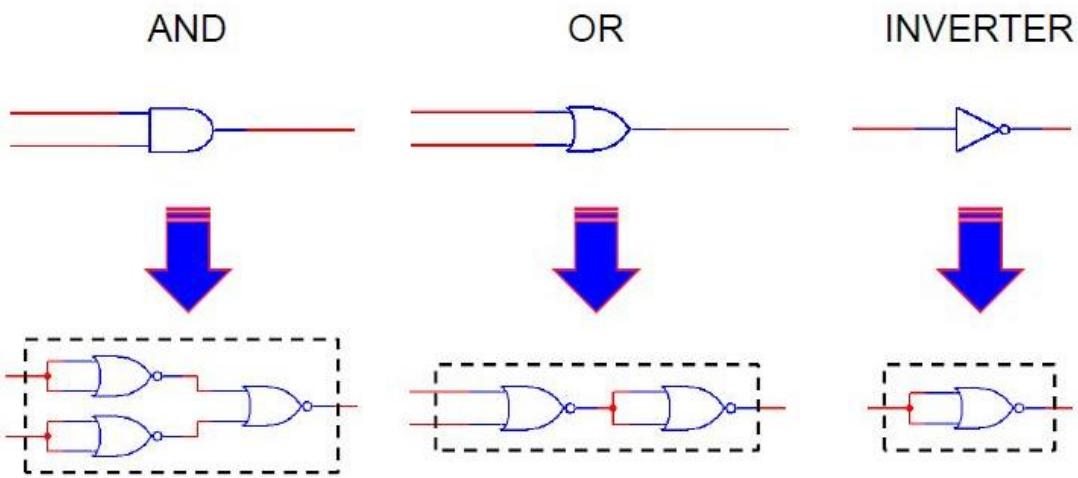
Side Notes on Primitive Logic Gates

- NAND & NOR are natural primitive (or Universal) gates for the simplest and fastest electronic circuits
- WHY?
  - Any digital logic can be implemented by using NAND and NOR gates
  - Also, we can also reduce delay/area of circuits

NAND equivalent of AOI gates



NOR equivalent of AOI gates



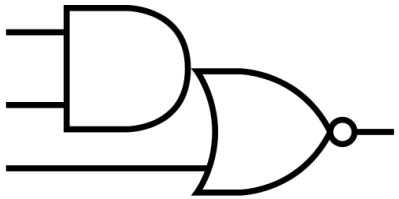
Complex Logic Gates

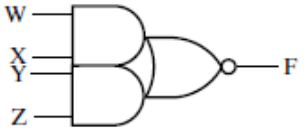
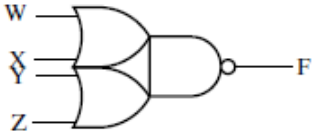
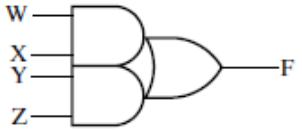
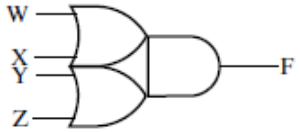
- XOR (exclusive-OR) gate
  - Same input values evaluates to 0
  - Different input values evaluates to 1

$$F = X \oplus Y$$

- XNOR gate
  - Complement of XOR gate

- AOI gate
  - Ex: 2-1 AOI

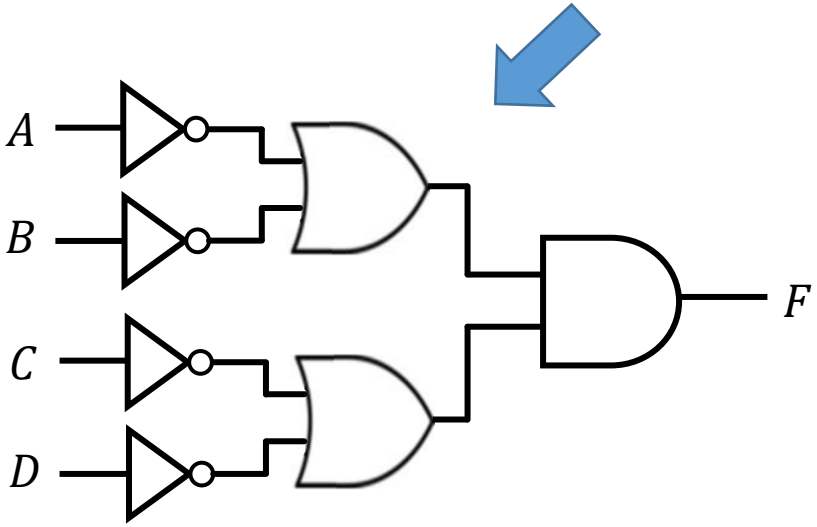


Name	Distinctive-Shape Graphics Symbol	Algebraic Equation	Truth Table
Exclusive-OR (XOR)		$F = X\overline{Y} + \overline{X}Y = X \oplus Y$	X Y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0
Exclusive-NOR (XNOR)		$F = XY + \overline{X}\overline{Y} = X \oplus Y$	X Y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1
AND-OR-INVERT (AOI)		$F = \overline{WX + YZ}$	
OR-AND -INVERT (OAI)		$F = \overline{(W + X)(Y + Z)}$	
AND-OR (AO)		$F = WX + YZ$	
OR-AND (OA)		$F = (W + X)(Y + Z)$	

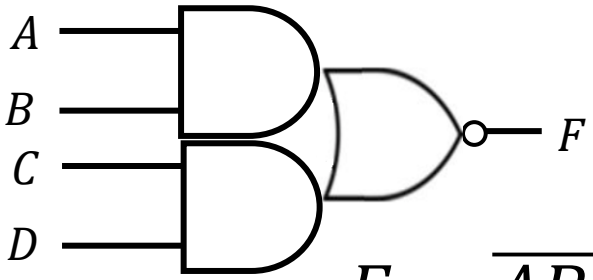
Bubble Pushing

- Assume we have the following Boolean function

$$F = (\bar{A} + \bar{B})(\bar{C} + \bar{D})$$



AOI equivalent



$$F = \overline{AB + CD}$$

Exclusive-OR Operator

- Algebraic identities of XOR
  - Equal to 1 if exactly one input variable is equal to 1

$$X \oplus Y = X\overline{Y} + \overline{X}Y$$

Name	Distinctive-Shape Graphics 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

Can be used for an adder

Identities of XOR

$$X \oplus 0 = X \qquad X \oplus 1 = \overline{X}$$

$$X \oplus X = 0 \qquad X \oplus \overline{X} = 1$$

$$X \oplus \overline{Y} = \overline{X \oplus Y} \qquad \overline{X} \oplus Y = \overline{X \oplus Y}$$

Exclusive-OR Operator

- Exclusive-NOR: known as the *equivalence*
  - Equal to 1 if both *X* and *Y* are equal to 1 or if both are equal to 0

$$\overline{X \oplus Y} = \overline{X\overline{Y} + \overline{X}Y} = (\overline{X} + Y)(X + \overline{Y}) = XY + \overline{X}\,\overline{Y}$$

Name	Distinctive-Shape Graphics Symbol	Algebraic Equation	Truth Table
Exclusive-NOR (XNOR)		$F = XY + \overline{X}\,\overline{Y}$ $= X \oplus Y$	X Y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1

Odd Function

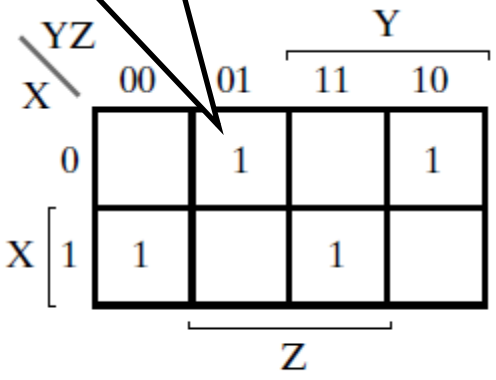
- Let's consider XOR operation with three or more variables
- For three variables, a Boolean expression is as follows:

$$X \oplus Y \oplus Z = (X\overline{Y} + \overline{X}Y)\overline{Z} + (XY + \overline{X}\,\overline{Y})Z$$

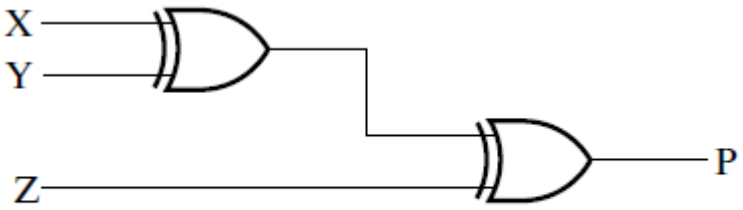
$$= X\overline{Y}\,\overline{Z} + \overline{X}Y\overline{Z} + \overline{X}\,\overline{Y}Z + XYZ$$

Distance 2 from each other

Odd number of variables needs to be 1

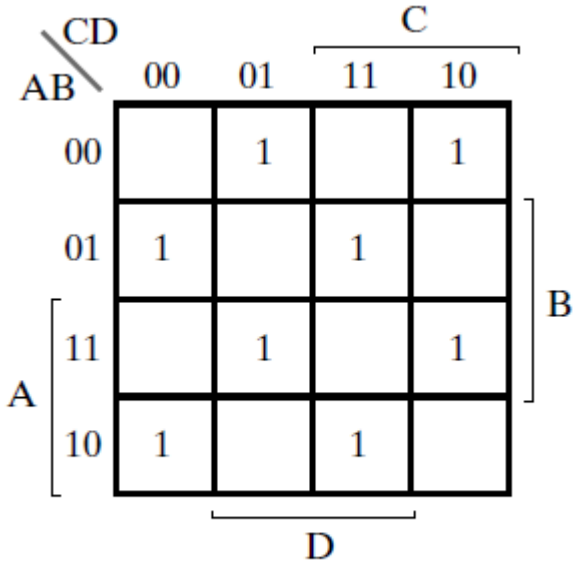


(a) X ⊕ Y ⊕ Z



(a) P = X ⊕ Y ⊕ Z

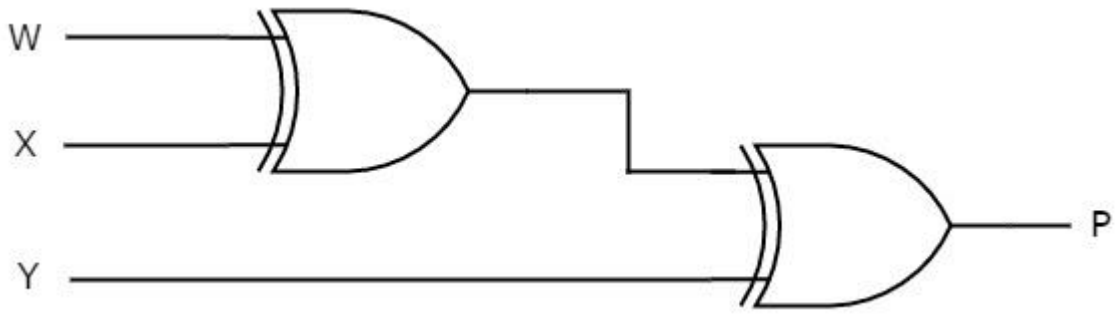
A map with 4 variables



(b) A ⊕ B ⊕ C ⊕ D

Odd Function Example – Parity Generator / Checker

- Even Parity Generator

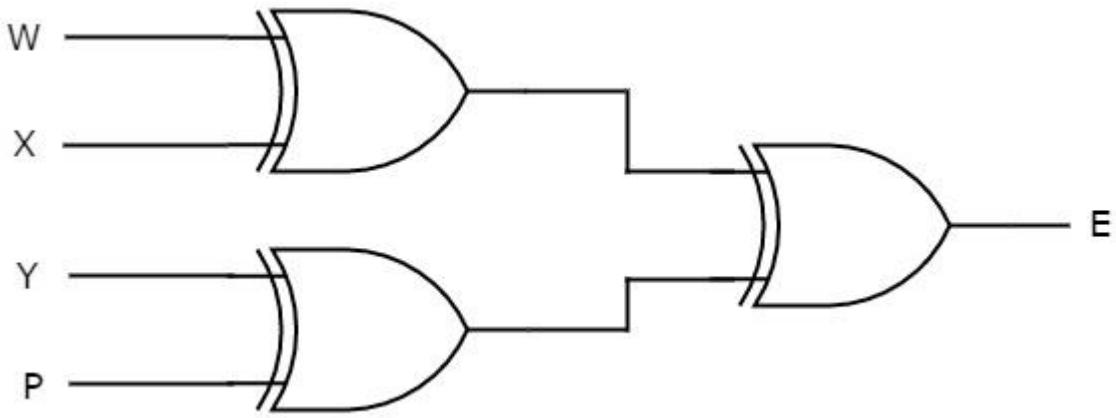


$$E = W \oplus X \oplus Y$$

Binary Input WXY	Even Parity bit P
000	0
001	1
010	1
011	0
100	1
101	0
110	0
111	1

Odd Function Example – Parity Generator / Checker

- Even Parity Checker



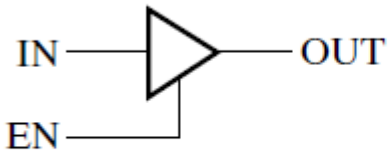
$$E = W \oplus X \oplus Y \oplus P$$

4-bit Received Data WXYP	Even Parity Check bit E
0000	0
0001	1
0010	1
0011	0
0100	1
0101	0
0110	0
0111	1
1000	1
1001	0
1010	0
1011	1
1100	0
1101	1
1110	1
1111	0

High Impedance Outputs (Hi-Z)

- So far, we only considered gates with output ‘0’ or ‘1’
- The third output value: high-impedance output (≃open-circuit)
- What are useful properties??
  - Can be connected together if they do not disagree at the same time
  - Can act as both an output and an input (bidirectional in/out)
    - Carry information in both directions → significantly reduces # of interconnections

3-state buffer

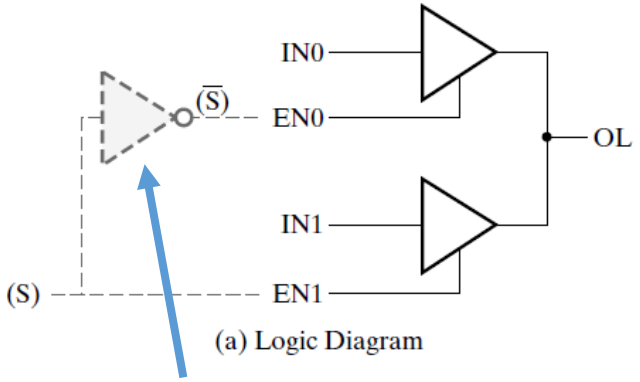


(a) Logic symbol

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

(b) Truth table

Forming multiplexed line OL



(a) Logic Diagram

EN1	EN0	IN1	IN0	OL
0	0	X	X	Hi-Z
(S) 0	(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

Use decoder to generate EN signal

High Impedance Outputs (Hi-Z) – Bidirectional Buffer

TEXAS INSTRUMENTS

SN74LVC4245A

SCAS375I – MARCH 1994 – REVISED JANUARY 2015

SN74LVC4245A Octal Bus Transceiver and 3.3-V to 5-V Shifter With 3-State Outputs

1 Features

- Bidirectional Voltage Translator
- 5.5 V on A Port and 2.7 V to 3.6 V on B Port
- Control Inputs V_{IH}/V_{IL} Levels Are Referenced to V_{CCA} Voltage
- Latch-Up Performance Exceeds 250 mA Per JESD 17
- ESD Protection Exceeds JESD 22
  - 2000-V Human-Body Model
  - 200-V Machine Model
  - 1000-V Charged-Device Model

2 Applications

- ATCA Solutions
- CPAP Machines
- Cameras: Surveillance Analog
- Chemical or Gas Sensors
- CT Scanners
- DLP 3D Machine Vision and Optical Networking
- Digital Signage
- ECGs: Electrocardiograms
- Field Transmitters: Pressure Sensors and Temperature Sensors
- High-Speed Data Acquisition and Generation
- HMI (Human Machine Interface)
- RF4CE Remote Controls
- Server Motherboards
- Software Defined Radios (SDR)
- Wireless LAN Cards and Data Access Cards
- X-ray: Medical, Dental, and Baggage Scanners

3 Description

This 8-bit (octal) noninverting bus transceiver contains two separate supply rails; B port has V_{CCB}, which is set at 3.3 V, and A port has V_{CCA}, which is set at 5 V. This allows for translation from a 3.3-V to a 5-V environment, and vice versa.

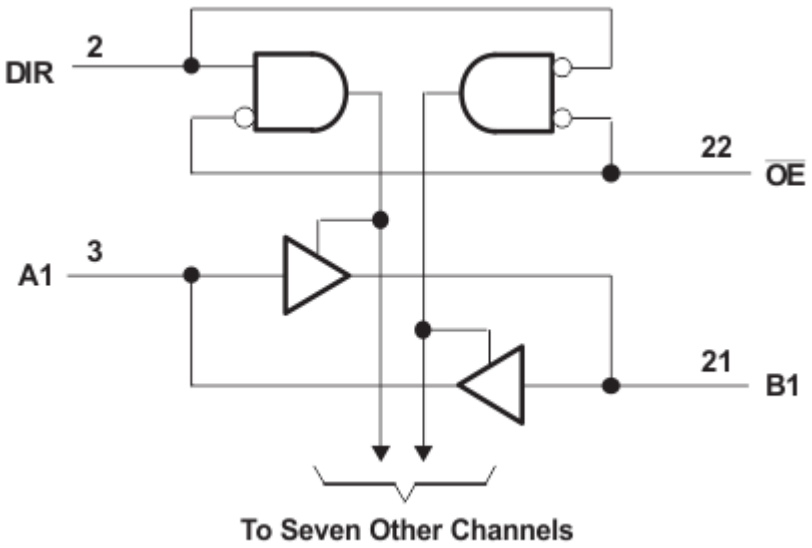
The SN74LVC4245A device is designed for asynchronous communication between data buses. The device transmits data from the A bus to the B bus or from the B bus to the A bus, depending on the logic level at the direction-control (DIR) input. The output-enable ($\overline{OE}$) input can be used to disable the device so the buses are effectively isolated. The control circuitry (DIR, $\overline{OE}$) is powered by V_{CCA}.

The SN74LVC4245A device terminal out allows the designer to switch to a normal all-3.3-V or all-5-V 20-terminal SN74LVC4245 device without board re-layout. The designer uses the data paths for pins 2–11 and 14–23 of the SN74LVC4245A device to align with the conventional '245 terminal out.

Device Information(1)

PART NUMBER	PACKAGE	BODY SIZE (NOM)
SN74LVC4245A	SSOP (24)	8.20 mm × 5.30 mm
	SOIC (24)	15.40 mm × 7.50 mm
	TSSOP (24)	7.80 mm × 4.40 mm

(1) For all available packages, see the orderable addendum at the end of the data sheet.

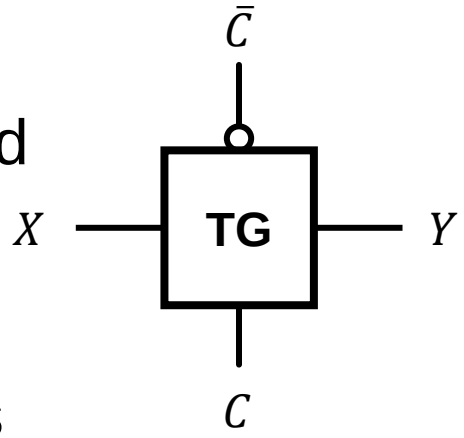


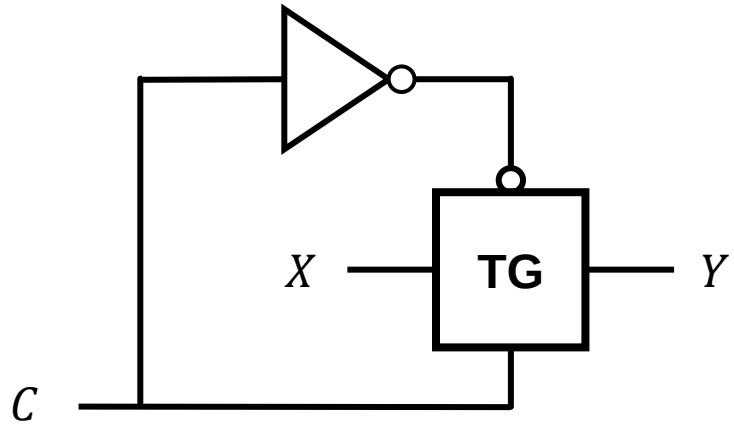
Function Table

INPUTS		OPERATION
$\overline{OE}$	DIR	
L	L	B data to A bus
L	H	A data to B bus
H	X	Isolation

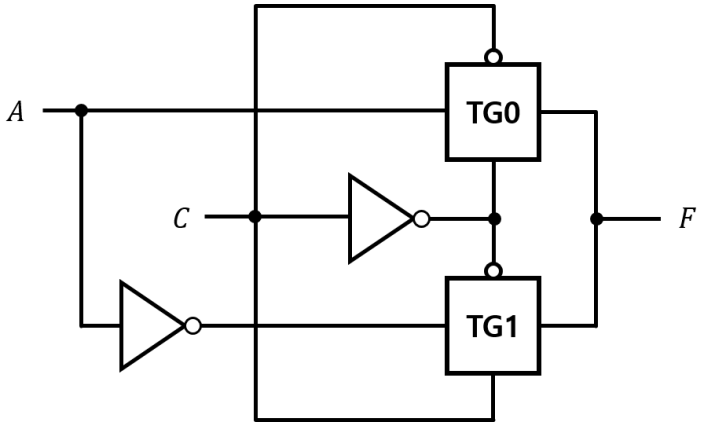
Transmission Gates (TG)

- An electronic switch for connecting and disconnecting two points in a circuit
  - If C = 1 & C' = 0, then *Y* outputs *X*
  - Otherwise, *X* and *Y* are disconnected





- Use of transmission gate
  - XOR gate constructed from two TGs



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

Summary

- Boolean algebra allows us to represent and manipulate binary logics
  - The standard forms: sum-of-products, product-of-sums
  - Logic minimization using Karnaugh map (K-map)
- The systematic logic reduction
  - The concept of prime implicant & essential prime implicant
- Various logic gates in designing digital circuits
  - Primitive logic gates
  - Complex logic gates