

CSEN  
605

Lecture  
**1**

# Digital Systems Design

## Review



# **Part 0**

# **Course content and rules**

# Personal information

- Dr. Shereen Moataz Afifi
  - Email: [shereen.moataz@guc.edu.eg](mailto:shereen.moataz@guc.edu.eg)
  - Office: C7-211
  - Day off: Saturday
- 
- **TAs**
  - Noura Gamal
  - Manar Hatem
  - Nada Alazab

# Course content

Digital Logic Design review	2	
Ch2, 3-Introduction to VHD, CAD tools and FPGA	2	2
Ch5- Numbers, adders, subtractors, multipliers (1)	2	2
Ch5- Numbers, adders, subtractors, multipliers (2)	2	2
Ch6- Combinational circuits	2	2
Ch6- Combinational circuits	2	2
Ch7- Sequential circuits	2	2
Ch7- Sequential circuits	2	2
Appendix. B, C- FPGA tools: Quartus and Altera board.	2	2
Ch8- Synchronous sequential circuits (FSM)	2	2
Ch8- ASM and Digital system design	2	2
Ch10- Advances topics	2	2

- Core course for computer engineers
- Important for creating digital systems with inputs and outputs, and creating useful projects in many areas

**Textbook: Fundamentals of Digital Logic with VHDL Design, THIRD EDITION by Stephen Brown and Zvonko Vranesic [ISBN-10 : 0077221435-ISBN-13 : 978-0077221430]**

# Course grading and rules

Assessment	
Student assessment methods	Assessment weighting
Quizzes	10%
Assignments	5%
Project	20%
Midterm Exam	25%
Final Exam	40%

- Project will be done on Altera FPGA board available at the university. Other parts might be purchased.
- Attendance in lectures is highly recommended to get the most out of the course
- Schedule and Office hours: TBA

# Lecture contents

1. Introduction to digital systems
2. Numbers
3. Combinational systems
4. Combinational circuit blocks
5. Sequential systems

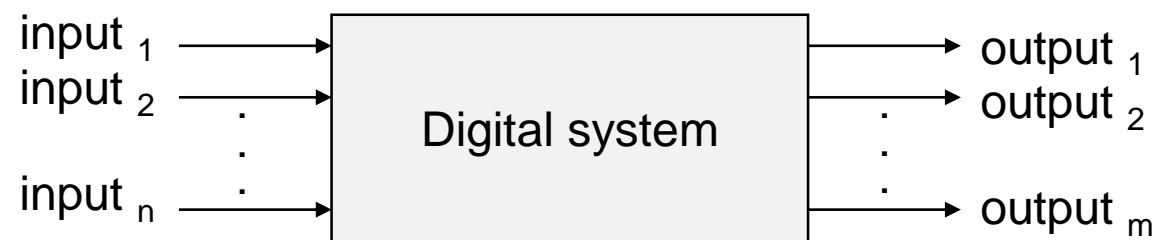
# **Part 1**

# **Introduction to digital systems**

# What is a digital system?

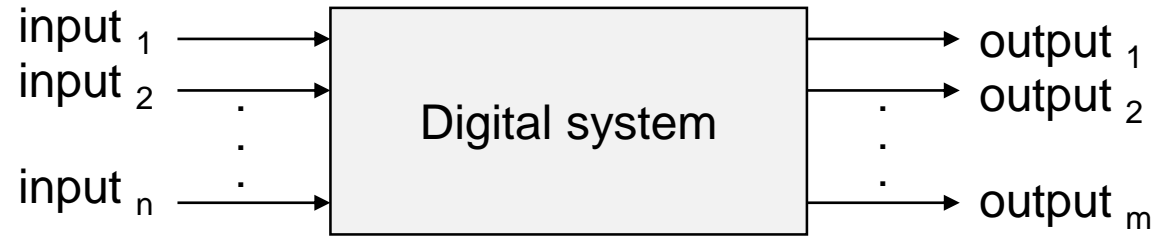


- Devices we use on a daily basis such as computers, smart phones, tablets, laptops, calculators, digital camera,...
- Systems that have inputs and outputs depending on the inputs.
- Can be used in many scientific, industrial and commercial applications.





# What is a digital system? (2)



- Examples include simple systems like adders, multipliers, traffic light controllers and so on.
- Some systems are:
  - **combinational** (output depend only on inputs)
  - or **sequential** (with a clock and the output depends on the input and previous output)

## **Part 2**

# **Numbers**

# Number systems

- Any integer  $N$  can be represented using the following:

$$N = a_{n-1} r^{n-1} + a_{n-2} r^{n-2} + \dots + a_2 r^2 + a_1 r^1 + a_0$$

- $n$  is the number of digits
- $r$  is the radix or base (Decimal is base ten for example)
- $0 \leq a_i < r$

## Example

- The number 17 in decimal:
- $n = 2$  (2 digits)
- $r = 10$  (Base)
- $a$  can be between 0 and 9
- $17 = a_1 10^1 + a_0 = 1 \times 10 + 7$

# Binary numbers

- Base 2
- Digits can only be 0 or 1
- To convert the number **101111** from binary to decimal, we apply the formula in the previous slide:
- $1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 32+0+8+4+2+1= 47$
- An easier way is to memorize the places in powers of 2:

	<b>1</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>	
binary number →							
<b>Places</b> →	<b>32</b>	<b>16</b>	<b>8</b>	<b>4</b>	<b>2</b>	<b>1</b>	<b>Total = 1+2+4+8+32 = 47</b>

# Binary numbers

Most significant bit (MSB)      Least significant bit (LSB)

**1   0   1   1   1   1**

Bit 5      Bit 0

The diagram shows a 6-bit binary number '101111'. Above the first bit '1' is the label 'Most significant bit (MSB)' with a blue arrow pointing to it. Above the last bit '1' is the label 'Least significant bit (LSB)' with a blue arrow pointing to it. Below the first bit '1' is the label 'Bit 5', and below the last bit '1' is the label 'Bit 0'.

# Question

- **Question:** What's the largest decimal number that be represented using 3-bit binary digits?
- **Answer:** The largest is when all digits are 1s: 111
- This equals:  $4 + 2 + 1 = 7$

# Quick way to know your binary numbers

3-bit binary	Decimal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

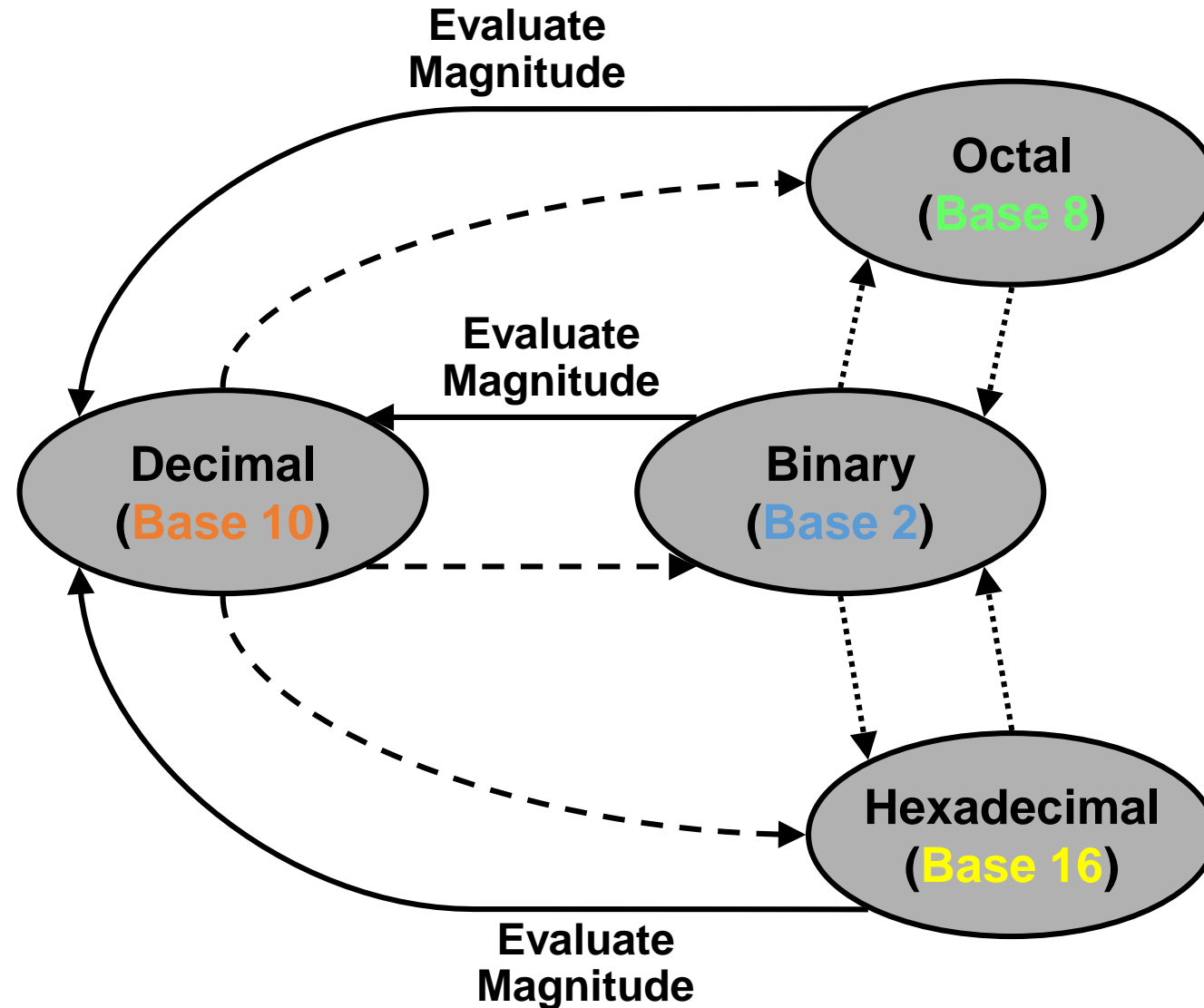
4-bit binary	Decimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
...	
1110	14
1111	15

# Decimal, Binary, Octal and Hexadecimal

Decimal	Binary	Octal	Hex
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F



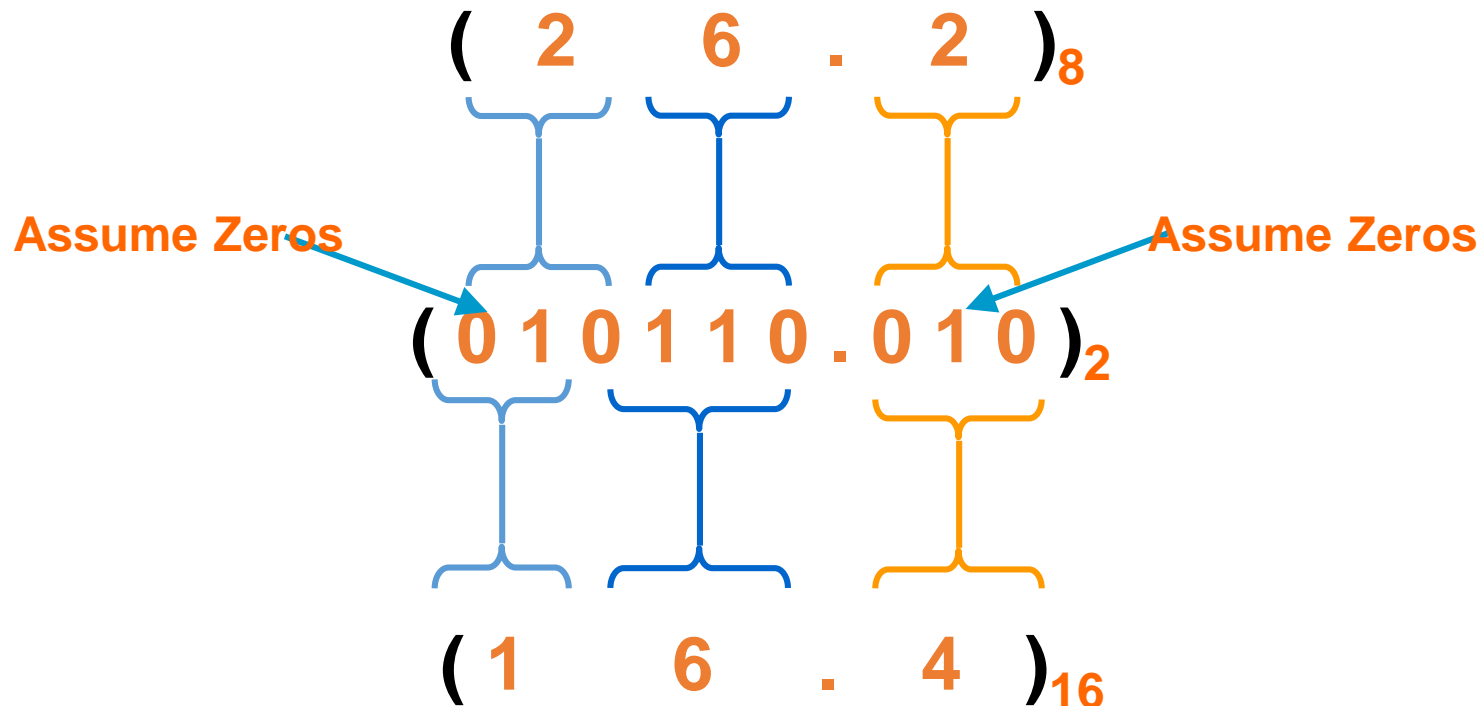
# Number Base Conversions



# Octal – Hexadecimal Conversion

- Convert to **Binary** as an intermediate step

**Example:**



Works **both** ways (Octal to Hex & Hex to Octal)


# Decimal (*Integer*) to Binary Conversion

- Divide the number by the 'Base' (=2)
- Take the remainder (either 0 or 1) as a coefficient
- Take the quotient and repeat the division

**Example:**  $(13)_{10}$

	Quotient	Remainder	Coefficient
<b>13</b> / <b>2</b> =	<b>6</b>	<b>1</b>	<b>a<sub>0</sub> = 1</b>
<b>6</b> / <b>2</b> =	<b>3</b>	<b>0</b>	<b>a<sub>1</sub> = 0</b>
<b>3</b> / <b>2</b> =	<b>1</b>	<b>1</b>	<b>a<sub>2</sub> = 1</b>
<b>1</b> / <b>2</b> =	<b>0</b>	<b>1</b>	<b>a<sub>3</sub> = 1</b>

**Answer:**  $(13)_{10} = (a_3 a_2 a_1 a_0)_2 = (1101)_2$

  
MSB                      LSB


# Decimal (*Fraction*) to Binary Conversion

- Multiply the number by the 'Base' (=2)
- Take the integer (either 0 or 1) as a coefficient
- Take the resultant fraction and repeat the division

Example:  $(0.625)_{10}$

		Integer	Fraction	Coefficient
$0.625$	$* 2 =$	1	. 25	$a_{-1} = 1$
$0.25$	$* 2 =$	0	. 5	$a_{-2} = 0$
$0.5$	$* 2 =$	1	. 0	$a_{-3} = 1$

Answer:  $(0.625)_{10} = (0.a_{-1}a_{-2}a_{-3})_2 = (0.101)_2$



# Example

- Convert the decimal number 25.35 to binary

2	25	
2	12	1 ← First remainder
2	6	0 ← Second Remainder
2	3	0 ← Third Remainder
2	1	1 ← Fourth Remainder
	0	1 ← Fifth Reaminder

Read Up

Binary Number = 11001

$$\begin{array}{lcl}
 0.35 \times 2 & = & 0.70 \\
 0.70 \times 2 & = & 0.40 \\
 0.40 \times 2 & = & 0.80 \\
 0.80 \times 2 & = & 0.60 \\
 0.60 \times 2 & = & 0.20
 \end{array}$$

with a carry of  
with a carry of  
with a carry of  
with a carry of  
with a carry of

	Binary Point
0	
1	
0	
1	
1	

Read Down

Circuit Globe

The fractional binary number = .01011

$$(25.35)_{10} = (11001.01011)_2$$

# Example

- Convert the decimal number 49 into binary by using the sum-of-weights method

- Write the decimal weight of each column
- Place 1's in the columns that sum to the decimal number

$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
64	32	16	8	4	2	1
0	1	1	0	0	0	1

Decimal 49 = binary 110001

Write:  $49_{10} = 110001_2$

# Binary addition

$$\begin{array}{r} 0 \\ + 0 \\ \hline 0 \end{array}$$

$$\begin{array}{r} 0 \\ + 1 \\ \hline 1 \end{array}$$

$$\begin{array}{r} 1 \\ + 0 \\ \hline 1 \end{array}$$

$$\begin{array}{r} 1 \\ + 1 \\ \hline \mathbf{c=1} \quad 0 \end{array} \quad \text{*c stands for carry}$$

if we concatenate carry to the result, we get (10) which is the right result of (1)+(1)

# Binary addition

$$\begin{array}{r} \phantom{+} \overset{\boxed{1}}{0} \overset{\boxed{1}}{1} 1 0 \\ + 0 1 1 1 \\ \hline 1 1 0 1 \end{array}$$

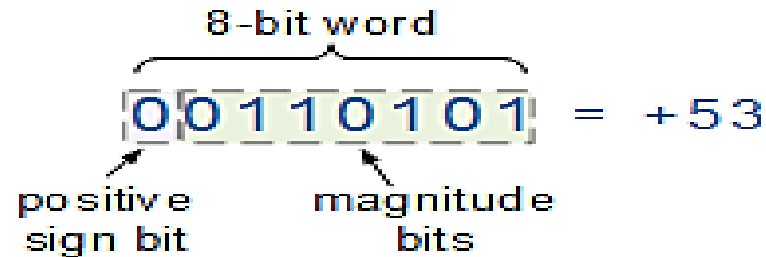
Generic form

$$\begin{array}{r} C_{in} \\ + a \\ + b \\ \hline C_{out} \quad S \end{array}$$



# 2's complement

- To represent negative numbers in binary, we use 2's complement:
  - Positive numbers are stored as is (MSB will be 0)
  - Negative numbers (MSB will be 1)



- To get the 2's complement of a number A: Flip (complement) and add 1:  **$(A'+1)$**
- **Example:** consider A= 0100 is number 4, to get -4, we do  $A'+1 = 1011 + 1 = 1100$

# 2's complement

- **Trick:** To quickly get 2's complement without flipping and adding 1, start from the right to the left, keep the digits as is until you hit the first 1, keep it also, but flip everything to its left.

1 0 1 1 0 0 0 0

0 1 0 1 0 0 0 0

- **Example re-visited:**  $A = 0100$   
→ we start from the right, we find 0, keep it, then 0 keep it, then 1 keep it but start flipping after, we get 1100
- Try it yourself: find the 2's complement of 0011 and 0010

# 2's complement

- How to compute the decimal number for a 2's complement binary number?
- If the MSB is 0, means the number is +ve, and we compute it in the regular way.
  - **Example: 0100 is 4**
- If the MSB is 1, means the number is -ve, so must find its 2's complement to know what was it.
  - **Example: 1100, we find its 2's complement which is 0100 which is 4 so the original number was -4**

# 2's complement

- A **quicker** way to get the decimal negative number from 2's complement representation is the following:
- Negative Binary number: 1 X X X X
- To get the decimal we do this:  $-2^{(\text{number of digits X})} + (\text{XXXX in normal binary})$
- **Example:**  $1100 \rightarrow -2^{(3)} + (100_{\text{base } 2} = 4_{\text{base } 10}) = -8 + 4 = -4$

# Binary subtraction

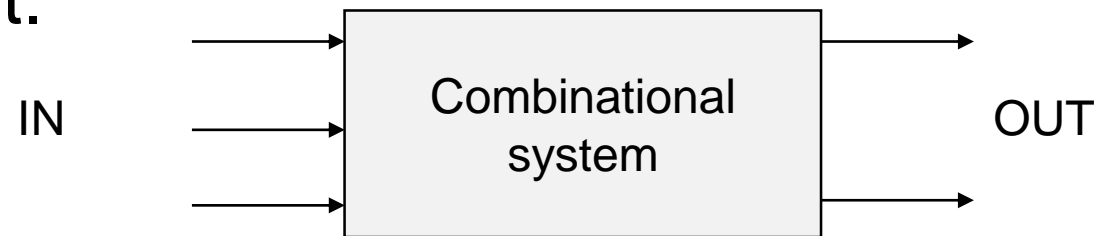
- $(A-B) = (A) + (-B) = (A) + (2\text{'s complement of } B)$

## **Part 3**

# **Combinational systems**

# Combinational systems

- A system consists of logic gates with a set of inputs and a set of outputs such that the output reacts to a certain combination of inputs.
- **Example:** An adder that adds the input bits and generates the sum as output.



What are the internal components of such a system to perform the way we want it to perform?

# Truth table

- A table that shows the output value(s) depending on the input values.
- **Example:** the truth table of 1-bit ***adder*** with 2 inputs

inputs		Output
A	B	Sum
0	0	0
0	1	1
1	0	1
1	1	0



# Truth table (cont.)

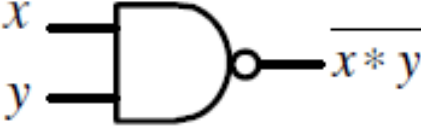
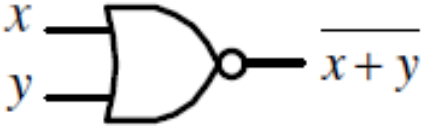
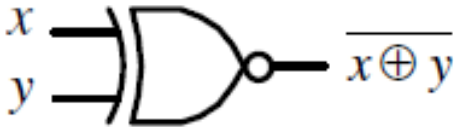
- *Don't cares*: represented as X
- Used when the output value can be either 0 or 1 for a certain input combination, in other words, we don't care
- **Example**: System that has 2 inputs A and B, and output Z such that Z should be 1, if any of the inputs is 1, the other input must be zero, otherwise  $z=0$ . But when both are zeros, we don't care about the output.

A	B	Z
0	0	X
0	1	1
1	0	1
1	1	0

# Basic Logic Gates

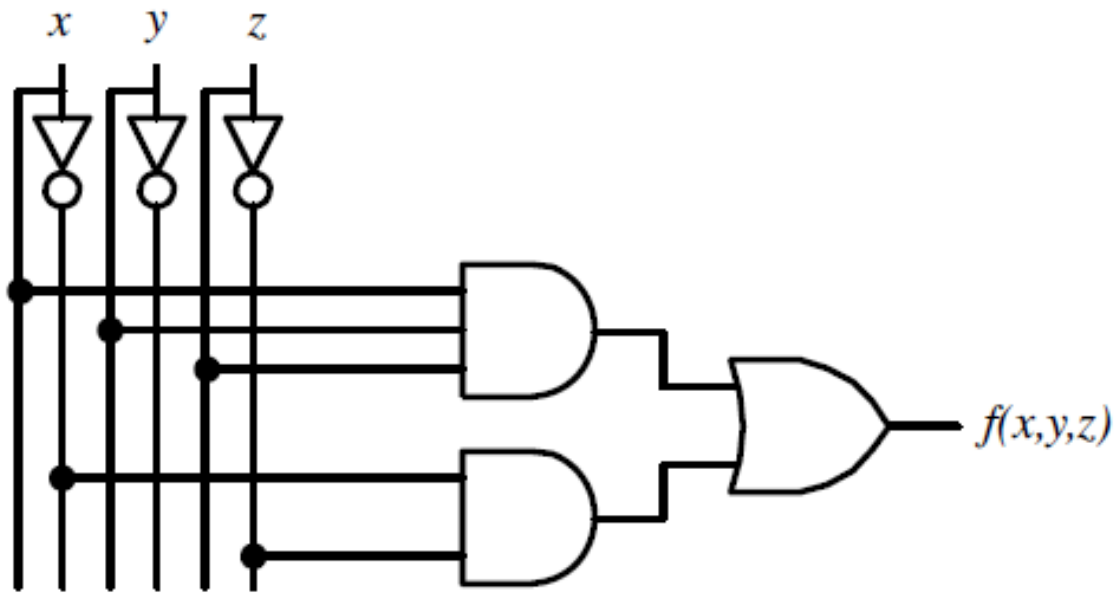
Truth Table	Function	Symbol															
<table><tr><th><math>x</math></th><th><math>\bar{x}</math></th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	$x$	$\bar{x}$	0	1	1	0	NOT	$x \rightarrow \bar{x}$									
$x$	$\bar{x}$																
0	1																
1	0																
<table><tr><th><math>x</math></th><th><math>y</math></th><th><math>x * y</math></th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	$x$	$y$	$x * y$	0	0	0	0	1	0	1	0	0	1	1	1	AND	$x \text{ AND } y \rightarrow x * y$
$x$	$y$	$x * y$															
0	0	0															
0	1	0															
1	0	0															
1	1	1															
<table><tr><th><math>x</math></th><th><math>y</math></th><th><math>x + y</math></th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	$x$	$y$	$x + y$	0	0	0	0	1	1	1	0	1	1	1	1	OR	$x \text{ OR } y \rightarrow x + y$
$x$	$y$	$x + y$															
0	0	0															
0	1	1															
1	0	1															
1	1	1															
<table><tr><th><math>x</math></th><th><math>y</math></th><th><math>x \oplus y</math></th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	$x$	$y$	$x \oplus y$	0	0	0	0	1	1	1	0	1	1	1	0	XOR	$x \text{ XOR } y \rightarrow x \oplus y$
$x$	$y$	$x \oplus y$															
0	0	0															
0	1	1															
1	0	1															
1	1	0															

# Basic Logic Gates (cont.)

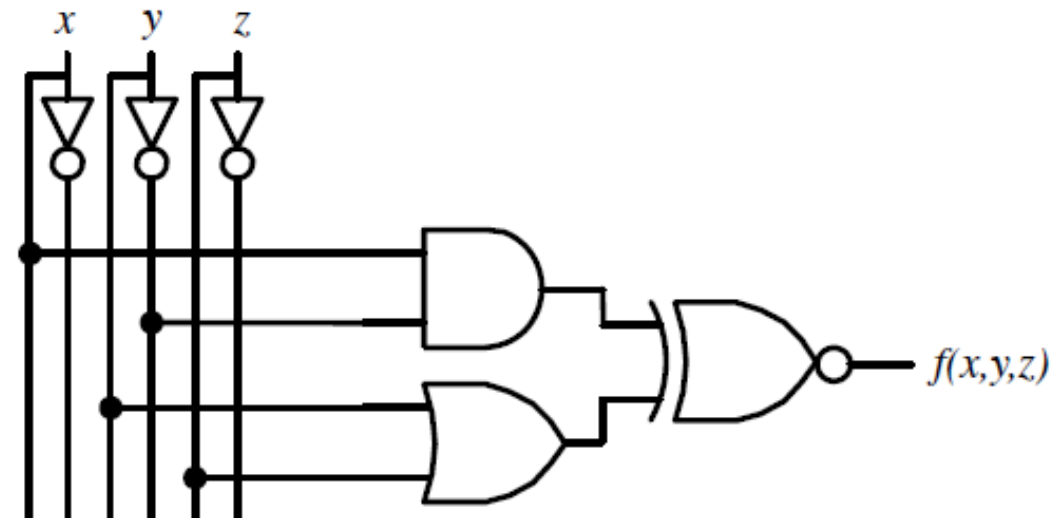
Truth Table	Function	Symbol															
<table><tr><th><math>x</math></th><th><math>y</math></th><th><math>\overline{x * y}</math></th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	$x$	$y$	$\overline{x * y}$	0	0	1	0	1	1	1	0	1	1	1	0	NAND	
$x$	$y$	$\overline{x * y}$															
0	0	1															
0	1	1															
1	0	1															
1	1	0															
<table><tr><th><math>x</math></th><th><math>y</math></th><th><math>\overline{x + y}</math></th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	$x$	$y$	$\overline{x + y}$	0	0	1	0	1	0	1	0	0	1	1	0	NOR	
$x$	$y$	$\overline{x + y}$															
0	0	1															
0	1	0															
1	0	0															
1	1	0															
<table><tr><th><math>x</math></th><th><math>y</math></th><th><math>\overline{x \oplus y}</math></th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	$x$	$y$	$\overline{x \oplus y}$	0	0	1	0	1	0	1	0	0	1	1	1	NXOR	
$x$	$y$	$\overline{x \oplus y}$															
0	0	1															
0	1	0															
1	0	0															
1	1	1															

# Using logic gates to build a combinational circuit

$$f(x, y, z) = x \cdot y \cdot z + \bar{x} \cdot \bar{z}$$

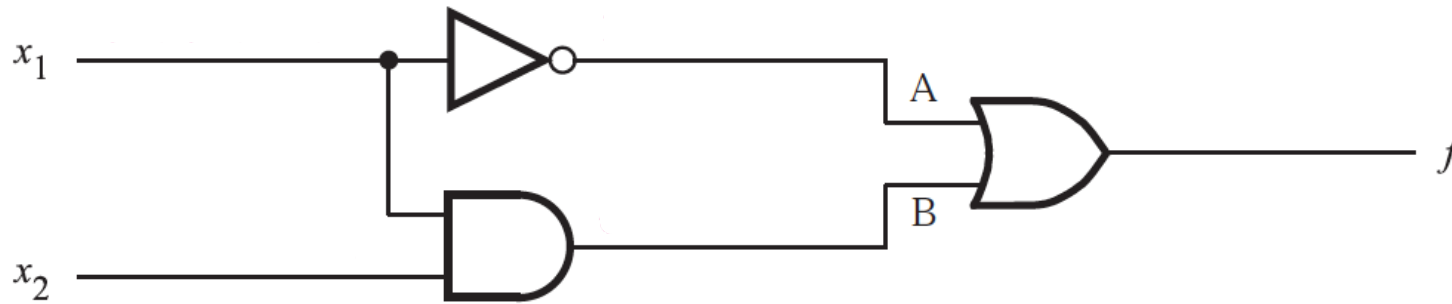


$$f(x, y, z) = \overline{(x \cdot \bar{y}) \oplus (y + z)}$$



# Timing diagram

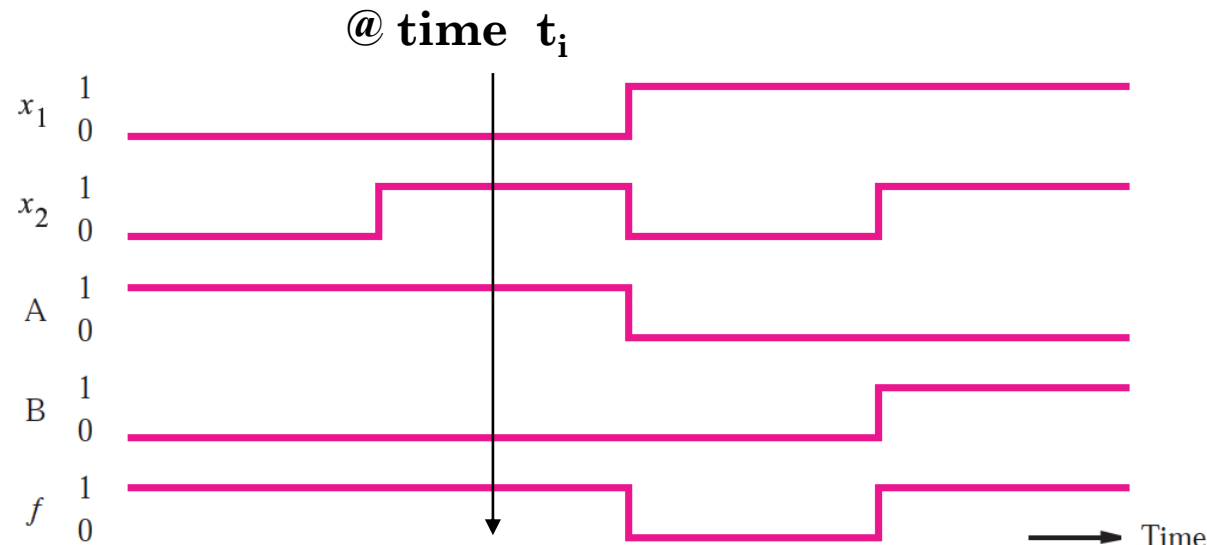
- Consider the following circuit with 2 inputs  $x_1$  and  $x_2$ , one output  $f = x_1' + x_1 x_2$
- A and B are intermediary outputs



$x_1$	$x_2$	$f(x_1, x_2)$	A	B
0	0	1	1	0
0	1	1	1	0
1	0	0	0	0
1	1	1	0	1

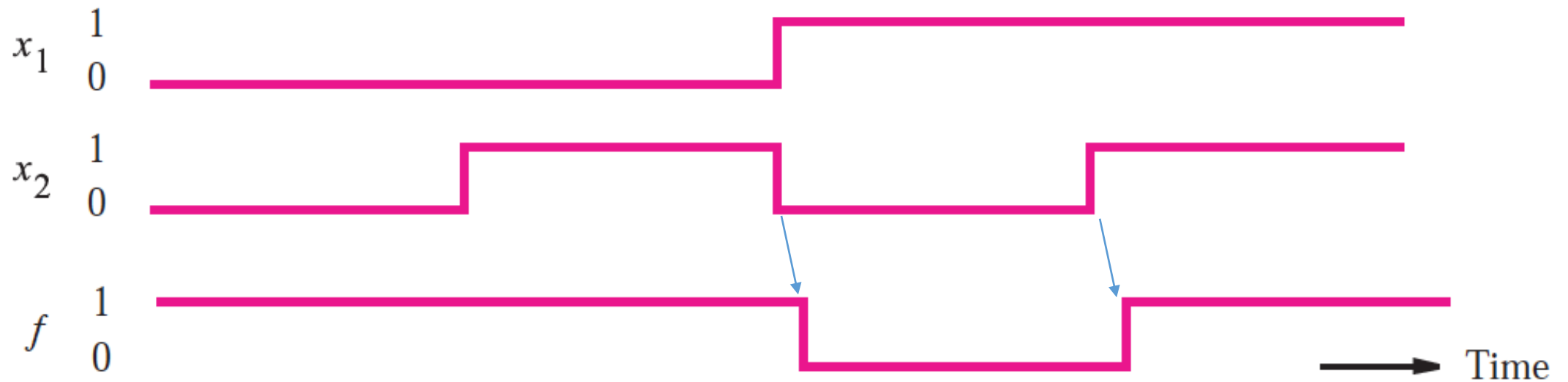
(b) Truth table

Timing diagram



(c) Timing diagram

# In a more realistic situation



(c) Timing diagram

- Why do you think there's a delay in the output?

# Boolean algebra (1)

- Axioms of Boolean algebra

$$0 \cdot 0 = 0$$

$$1 + 1 = 1$$

$$1 \cdot 1 = 1$$

$$0 + 0 = 0$$

$$0 \cdot 1 = 1 \cdot 0 = 0$$

$$1 + 0 = 0 + 1 = 1$$

$$\text{If } x = 0, \text{ then } \bar{x} = 1$$

$$\text{If } x = 1, \text{ then } \bar{x} = 0$$

- Single variable theorem

$$x \cdot 0 = 0$$

$$x + 1 = 1$$

$$x \cdot 1 = x$$

$$x + 0 = x$$

$$x \cdot x = x$$

$$x + x = x$$

$$x \cdot \bar{x} = 0$$

$$x + \bar{x} = 1$$

$$\bar{\bar{x}} = x$$

# Boolean algebra (2)

- Two and three variable properties

$$x \cdot y = y \cdot x$$

*Commutative*

$$x + y = y + x$$

$$x \cdot (y \cdot z) = (x \cdot y) \cdot z$$

*Associative*

$$x + (y + z) = (x + y) + z$$

$$x \cdot (y + z) = x \cdot y + x \cdot z$$

*Distributive*

$$x + y \cdot z = (x + y) \cdot (x + z)$$

$$x + x \cdot y = x$$

*Absorption*



# Boolean algebra (3)

$$\overline{x \cdot y} = \bar{x} + \bar{y}$$

$$\overline{x + y} = \bar{x} \cdot \bar{y}$$

$$x + \bar{x} \cdot y = x + y$$

$$x \cdot (\bar{x} + y) = x \cdot y$$

*DeMorgan's theorem*

# Minterms and Maxterms

Minterm (standard product): an AND term consists of all literals in their normal form or in their complement form.

Maxterm (standard sum): an OR term

$x_1$	$x_2$	$x_3$	Minterm	Maxterm
0	0	0	$m_0 = \bar{x}_1\bar{x}_2\bar{x}_3$	$M_0 = x_1 + x_2 + x_3$
0	0	1	$m_1 = \bar{x}_1\bar{x}_2x_3$	$M_1 = x_1 + x_2 + \bar{x}_3$
0	1	0	$m_2 = \bar{x}_1x_2\bar{x}_3$	$M_2 = x_1 + \bar{x}_2 + x_3$
0	1	1	$m_3 = \bar{x}_1x_2x_3$	$M_3 = x_1 + \bar{x}_2 + \bar{x}_3$
1	0	0	$m_4 = x_1\bar{x}_2\bar{x}_3$	$M_4 = \bar{x}_1 + x_2 + x_3$
1	0	1	$m_5 = x_1\bar{x}_2x_3$	$M_5 = \bar{x}_1 + x_2 + \bar{x}_3$
1	1	0	$m_6 = x_1x_2\bar{x}_3$	$M_6 = \bar{x}_1 + \bar{x}_2 + x_3$
1	1	1	$m_7 = x_1x_2x_3$	$M_7 = \bar{x}_1 + \bar{x}_2 + \bar{x}_3$

# Sum of products (SoP)

- Any function  $f$  can be represented by a sum of **Minterms** that correspond to the rows in the truth table for which  $f = 1$ .
- The result is called sum of products.

## Example

$x_1$	$x_2$	$x_3$	$f(x_1, x_2, x_3)$
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	0

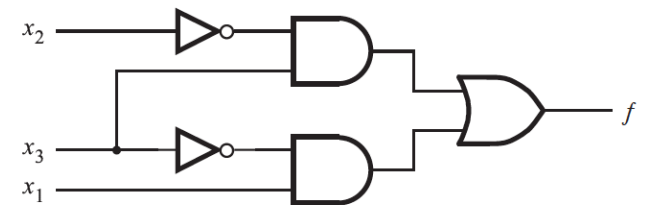
Looking at the 1s, we get this SoP

$$f(x_1, x_2, x_3) = \sum (m_1, m_4, m_5, m_6)$$

$$\Rightarrow f(x_1, x_2, x_3) = \bar{x}_1\bar{x}_2x_3 + x_1\bar{x}_2\bar{x}_3 + x_1\bar{x}_2x_3 + x_1x_2\bar{x}_3$$

We can further minimize it:

$$\begin{aligned} f &= (\bar{x}_1 + x_1)\bar{x}_2x_3 + x_1(\bar{x}_2 + x_2)\bar{x}_3 \\ &= 1 \cdot \bar{x}_2x_3 + x_1 \cdot 1 \cdot \bar{x}_3 \\ &= \bar{x}_2x_3 + x_1\bar{x}_3 \end{aligned}$$



# Product of sums (PoS)

- Any function  $f$  can be represented by a product of **Maxterms** that correspond to the rows in the truth table for which  $f = 0$ .
- The result is called product of sums.

## Example

$x_1$	$x_2$	$x_3$	$f(x_1, x_2, x_3)$
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	0 ←

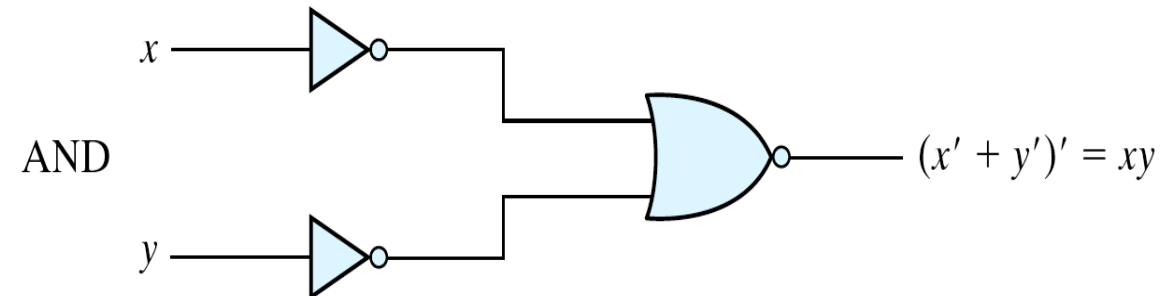
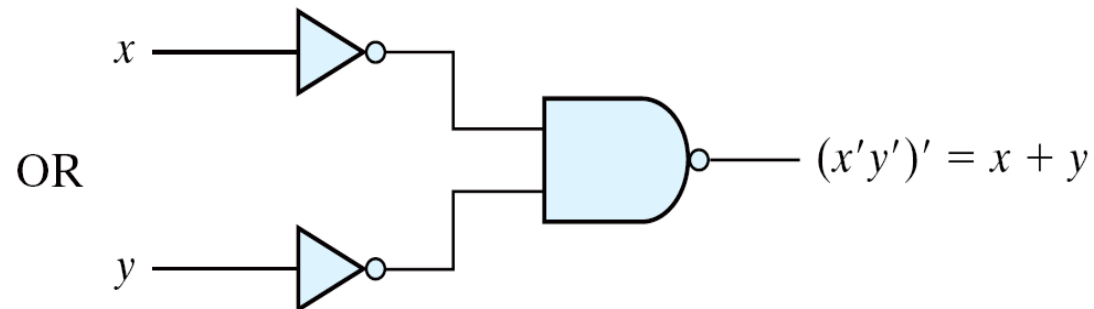
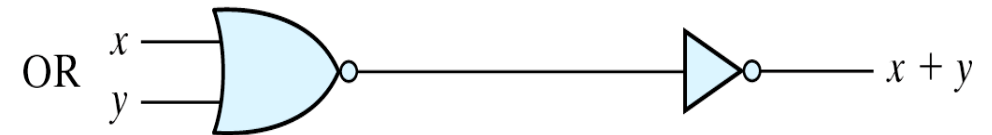
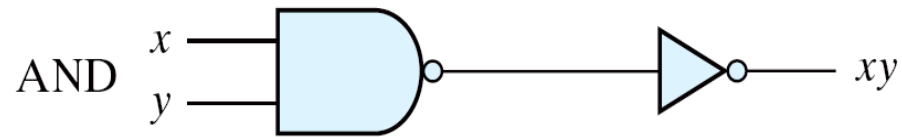
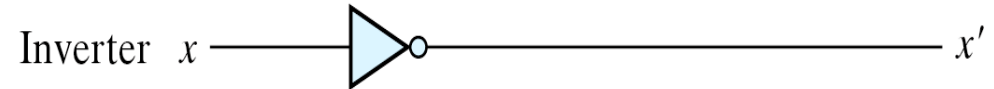
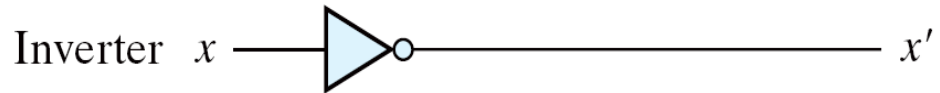
Looking at the 0s, we get this Pos

$$f(x_1, x_2, x_3) = \Pi(M_0, M_2, M_3, M_7)$$

$$\Rightarrow = (x_1 + x_2 + x_3)(x_1 + \bar{x}_2 + x_3)(x_1 + \bar{x}_2 + \bar{x}_3)(\bar{x}_1 + \bar{x}_2 + \bar{x}_3)$$

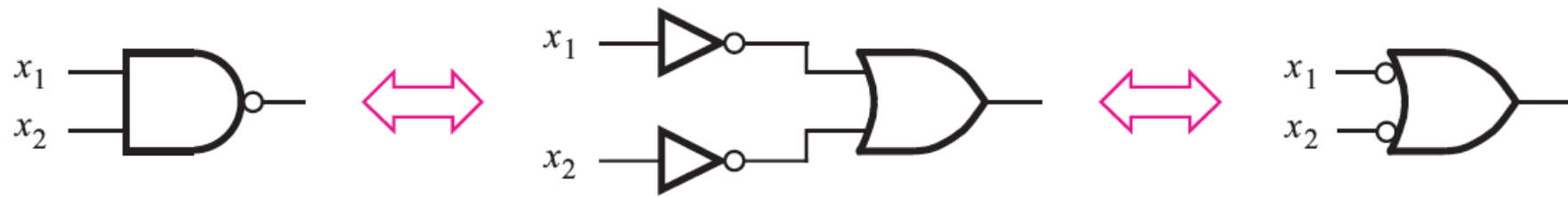
# Using NAND and NOR in implementations

- We can implement all other gates using either all NANDs or all NORs (universal gates)
- This is actually more efficient when it comes to hardware implementation if we can limit the use of gate types.

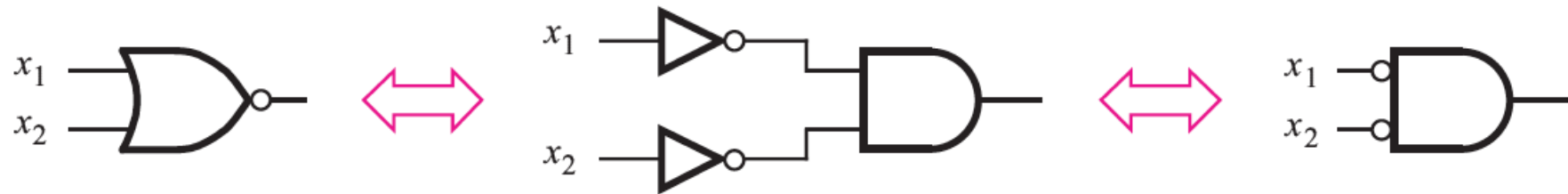


# Using NAND and NOR in implementations

- Using Demorgan's theorem we can do the following:

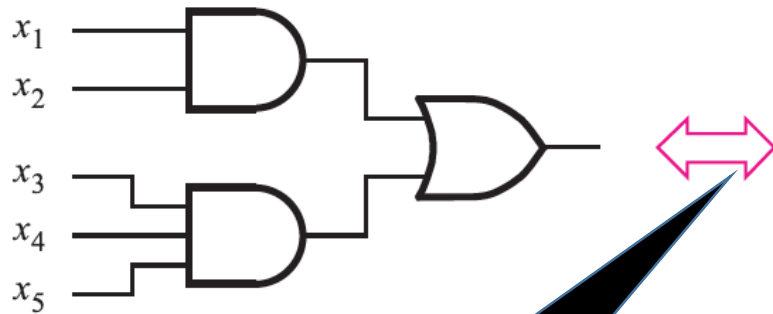


$$(a) \overline{x_1 x_2} = \bar{x}_1 + \bar{x}_2$$



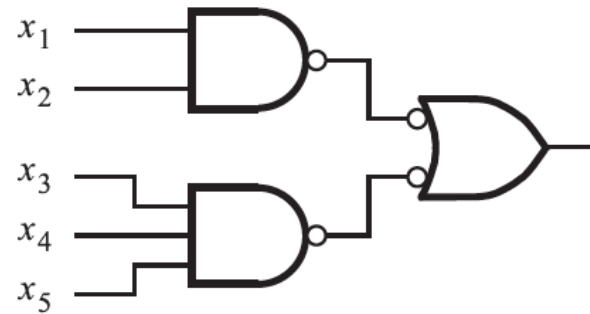
$$(b) \overline{x_1 + x_2} = \bar{x}_1 \bar{x}_2$$

# Converting to NAND



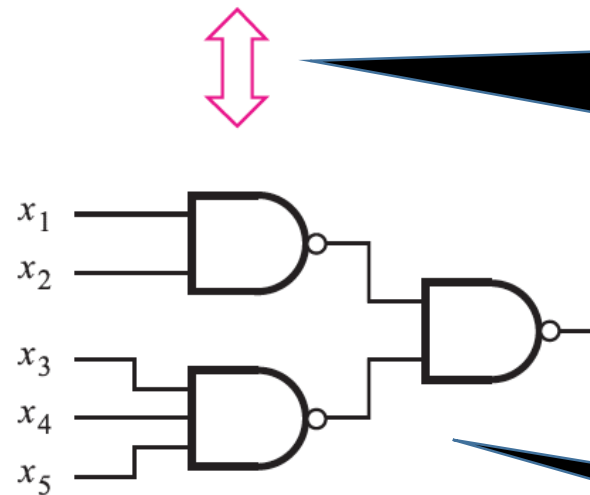
1

- Add bubbles (inverters) between ANDs and OR from 2 sides.
- This doesn't change anything really in the function



2

- Convert the OR gate with 2 bubbles on input to NAND (from previous slide)

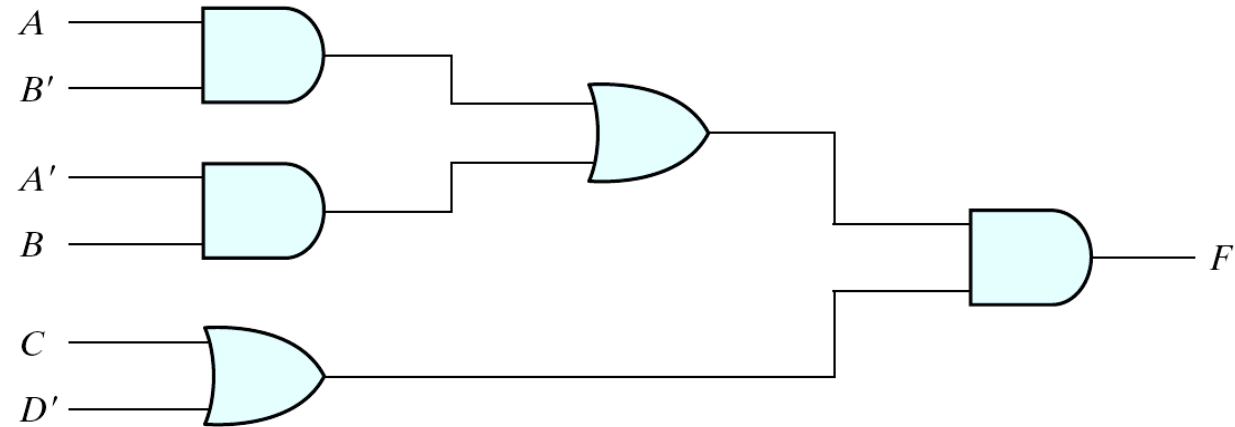


- ALL NAND circuit

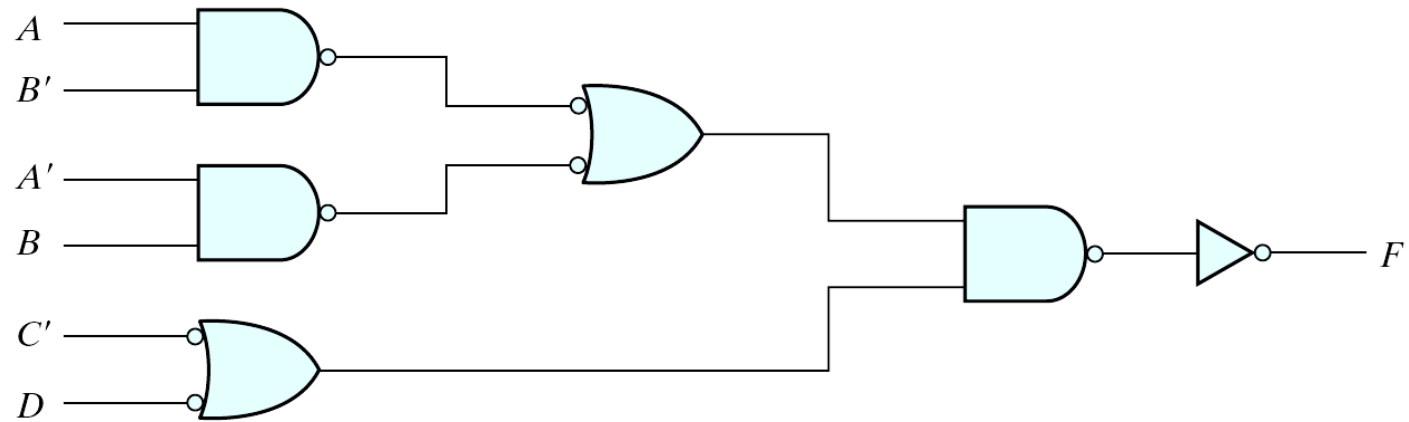
3

Using NAND gates to implement a sum-of-products.

# NAND Implementation



(a) AND-OR gates

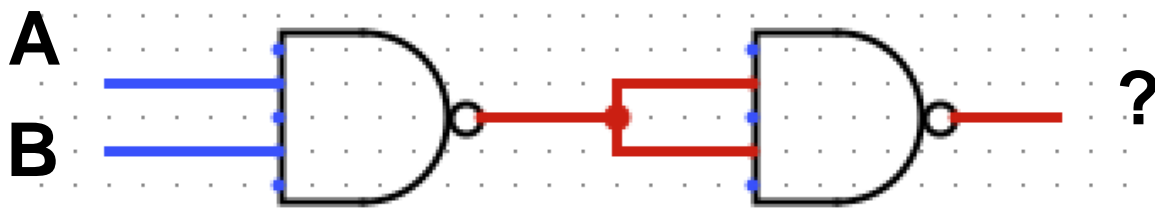
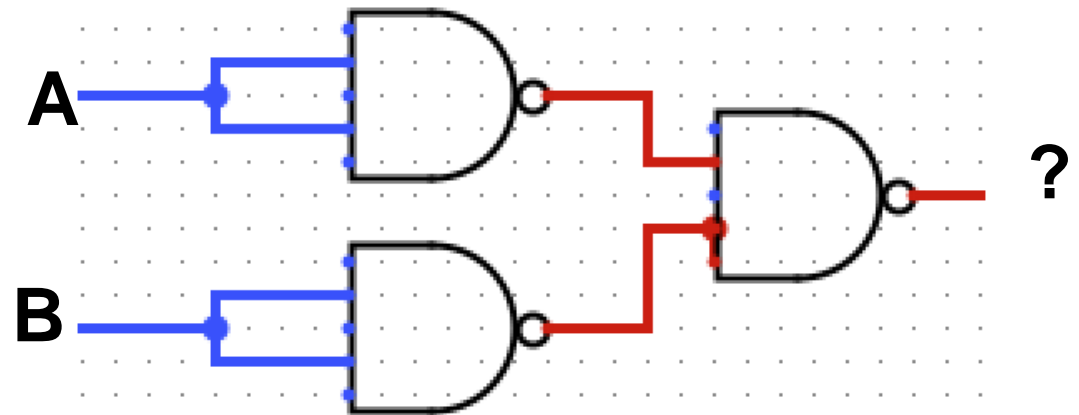
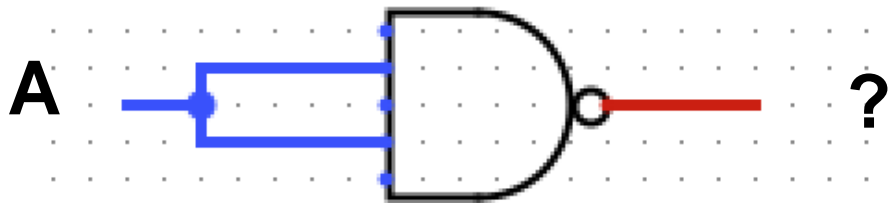


(b) NAND gates

Implementing  $F = (AB' + A'B)(C + D')$



# Can you guess the job of each one?



# Karnaugh-maps (K-maps)

- A systematic way to optimize your boolean expression without the hassle of looking into all theorems.
- General rule: a k-map for each output to try to minimize its SoP expression

# 2-variable K-maps

$x_1$	$x_2$	
0	0	$m_0$
0	1	$m_1$
1	0	$m_2$
1	1	$m_3$

(a) Truth table

$m_0$	$m_1$
$m_2$	$m_3$

(b) Karnaugh map

		$y$	
		0	1
$x$	0	$m_0$ $x'y'$	$m_1$ $x'y$
	1	$m_2$ $xy'$	$m_3$ $xy$

**Figure 4.2** Location of two-variable minterms.

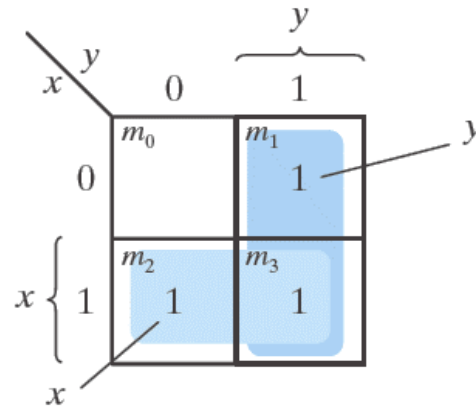
# 2-variable K-maps

- **RULES:**

- Try to group the adjacent ones (1's) horizontally or vertically or both. A don't care value can be used as a 1.
- Find the biggest group you can. Group sizes should be 1, 2, 4, 8, 16 ONLY
- You can re-use grouped ones to help other groups get bigger.

## Example

$$f(x, y) = m_1 + m_2 + m_3$$



$$f = x + y$$

- this group spans 1 for y (so y) and 0 and 1 for x so it's removed.

- When the group spans 0 and 1 for the same variable, we omit this variable
- This group spans 0 and 1 for y but only 1 for x, so it corresponds to x

# 3-variable K-maps

- Note the 11 before the 10
- This only helps in optimization

$m_0$	$m_1$	$m_3$	$m_2$
$m_4$	$m_5$	$m_7$	$m_6$

(a)

		$y$			
		$yz$			
$x$	0	00	01	11	10
		$m_0$ $x'y'z'$	$m_1$ $x'y'z$	$m_3$ $x'yz$	$m_2$ $x'yz'$
1		$m_4$ $xy'z'$	$m_5$ $xy'z$	$m_7$ $xyz$	$m_6$ $xyz'$

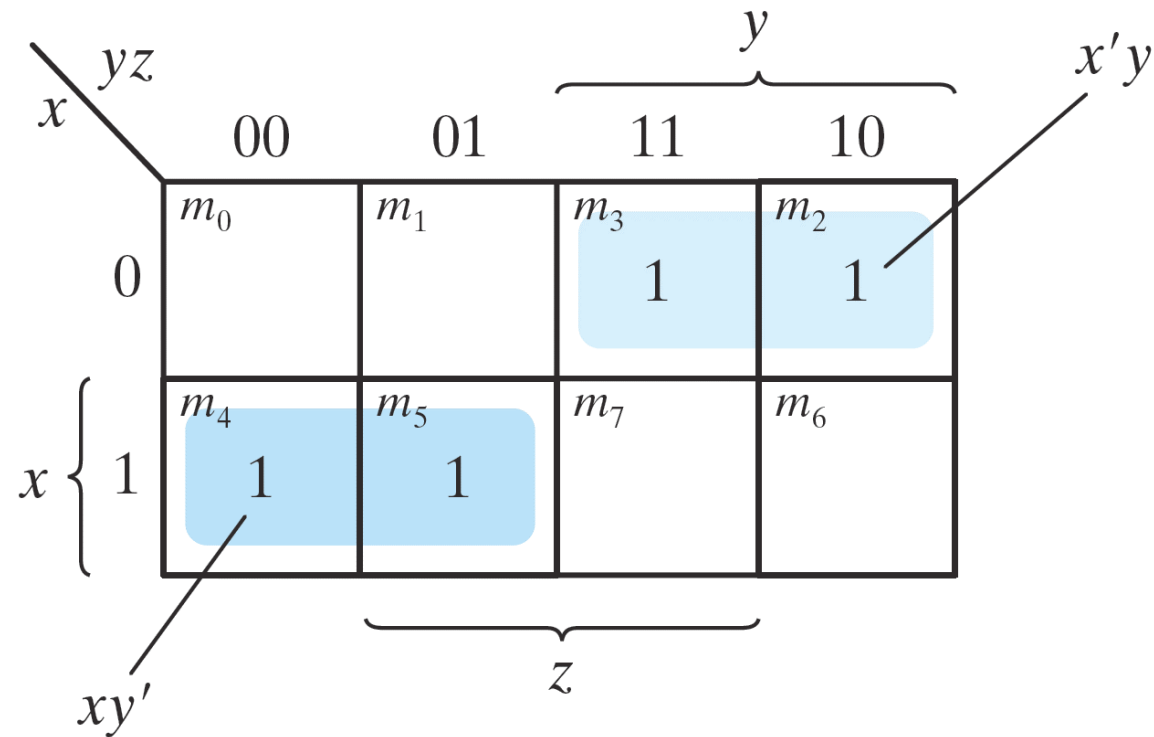
(b)

**Note:** the k-map behaves like a sphere so you can wrap around groups

# 3-variable K-maps

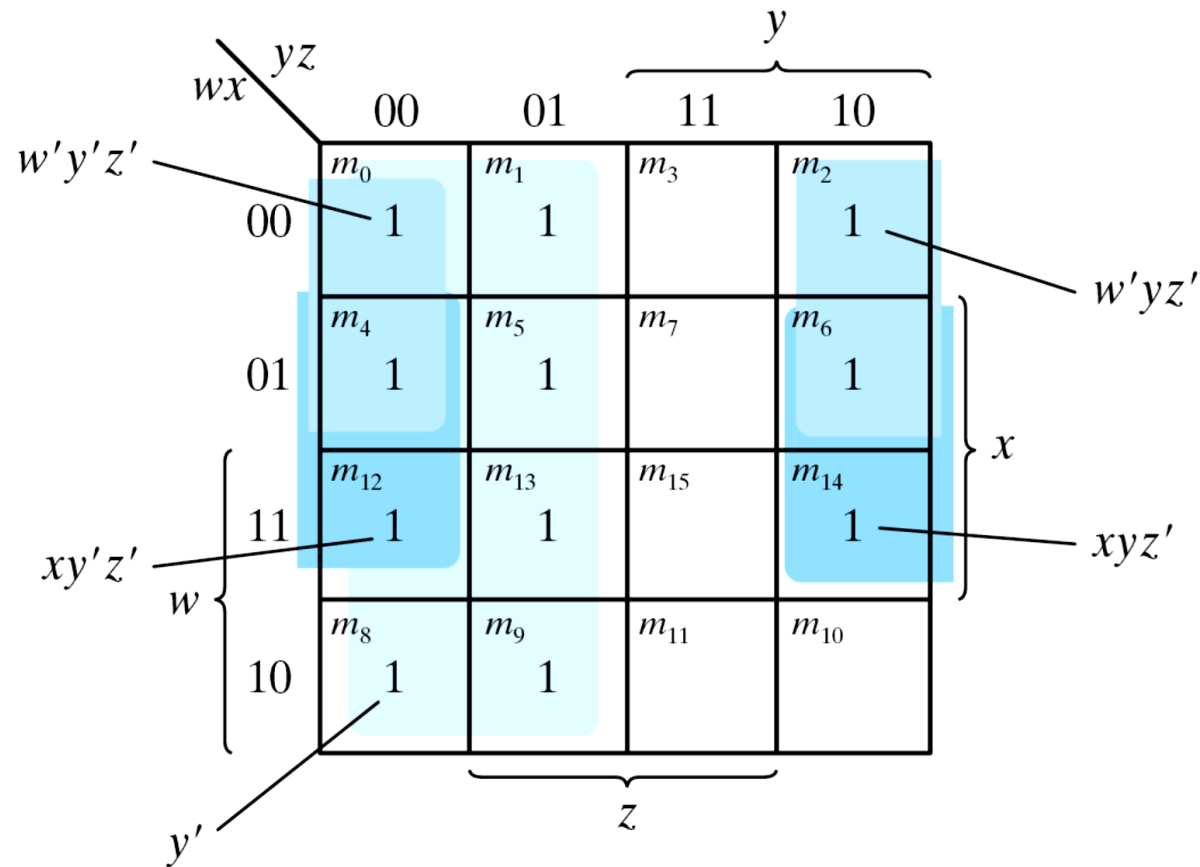
Example

$$F(x, y, z) = \Sigma(2, 3, 4, 5) = x'y + xy'$$



$$F(x, y, z) = x'y + xy'$$

# 4-variable k-maps

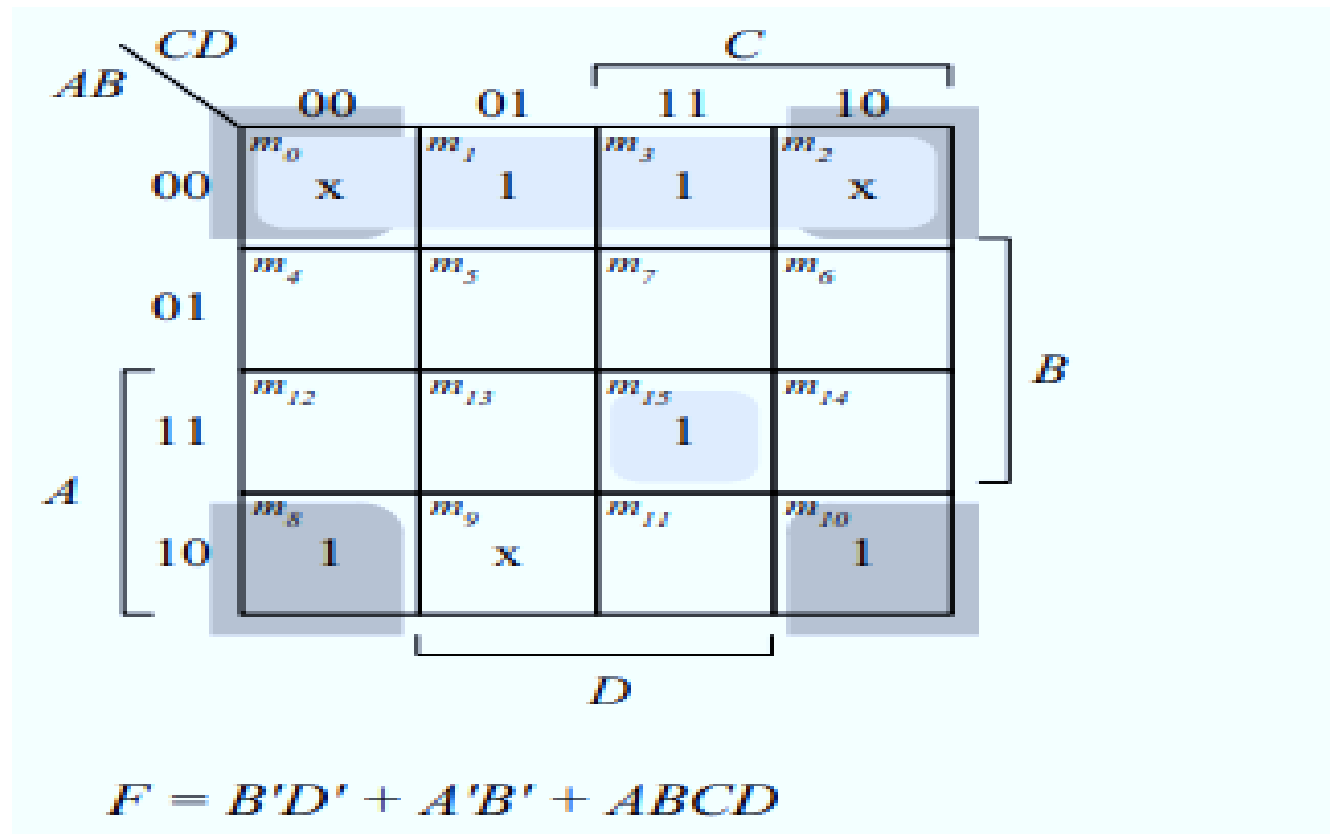


Note:  $w'y'z' + w'yz' = w'z'$   
 $xy'z' + xyz' = xz'$



$$F = y' + w'z' + xz'$$

# Example with don't-care Conditions

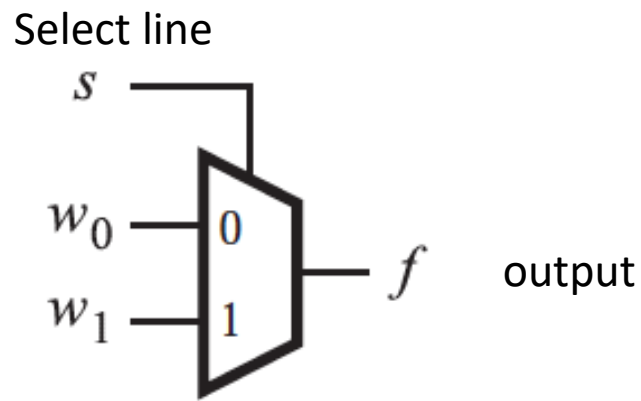




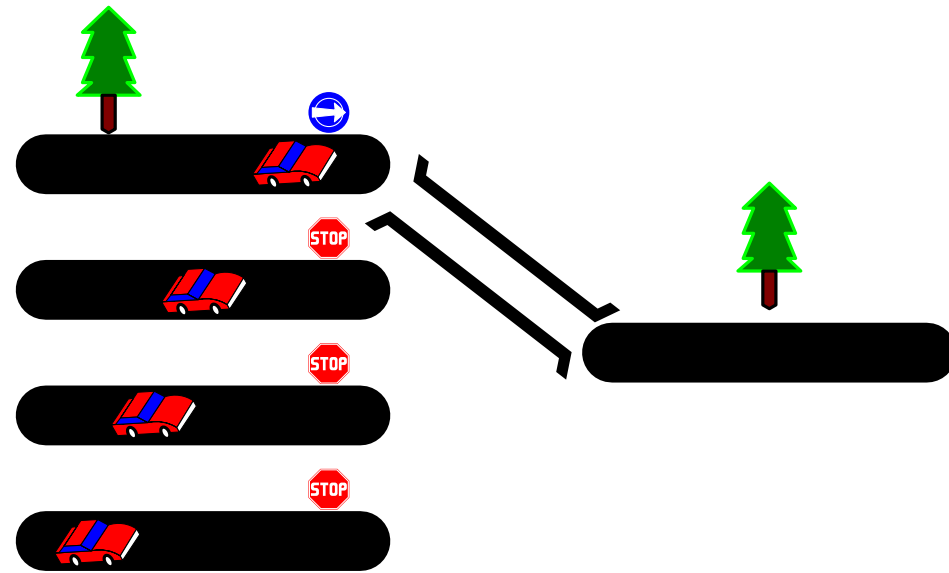
## **Part 4**

# **Combinational circuits building blocks**

# Multiplexers



(a) Graphical symbol



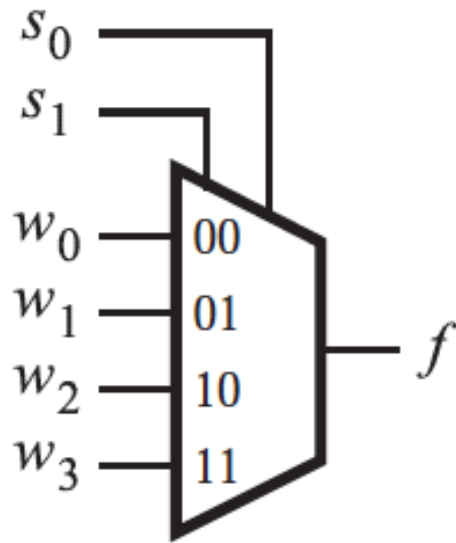
$s$	$f$
0	$w_0$
1	$w_1$

(b) Truth table

- Multiplexers are used to allow one input to go through to the output depending on the select line value
- They are usually 2-to-1, 4-to-1, 8-to-1 and so on.

# Multiplexers (cont.)

- Can you guess the truth table of this mux?

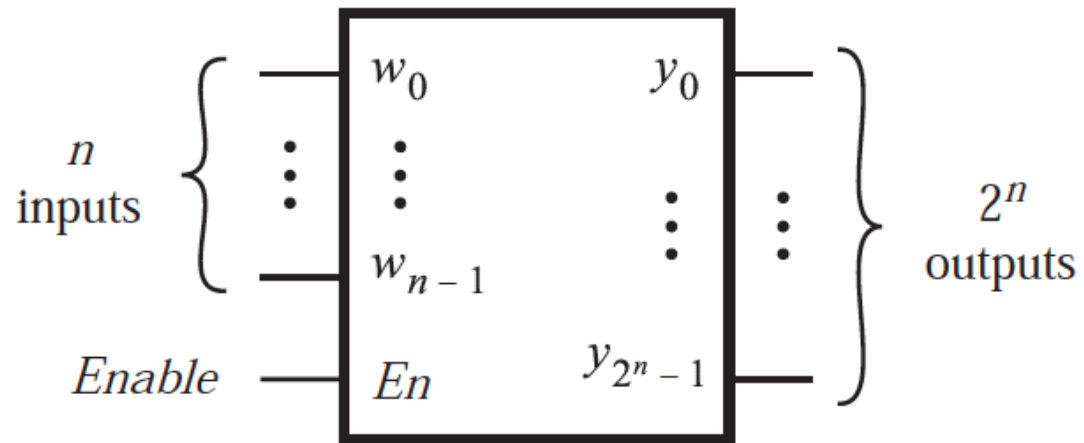


$s_1$	$s_0$	$f$
0	0	$w_0$
0	1	$w_1$
1	0	$w_2$
1	1	$w_3$

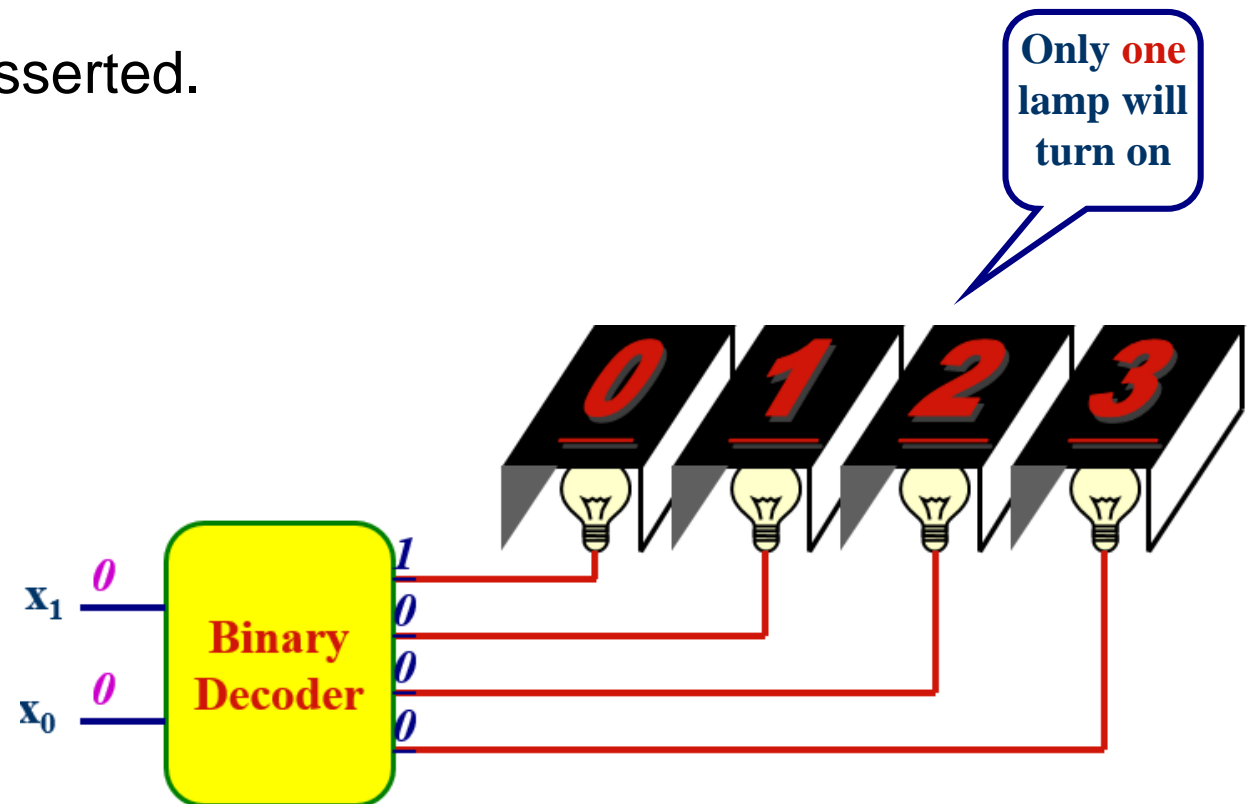
(a) Graphical symbol

# Decoders

- Extract “*Information*” from the code
- With  $n$ -inputs, decoders have  $2^n$  outputs
- Only one output is asserted ( $=1$ ) at any given time, depending on the input and given that Enable is 1
- When Enable  $=0$ , all outputs are not asserted.



**Figure 6.15** An  $n$ -to- $2^n$  binary decoder.

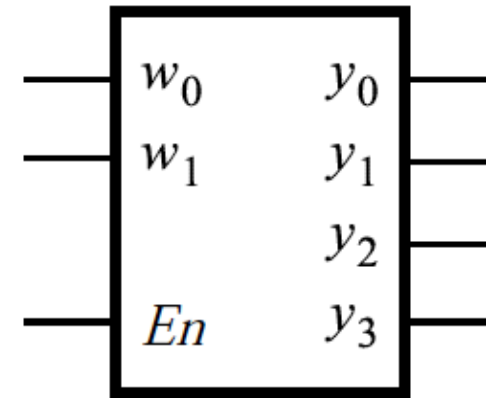


# Decoders (cont.)

- A 2-to-4 decoder

$En$	$w_1$	$w_0$	$y_0$	$y_1$	$y_2$	$y_3$
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1
0	x	x	0	0	0	0

(a) Truth table



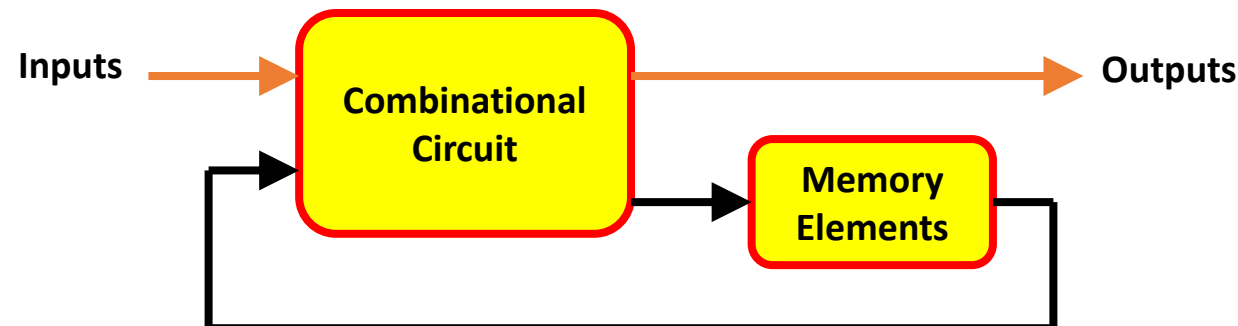
(b) Graphical symbol

## **Part 5**

# **Sequential systems**

# Sequential systems

- Everything we have seen so far is based on combinational systems
- In combinational systems, the output depends on the current input
- In sequential systems, the output depends on the current input and the history of the output/state of the system: **memory** units



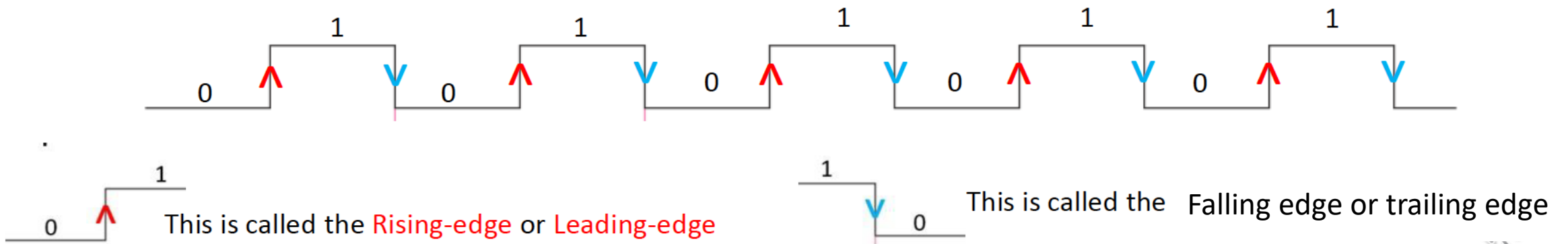
# Sequential systems

- A simple example is the system that controls the sound volume on a TV. The input is the command button: Volume up or Volume down
- The output/state is the sound volume
- The next output/state depends on the given command (up or down) and the current volume
- E.g.: if current volume= 18 and volume up is pressed, next volume will be 19 and it will become the current volume



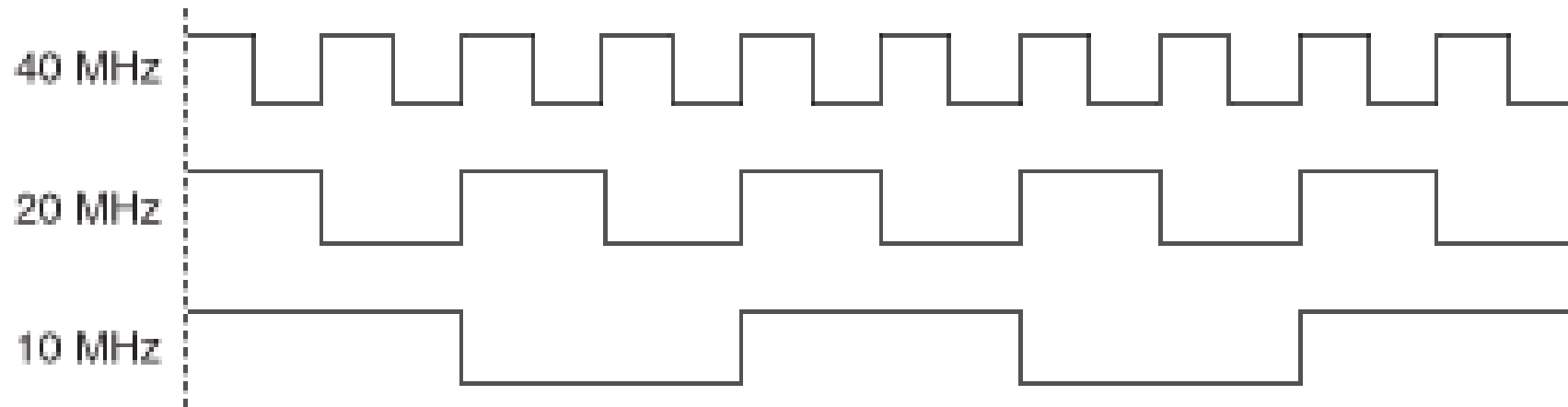
# Concept of the clock (clk)

- A periodic signal that alternates between high and low
- Sequential elements are usually triggered by the clock signal to do a job
- They can triggered by the rising/leading edge (going from 0 to 1) or the falling edge (from 1 to 0)



# Concept of the clock (clk)

- Clocks are usually used also to tell how fast are systems (or processors) by increasing their frequencies



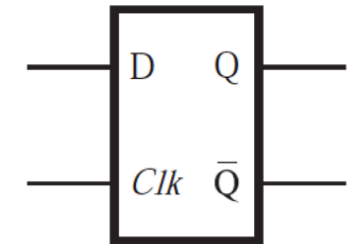
# D-Flip flop

- Two inputs: D and the clk
- Two outputs: Q and its inverse Q'
- This D-flip flop is leading/positive edge triggered
- When the clock is 0, the D flip flop retains its output Q. ( $Q_{t+1} = Q_t$ )
- When the clock becomes 1, the value of D **just before the edge** is propagated to Q. ( $Q_{t+1} = D$ )
- Q stays unchanged until the next edge (even if D changed in the middle of the clock period)

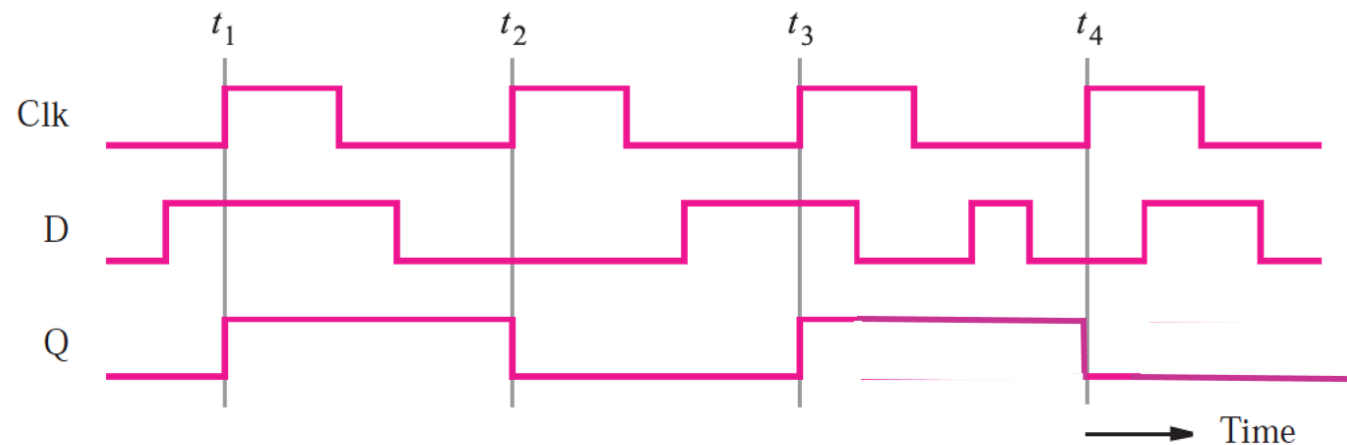
Clk	D	$Q(t+1)$
0	x	$Q(t)$
1	0	0
1	1	1

**Reset**  
**Set**

(b) Characteristic table



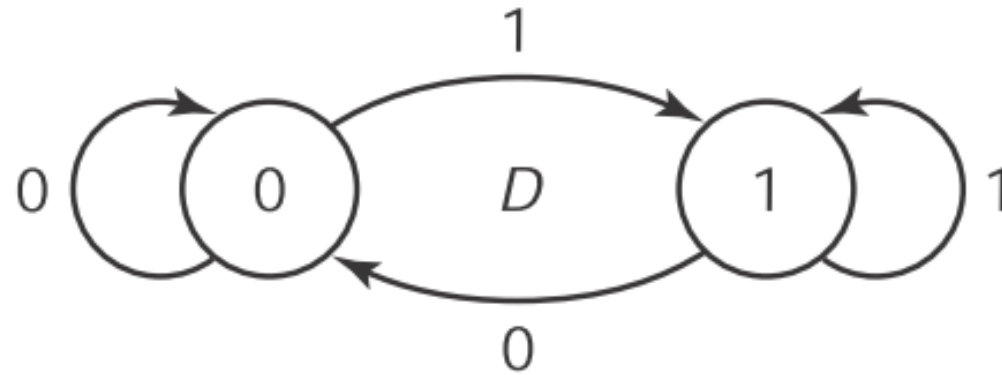
(c) Graphical symbol



(d) Timing diagram

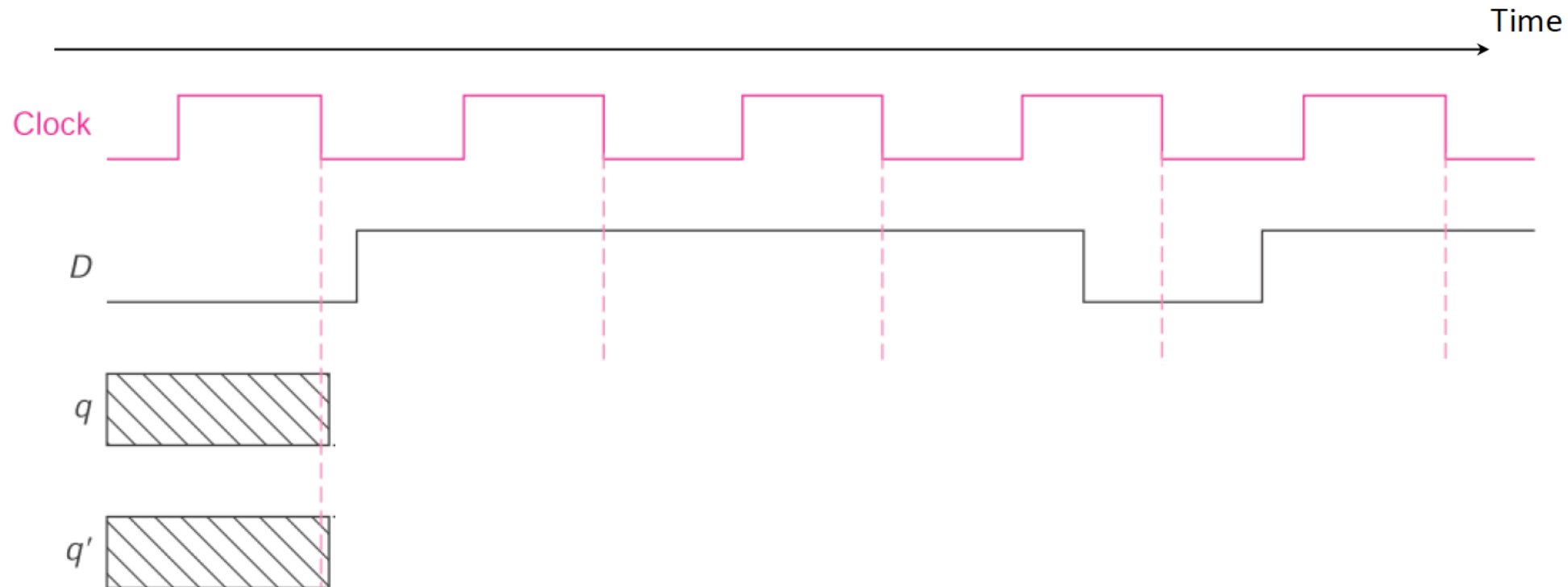
# D-Flip flop

- The behavior of the D-flip flop can be depicted in the following state diagram



# Can you complete this timing diagram?

- The barred squares denote that the output was unknown before the first edge.

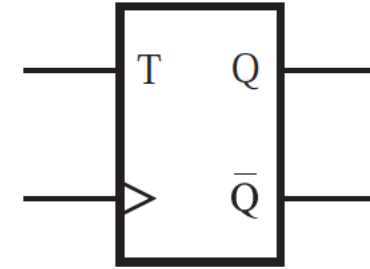


# T-Flip flop

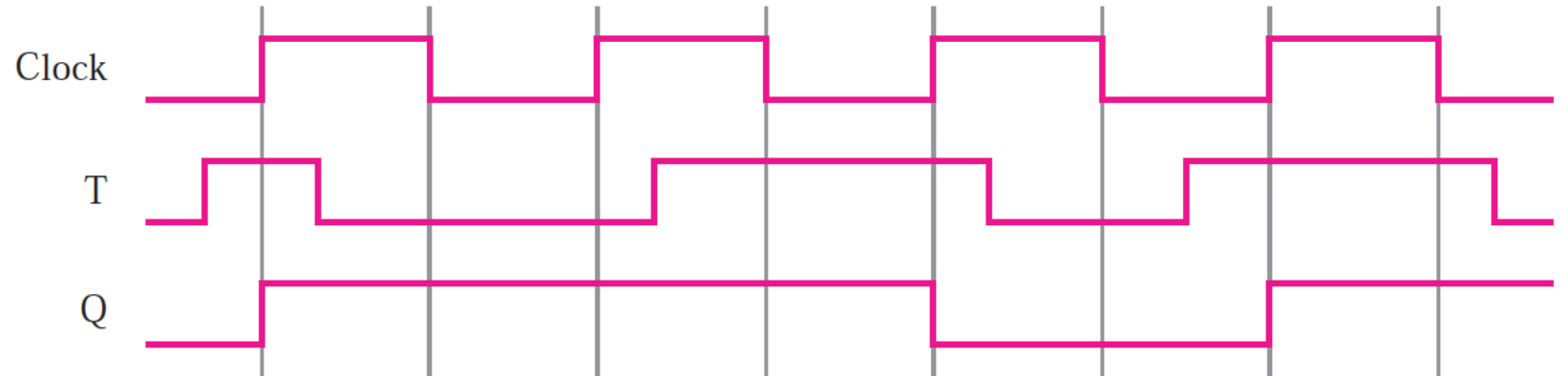
- **T**oggles its state when  $T=1$
- Keep same output if  $T=0$

T	$Q(t+1)$
0	$Q(t)$
1	$\bar{Q}(t)$

(b) Characteristic table



(c) Graphical symbol

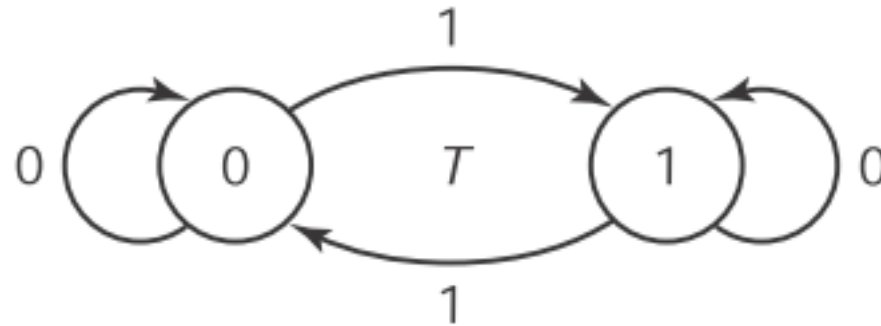


(d) Timing diagram

# T-Flip flop (cont.)

- the truth table can be seen as follows
- $q^*$  is the new output,  $q$  is the old

$T$	$q$	$q^*$
0	0	0
0	1	1
1	0	1
1	1	0



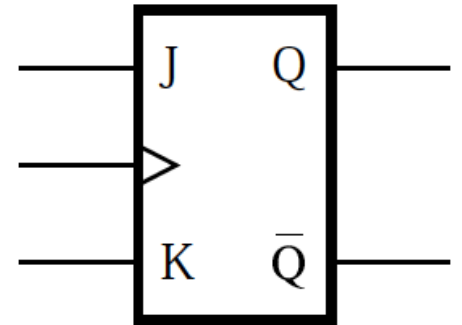
$$q^* = T \oplus q$$

# JK-Flip flop

- Retains its value when J and K are 0
- Toggles when both are 1 (like T-flip flop)
- $Q = J$  otherwise (like a D-flip flop)

J	K	$Q(t+1)$	
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$\bar{Q}(t)$	Complement

(b) Characteristic table



(c) Graphical symbol

**Figure 7.17** JK flip-flop.

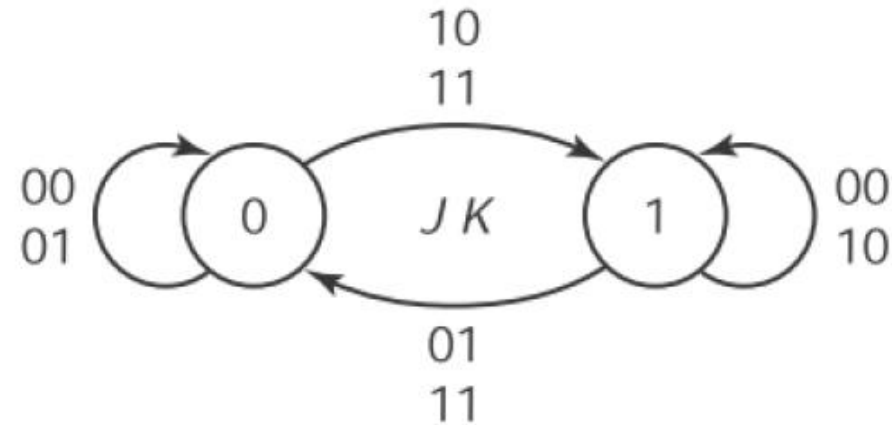


# JK-flip flop truth table

- How did we get that?
- K-map?

$J$	$K$	$q$	$q^{\star}$
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	0

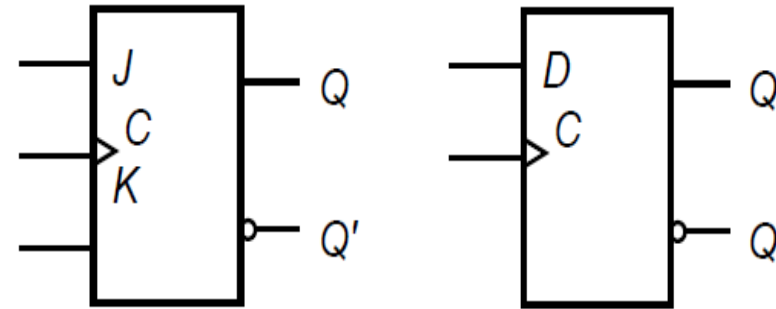
$$q^{\star} = Jq' + K'q$$



# Flip flop Characteristic Tables

*JK* Flip-flop

<i>J</i>	<i>K</i>	$Q(t+1)$	
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$Q'(t)$	Complement



*D* Flip-flop

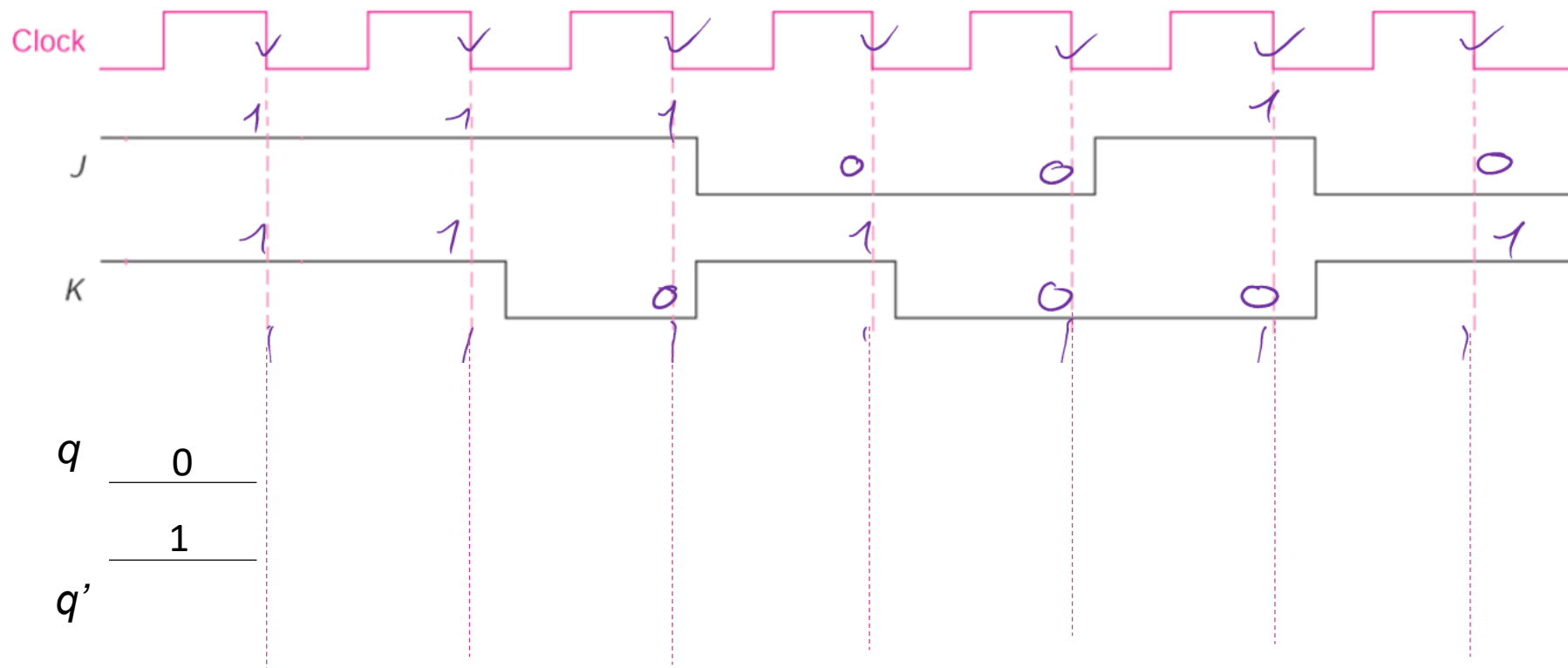
<i>D</i>	$Q(t+1)$	
0	0	Reset
1	1	Set

*T* Flip-flop

<i>T</i>	$Q(t+1)$	
0	$Q(t)$	Hold
1	$Q'(t)$	Toggle

# Exercise:

- Complete the following timing diagram of a trailing-edge triggered JK flip flop. PS:  $q$  is 0 at the beginning.



$J$	$K$	$q^*$
0	0	$q$
0	1	0
1	0	1
1	1	$q'$